IX Jornadas de Enseñanza Universitaria de la Informática

THOMSON

Australia ● Canada ● Mexico ● Singapore ● Spain ● United Kingdom ● United States

IX Jornadas de Enseñanza Universitaria de la Informática

Cádiz, del 9 al 11 de julio de 2003

Datos de catalogación bibliográfica

IX Jornadas de Enseñanza Universitaria de la Informática
Jenui 2003

Thomson Paraninfo, S.A., Madrid, 2003.

ISBN:
Materia: Informática 681.3
Enseñanza superior 378

Formato:240x170 Páginas 650

Diseño de portada: Nuria Hurtado Rodríguez
Maquetación: Inmaculada Medina Bulo y Francisco Palomo Lozano

Primera edición: julio de 2003

I.S.B.N.:
Depósitio legal:
Imprime:

ACTAS DE LAS IX JORNADAS DE ENSEÑANZA UNIVERSITARIA DE LA INFORMÁTICA (JENUI 2003)

ORGANIZADAS POR:

Departamento de Lenguajes y Sistemas Informáticos
Universidad de Cádiz

ENTIDADES COLABORADORAS:

Ayuntamiento de Cádiz
Ministerio de Ciencia y Tecnología
Junta de Andalucía
Consejo Social de la UCA
Vicerrectorado de Profesorado de la UCA
Vicerrectorado de Extensión Universitaria de la UCA
Escuela Superior de Ingeniería de Cádiz
Novática
Colegio Oficial de Ingenieros Informáticos del País Vasco
Agencia de viajes El Corte Inglés
IBERIA

COMITÉ DE PROGRAMA

PRESIDENTA

Rosalía Peña Ros	(Universidad de Alcalá)

Joaquín Ezpeleta Mateo	(Universidad de Zaragoza)
Jesús García Molina	(Universidad de Murcia)
Faraón Llorens Largo	(Universidad de Alicante)
Cristóbal Pareja Flores	(Universidad Complutense de Madrid)
Edmundo Tovar Caro	(Universidad Politécnica de Madrid)
Miguel Valero García	(Universitat Politècnica de Catalunya)

COMITÉ ORGANIZADOR
UNIVERSIDAD DE CÁDIZ

PRESIDENTE

Francisco Palomo Lozano

COORDINADORES

Juan José Domínguez Jiménez
Antonia Estero Botaro

Esther Gadeschi Díaz
Mª. Teresa García Horcajadas
Nuria Hurtado Rodríguez
Inmaculada Medina Bulo
Elena Orta Cuevas
Carlos Rioja del Río
Mercedes Ruiz Carreira
Esther Lydia Silva Ramírez

COLABORADORES EN EL PROCESO DE REVISIÓN

José Joaquín Aguilera García	UJAEN	Esperanza Marcos Martínez	URJC
Sergio Albiol	UNIZAR	Mercedes Marqués Andrés	UJI
Carlos Aliagas	URV	Olga Marroquín Alonso	UCM
Pedro J. Álvarez	UNIZAR	Antonio Martí	UPV
Fernando Álvarez García	UNIOVI	F. José Martínez Domínguez	UNIZAR
Joan Aranda López	UPC	Margarita Martínez Santamarta	URJC
Sandra Baldassarri	UNIZAR	Gloria Martínez Vidal	UJI
José Ramón Balsas Almagro	UJAÉN	Manuel Mejías Risoto	US
José Ángel Bañares Bañares	UNIZAR	Marcos Menárguez Tortosa	UM
Pedro Blesa Pons	UPV	José Merseguer Hernáiz	UNIZAR
Juan A. Botía Blaya	UM	Roc Messeguer	UPC
José Manuel Burgos	UPM	José Miró Julià	UIB
Mateo Camps	UPCO	Andrés Molina	UJAÉN
Juan Carlos Cano Escriba	UPV	M. Angels Moncusi	URV
Pedro José Clemente Martín	UNEX	Antonio Moreno Ribas	URV
Alberto de la Encina Vara	UCM	Susana Muñoz Hernández	UPM
Antonio J. de Vicente	UAH	Raúl Murciano	UEM
Luis Díaz de Cerio Ripalda	UPC	Juan Manuel Murillo Rodríguez	UNEX
Juan José Escribano Otero	UEM	Juan José Navarro Guerrero	UPC
Luis Fernández Muñoz	UPM	Leandro Navarro Moldes	UPC
Joseph Fernández Ruzafa	UPC	Iñaki Iñigo Ochoa de Chinchetru	UNIZAR
José Fortes Gálvez	ULPGC	Carmen Nieves Ojeda	ULPGC
Isabel Gallego Fernández	UPC	Santiago Ortego Carazo	UPC
Javier Galve Francés	UPM	María José Ortín Ibáñez	UM
José A. Gamez Martin	UCLM	Juan José Pantrigo	URJC
Maria José García	UEM	Rosalía Peña Ros	UAH
Montse García	URV	Ángel F. Perles Ivars	UPV
Pedro García López	URV	José Poveda	UV
Gines García Mateos	UM	Alberto Prieto	UGR
Jesús García Molina	UM	Mar Pujol López	UA
Piedad Garrido Picazo	UNIZAR	Robert Rallo	URV
María José Gil	DEUSTO	Jorge Ramió Aguirre	UPM
Domingo Giménez Canovas	UM	Isabel Ramos	US
Estrella Gómez	UEM	Miguel Rebollo Pedruelo	UPV
María Engracia Gómez	UPV	Jairo Rocha	UIB
Mercedes Gómez Albarrán	UCM	Fernando Rubio Diez	UCM
Alberto Gómez Mancha	UNEX	Francisco Ruiz González	UCLM
Pilar González Ferez	UM	Jesús Salido Tercero	UCLM
Julia González Rodríguez	UNEX	Marcial Samper Asensio	UA
Antoni Grau	UPC	Fermín Sánchez	UPC
Ángel Grediaga	UA	Fernando Sánchez Figueroa	UNEX
Diego Gutiérrez Pérez	UNIZAR	Rosana Satorre Cuerda	UA
Ángel Herranz Nieva	UPM	Clara María Segura Díaz	UCM
Inés Jacob	DEUSTO	Diego Sevilla Ruiz	MURCIA
M. Carmen Juan	UPV	Álvaro Suárez Sarmiento	ULPGC
Lenin G. Lemus Zúñiga	UPV	Lourdes Tajes Martínez	UNIOVI
Fernando Llopis Pascual	UA	Jaime Urquiza Fuentes	URJC
Faraón Llorens Largo	UA	Maite Urretavizcaya Loinaz	EHU
Natalia López Barquilla	UCM	Alberto Verdejo López	UCM
Adolfo Lozano Tello	UNEX	María Teresa Villalba de Benito	UEM
Elsa Macías López	ULPGC	Ferrán Virgós	UPC
Antonio Maña	UMA	F. Javier Zarazaga Soria	UNIZAR

Enseñar es aprender dos veces

Joseph Joubert

Presentación

El presente volumen recoge los trabajos aceptados para su presentación en las IX Jornadas de Enseñanza Universitaria de la Informática (JENUI 2003), celebradas en Cádiz los días 9, 10 y 11 de julio de 2003 organizado por el Departamento de Lenguajes y Sistemas Informáticos de la Universidad de Cádiz. Posiblemente, la informática es una de las disciplinas que está experimentando un cambio más rápido y profundo en la actualidad y, al igual que sus contenidos y técnicas, los métodos con que se enseña, están transformándose drásticamente. De ahí la necesidad y la utilidad de este congreso. Estas actas pretenden ser una herramienta útil para quienes enseñamos informática en la universidad.

En esta edición las áreas estratégicas elegidas han sido Enseñanza y uso de métodos formales en los estudios universitarios de informática y Educar/formar profesionales informáticos, relacionadas con ellas se han organizado una conferencia impartida por el profesor J S. Moore, Decano de la Facultad de Ciencias de la Computación de la Universidad de Texas en Austin y un debate en el que participarán profesores universitarios de Informática y profesionales que desarrollan su labor en empresas del sector, que será moderado por el profesor Luis Fernández Muñoz de la Universidad Europea de Madrid. Queremos agradecer a D. José Antonio Martínez Ruiz, decano del Colegio Oficial de Ingenieros en Informática de Euskadi, y a D. Juan José Monedero Rojo, director técnico de SERVIPORT Andalucía su participación en este debate.

Asimismo, dado que nos encontramos en pleno proceso de integración de nuestro sistema universitario en el espacio europeo de educación superior, hemos considerado interesante abordar estos temas. En este sentido se ha organizado una conferencia que trata sobre los créditos ECTS impartida por el profesor Miguel Valero García.

Siguiendo la línea de ediciones anteriores, se ha trabajado en el perfeccionamiento del proceso de revisión, mediante la elaboración y publicación de directrices que ayuden a los autores durante la preparación de sus trabajos y a los revisores para unificar los criterios de evaluación. En esta difícil tarea, los revisores han efectuado un esfuerzo mayor, cumplimentando informes técnicos más completos, justificados y detallados que en años anteriores. Por tanto, debemos agradecer el esfuerzo desinteresado de estos expertos anónimos, cuyos múltiples e interesantes comentarios han contribuido muy positivamente a incrementar la calidad de los trabajos. Como en las últimas ediciones, el trabajo de selección de las ponencias ha sido doblemente anónimo. Cada trabajo ha sido evaluado por al menos 3 revisores cuyos comentarios, en numerosas ocasiones, han sido de gran ayuda a la hora de mejorar la versión final. En casos de evaluaciones contradictorias se ha solicitado una cuarta opinión y, en los trabajos en que se ha considerado necesario incorporar alguna mejora a propuesta de los revisores, se han vuelto a revisar las versiones finales.

El resumen estadístico de las contribuciones es el siguiente: se recibieron 13 trabajos como demostraciones y 107 propuestas de ponencias, procedentes de 37 universidades, 2 de ellas cubanas, 1 alemana, 1 checa y 33 españolas, así como profesionales de distintas empresas. Teniendo en cuenta las evaluaciones, el Comité de Programa ha aceptado 7 demostraciones, y 69 ponencias, 4 propuestas de ponencias se recondujeron como demostraciones. Globalmente, estas cifras suponen un índice de aceptación de un 66%, cifra algo superior a la de la edición anterior.

Una importante novedad de esta edición de JENUI ha sido el establecimiento de una colaboración con la editorial Thomson para la publicación del texto íntegro de los trabajos presentados en estas Jornadas. La mejora de la calidad del documento y de su difusión ha supuesto un incentivo más para todos los participantes . Nos consta que éste es uno de los motivos fundamentales por los que hemos apreciado mayor calidad técnica y esmero en la presentación de los trabajos propuestos. Queremos por ello agradecer a la editorial Thomson y en especial a D. Andrés Otero esta oportunidad y la confianza que desde el principio han depositado en nosotros.

Consideramos que es fundamental acercar la universidad a la empresa, y las asociaciones profesionales son un buen vehículo para este acercamiento. ATI, en su revista Novática, ha difundido la convocatoria de las Jornadas y publicará dos trabajos seleccionados de entre los presentados en su columna enseñanza universitaria de la informática. Debemos agradecer a Rafael Fernández Calvo esta colaboración y las facilidades que nos ha dado.

Todo ello ha dado lugar a una mejora en la calidad en los contenidos y la presentación de los trabajos, por la que se ha de felicitar a los autores y por la que, también nosotros, nos sentimos orgullosos.

Queremos agradecer la colaboración prestada por distintos órganos de la Universidad de Cádiz, en especial al Vicerrectorado de Profesorado y al Consejo Social, así como al Excmo. Ayuntamiento de Cádiz.

La Asociación AENUI ha delegado en el Comité de Programa la selección de la sede de las próximas Jornadas de JENUI. Ha sido difícil dado el número y calidad de las candidaturas que hemos recibido. Esto es buena señal. Finalmente, la sede para las JENUI 2004 será la Escuela Politécnica Superior de la Universidad de Alicante (la información relativa a las nuevas jornadas estará accesible a través de la siguiente URL: http://www.dccia.ua.es/jenui2004).

Desde estas líneas damos la bienvenida a todos, esperando que sean unos días de grata y fructífera convivencia de la que los verdaderos beneficiados sean nuestros alumnos.

Cádiz, 9 de julio de 2003

Comité de Programa y Comité Organizador

CONTENIDOS

Ponencias

Demostraciones

ÍNDICE DE AUTORES

Ponencias

Tema estratégico

Enseñanza y uso de métodos formales en los estudios universitarios de informática

Métodos formales en programación: ¿desmitificar para motivar?

Inés Jacob
Facultad de Ingeniería
Universidad de Deusto
Apartado 1, 48080 Bilbao
e-mail: ines@eside.deusto.es

Resumen

Los métodos formales gozan de una imagen entre alumnos y profesores que plantea dificultades en el aprendizaje y la docencia de su aplicación a la programación. Son reconocidas las dificultades de aplicar en la industria mediante métodos formales los razonamientos matemáticos que son abstractos por naturaleza. Se presentan en este trabajo algunas reflexiones que tienen como resultado la identificación de la necesidad de encontrar fuentes de motivación para los alumnos. Se propone la incorporación de actividades en la preparación y docencia de asignaturas que tratan métodos formales que contribuyan a su innovación pedagógica.

1. La experiencia docente

Las reflexiones que en este artículo presentamos vienen provocadas por la experiencia docente de la asignatura Metodología de la Programación impartida en la Universidad de Deusto en segundo de Ingeniería Técnica de Informática de Gestión.

Se basa en los conocimientos de lógica formal y de programación que el alumno adquiere en primero. Es la primera asignatura de la titulación en la que se trata la aplicación de métodos formales al desarrollo de software.

Con el fin de:

- cubrir el contenido fijado en el plan de estudios para esta asignatura troncal,
- completar el itinerario del alumno por las asignaturas del área de Programación y Lenguajes,
- fijar y afianzar las bases para el aprendizaje de contenidos de disciplinas relacionadas y abordadas en parte en cursos posteriores.

se fijan los siguientes objetivos generales para la asignatura:

- familiarizarse con la verificación y derivación formal de programas
- razonar formalmente el diseño recursivo de funciones
- entender la recursividad y su relación con los programas iterativos.

Los objetivos específicos han sido elegidos en función de estos objetivos generales de forma que el alumno identifique en ellos los logros que ha de conseguir a lo largo de la asignatura. Coinciden con los aspectos que serán evaluados.

Este diseño de objetivos [14], da lugar al programa de la asignatura, dividido en seis capítulos que tratan:

1. La motivación para el estudio de métodos formales en programación y los conceptos básicos empleados durante el curso.
2. El lenguaje utilizado, con la descripción de los métodos de verificación y derivación de cada una de las instrucciones.
3. El principio de inducción aplicado a la demostración de teoremas inductivos y a la validación de bucles y funciones recursivas**.**
4. La derivación de programas recursivos.
5. La transformación de programas por inmersión.
6. La derivación de funciones iterativas.

El equipo docente de la asignatura considera que este diseño puede permanecer estable, en líneas generales, durante un tiempo. Tras varios años dedicados a su ajuste estima por lo tanto que es el momento de hacer una reflexión crítica y constructiva sobre la asignatura.

2. Sobre la motivación

Las calificaciones obtenidas por los alumnos de Metodología de la Programación no son muy

buenas, siendo además bastante elevado el porcentaje de alumnos matriculados que ni siquiera se presentan a examen.

Se plantea como prioritario en este momento mejorar los resultados académicos obtenidos en esta asignatura, incidiendo en la motivación del alumno como motor de su aprendizaje.

Las dos siguientes consideraciones justifican la necesidad de proponerse que el alumno esté motivado como medio para mejorar su rendimiento:

- La motivación es la responsable de iniciar, mantener y dirigir la conducta de un individuo hacia la consecución de una meta u objetivo [18].
- La calificación constituye un indicador del grado de consecución de los objetivos planteados en una asignatura, siempre y cuando las herramientas y métodos de evaluación sean adecuados.

Centrar nuestro interés en la motivación no supone ignorar la importancia que los métodos y herramientas de evaluación tienen. Somos plenamente conscientes de que no es trivial lograr una evaluación que se ajuste realmente a los objetivos de la asignatura y valore el proceso desarrollado por el estudiante reconociéndolo en la calificación final [15].

En cualquier caso la motivación y la evaluación, no deberían ser consideradas independientemente dadas las influencias mutuas existentes entre amabas.

Los estudiantes no motivados no aprenden [16] y por lo tanto los profesores deben encontrar la forma de motivar a sus alumnos. Pensar en los conceptos castigo, recompensa e incentivo relacionados con la motivación extrínseca no es suficiente y se hace necesario estudiar las vías para aumentar la motivación intrínseca. Se trata de lograr que los alumnos tengan interés por la asignatura, pasando si es necesario por un estado anterior de curiosidad.

3. Posibles orígenes de las dificultades

Nos parece fundamental reflexionar sobre las particularidades de la asignatura, que se derivan de que los métodos explicados y aplicados son formales.

3.1. Las matemáticas y la informática

La definición "Método formal es cualquier técnica que trate la construcción y/o el análisis de modelos matemáticos que contribuyen a la automatización del desarrollo de sistemas informáticos" [10] podría ser útil para explicar a los estudiantes el sentido de estudiar métodos formales, para contestar al tan frecuente "para qué sirve esto".

Quedaría además patente la relación de la asignatura con las matemáticas, con lo que el anclaje de esta asignatura en otras de cursos precedentes quedaría explícito.

El estudiante de informática debería conocer los motivos por los que las matemáticas son importantes en su formación [12]:

1. El software es abstracto.
2. El uso preciso de notaciones y símbolos es propio tanto de las matemáticas como de la ingeniería del software.
3. Las matemáticas son necesarias para modelar sistemas de software.
4. Muchos dominios de aplicación (ingeniería ciencia, economía, etc.) tienen bases matemáticas.
5. Es esencial aplicar razonamiento matemático a los sistemas informáticos.

La informática es una ciencia matemática, abstracta. De hecho, los estudiantes, aunque a veces no son muy conscientes de ello, comienzan sus estudios utilizando lenguajes formales (los lenguajes de programación tienen una sintaxis y una semántica formalmente definida) y utilizando fórmulas matemáticas (programas) para resolver problemas con un ordenador [1].

3.2. Los métodos formales y la programación

En el conjunto de asignaturas que los alumnos cursan en el área de programación, Metodología de la Programación se distingue de las demás porque

- los programas obtenidos no pueden probarse por ejecución, pues se utiliza un pseudocódigo,
- contradice en cierta forma el modo en el que los estudiantes han aprendido a programar en su primer año de carrera,
- los métodos que se explican no se aplican habitualmente en otras asignaturas.

Estas características no sirven precisamente de refuerzo para el estudiante que, atendiendo a los resultados obtenidos de las encuestas realizadas para evaluar la docencia, considera que esta asignatura es una de las más difíciles.

Esta asignatura parece quedar así aislada de sus compañeras del área de programación cuando en realidad tiende un puente entre las matemáticas y la programación tradicional, apoyándose en los conocimientos que de esas materias los alumnos han adquirido en el curso precedente (Figura 1).

Creemos necesario que los estudiantes puedan apreciar realmente los vínculos entre la programación y las matemáticas tradicionales a través de la aplicación de los métodos formales a la programación.

Aunque en los planes de estudio de informática se da importancia a las matemáticas, normalmente aparecen aisladas, no aplicadas al desarrollo de software [2].

Algunos de los contenidos de las asignaturas de primer curso: lógica (implicación, negación, cuantificación, principio de inducción), teoría de conjuntos (unión, intersección, pertenencia), álgebra de Boole (and, not, or, leyes de Morgan), se aplican directamente en Metodología de la Programación.

Sería conveniente [17] que estos conceptos se explicaran aplicados a la programación o a otros campos de la ingeniería del software.

Si esto no fuera posible habría que ver la manera de establecer el vínculo desde 2º curso hacia esas asignaturas del curso precedente.

Los estudiantes tienen presente desde el principio la relación con las asignaturas de programación, pero se hace necesario ahondar en los vínculos que se pueden establecer con asignaturas posteriores. Sería así mismo deseable transcribir algunos de los programas obtenidos en pseudocódigo a alguno de los lenguajes de programación conocidos por los estudiantes.

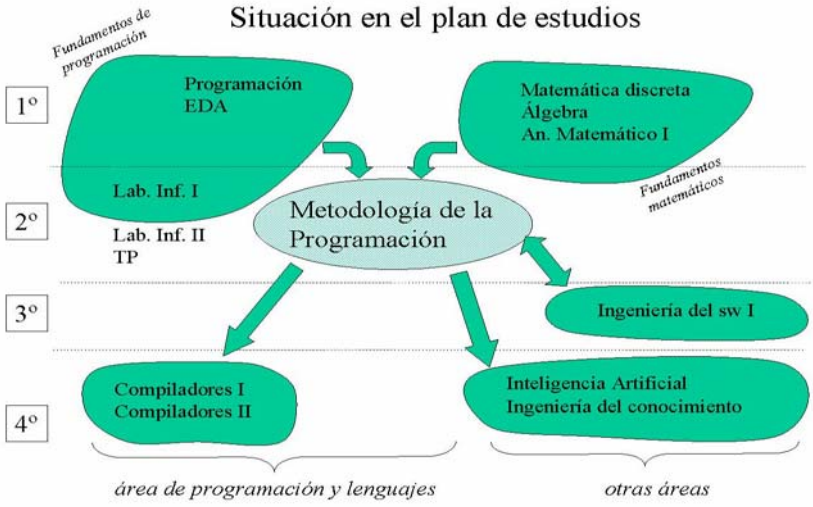


Figura 1. Situación de la asignatura Metodología de la Programación en los planes de estudios de Ingeniero Técnico en Informática de Gestión y de Ingeniero en Informática en la Universidad de Deusto

3.3. Aplicaciones en la industria

El uso de los métodos formales en la industria no está tan extendido como sería deseable desde el punto de vista de sus defensores. Se han estudiado las causas [7][8][9][13] de esta situación, entre las que señalamos:

- El desarrollo de herramientas que apoyen la aplicación de métodos formales es complicado y los programas resultantes son incómodos para los usuarios.
- Los investigadores por lo general no conocen adecuadamente la realidad industrial.
- Es escasa la colaboración entre la industria y el mundo académico, que en ocasiones se muestra demasiado dogmático.
- Se considera que la aplicación de métodos formales encarece los productos y ralentiza su desarrollo.

Parece claro que los métodos formales se implantarán en la industria a través de nuevos profesionales con sólidos conocimientos de las técnicas matemáticas [10].

Luego el cambio deberá venir desde el mundo académico que tendrá que hacerse eco tan pronto como sea posible de los avances de las técnicas formales en la industria. Ésta ya hace algún tiempo que con esperanza se interesa por los métodos formales como herramienta para asegurar la calidad de productos cuyo desarrollo se complica cada día.

Además ya existen productos y estándares ampliamente aceptados con un componente matemático y formal importante en su diseño [5].

4. Mitos y mandamientos

Los mitos y mandamientos sobre los métodos formales constituyen dos aproximaciones clásicas a las particularidades de tales métodos.

4.1. Los catorce mitos sobre los métodos formales.

Podríamos decir que a los métodos formales "la fama les precede". Los conocidos 14 mitos sobre los métodos formales (7 mitos [11] y 7 más [3]) recogen ideas preconcebidas sobre ellos:

1. Los métodos formales garantizan la perfección del software y hacen innecesaria su verificación.
2. Los métodos formales sólo sirven para demostrar que los programas son correctos.
3. Los métodos formales sólo es necesario aplicarlos en sistemas donde la seguridad es crítica.
4. Los métodos formales sólo pueden ser aplicados por expertos en matemáticas.
5. La aplicación de métodos formales aumenta los costes de desarrollo.
6. Los métodos formales no son bien vistos por los usuarios.
7. Los métodos formales no se aplican en sistemas reales de gran tamaño.
8. Los métodos formales retrasan el desarrollo.
9. No existen herramientas informáticas de apoyo a la aplicación de los métodos formales.
10. Los métodos formales implican la renuncia a los métodos tradicionales de diseño.
11. Los métodos formales sólo sirven para diseño de software, no para diseño de hardware.
12. Los métodos formales no son necesarios.
13. Los métodos formales no son sustentados por la industria y los investigadores.
14. Los defensores de los métodos formales sólo utilizan métodos formales.

Estos mitos ofrecen una imagen verdaderamente negativa de los métodos formales pero es previsible que sean aceptados como ciertos por los alumnos que se acercan por primera vez a la aplicación de métodos formales a tareas propias de informáticos.

En la Tabla 1 recogemos los resultados de una pequeña encuesta realizada para conocer el calado que estos mitos tienen entre los alumnos.

Participaron en el sondeo 45 alumnos de Metodología de la Programación que calificaron como cierto o falso cada uno de los enunciados que aparecen listados más arriba.

Para algunos de los mitos el porcentaje de alumnos que lo consideran cierto no es muy elevado. Hay que tener en cuenta que en el momento de realizar la encuesta los métodos formales ya habían sido explicados y aplicados en clase por lo que es muy posible que ya se hubiera producido una cierta desmitificación.

Mito nº	% de alumnos que lo consideran *cierto*	% de alumnos que lo consideran *falso*
1	56%	44%
2	58%	42%
3	4%	96%
4	4%	96%
5	44%	56%
6	60%	40%
7	40%	60%
8	60%	40%
9	58%	42%
10	29%	71%
11	87%	13%
12	33%	67%
13	31%	69%
14	7%	93%

Tabla 1. Mitos de los métodos formales entre los estudiantes

4.2. Los diez mandamientos

Los diez mandamientos [4] sobre los métodos formales sugieren

1. Elegir una notación apropiada.
2. No formalizar en exceso.
3. Hacer una buena estimación de costes.
4. Tener disponible un conocedor de los métodos.
5. No abandonar los métodos tradicionales.
6. Documentar suficientemente.
7. Alcanzar los estándares de calidad.
8. No ser dogmático.
9. Repasar una y otra vez.
10. Reutilizar.

Pueden ser considerados como una serie de recomendaciones a tener en cuenta durante el diseño y la docencia de asignaturas que incluyen la aplicación de métodos formales como algo central.

Tener presentes estos consejos es también conveniente para intentar dar solución a los problemas de implantación de los métodos formales en la industria.

5. Propuesta de mejora

Hacemos nuestra propuesta con el fin de procurar la mayor motivación de los estudiantes de la asignatura Metodología de la Programación.

Pensamos que los mitos sobre los métodos formales [3][11] dan de ellos una imagen negativa que no se justifica y sospechamos que estos mitos son considerados realidades por nuestros estudiantes. Proponemos una serie de actividades orientadas a desmitificar los métodos formales bajo la hipótesis de que el lograrlo puede hacer que el alumno se sienta más motivado.

Tras haber estudiado algunos de los condicionantes del proceso de enseñanza-aprendizaje de los métodos formales y a la luz de los diez mandamientos relativos a su aplicación [4], proponemos para el equipo docente de la asignatura las siguientes acciones:

1. Organizar una actividad en grupo a principio de curso en la que los alumnos debatan los mitos de los métodos formales.
2. Analizar los temarios de las asignaturas que en el futuro los estudiantes cursarán para tener ocasión de establecer vínculos concretos con ellas.
3. Localizar, en cursos anteriores, los momentos concretos en los que los alumnos aprenden conceptos básicos empleados en la aplicación de métodos formales, considerando la posibilidad de proporcionar a los profesores de esas asignaturas ejemplos o comentarios que hagan referencia a la nuestra.
4. Documentar para los estudiantes la afirmación de que la informática, como ciencia, tiene sus bases en las matemáticas.
5. Informar a los alumnos sobre aplicaciones de los métodos formales en la industria.
6. Recurrir durante las explicaciones a programas ya desarrollados por los alumnos aplicando los nuevos métodos sólo en algunos fragmentos del código, de forma que el resto sea considerado de utilidad.
7. Presentar algoritmos de cierta complejidad desarrollados formalmente, aunque no se espere de los alumnos que sean capaces de obtener soluciones parecidas. Permitirá observar que los métodos no sólo se aplican a problemas triviales.

8. Transcribir algunos programas obtenidos a un lenguaje de programación conocido por los alumnos, para bajar el nivel de abstracción.
9. Repetir la encuesta sobre el calado de los mitos entre los estudiantes, a principio y a final de curso. Podremos así observar la evolución de la visión que los alumnos tienen de los métodos formales.

En esta asignatura ya se ha probado con éxito la realización de actividades en grupo, que son recibidas con notable entusiasmo por los estudiantes.

Se han hecho también referencias a programas desarrollados en C o Pascal y se ha podido observar que tratar con los lenguajes que conocen da confianza a los estudiantes.

En el diseño de actividades y presentación de documentos complementarios en clase hay que tener en cuenta que normalmente los alumnos consideran que saben programar y que no dominan las matemáticas. Es positivo aprovechar su confianza en sus habilidades como programadores evitando presentar los métodos formales como algo que las hace prescindibles y ayudarles a superar su complejos en lo que se refiere a las matemáticas.

6. Conclusión

Somos conscientes de la dificultad de la tarea de incidir en la motivación de una persona, sobre todo si prescindimos de los conceptos de recompensa, castigo e incentivo propios de la motivación extrínseca más fácilmente modulable desde el exterior del individuo. Sin embargo la motivación es el desencadenante del esfuerzo necesario para el aprendizaje y debe ser por lo tanto tenida en cuenta.

Referencias

[1] Almstrum,V.L., Dean,C.N., Goelman,D., Hilburn,T.B., Smith,J. *Support for teaching formal methods*, ACM SIGCSE Bulletin,Volume 33 , Issue 2 ,June 2001.

[2] Baldwin,D., Henderson P.B., *The importance of mathematics to the software practitioner,* IEEE Software, 19(2): 110-112,March-April,2002.

[3] Bowen,J.P., Hinchey,M.G., *Seven more myths of formal methods*, IEEE Software, 12(4):34-41, July 1995.

[4] Bowen,J.P., Hinchey,M.G., *"Ten Commandments of Formal Methods"*, IEEE Computer, 28(4):56-63, April 1995.

[5] Bruce, K.B., Drysdale R.L.S., Kelemen C., *Why Math?* En preparación para Communications of the ACM, 2002. http://brastias.cs.geneseo.edu/~baldwin/math-thinking/CACM4.pdf

[6] *Computing Curricula 2001, Computer Science*. The Join Task Force on Computing Curricula, IEEE Computer Society, ACM.

[7] Craigen,D., Gerhart,S., Ralston,T., *An International Survey of Industrial Applications of Formal Methods*, Technical Report, National Institute of Standards and Technology, Nistgcr 93/626. 1993.

[8] Craigen,D., Gerhart,S., Ralston,T., *Formal Methods Reality Check: Industrial Usage* IEEE Transaactions on Sofware Engineering, 21 (2): 90-98, 1995.

[9] Crocker,D., *Making formal methods popular thorugh automated verification.* International Joint Conference on Automated Reasoning, Siena, Italy, June 2001.

[10] Gibson,P., Méry,D., "*Teaching Formal Methods: Lessons to Learn"* BCS electronic workshops in computing, ISSN: 1477-9358, Butterfield and Flynn, 1998.

[11] Hall,A., *Seven myths of formal methods*. IEEE Software, **7**(5):11-19, September 1990.

[12] Henderson, P., *Mathematical Reasoning in Software Engineering Education*, en preparación para Communications of the ACM. http://brastias.cs.geneseo.edu/~baldwin/math-thinking/MathInSE.PDF

[13] Holloway,C.M., Butler,R.W., *Impediments to industrial use of formal methods,* IEEE Computer: 25-26, April 1996.

[14] Jacob,I., *Metodología y herramientas para el aprendizaje de la verificación y derivación formal de programas*, Jenui 2000 (Libro de comunicaciones de las VI Jornadas sobre la Enseñanza Universitaria de la Informática) ISBN: 84-138-409-7.

[15] Martínez,J.R., Galán,F. *Estrategias de aprendizaje, motivación y rendimiento académico en alumnos universitarios*. Revista

Española de Orientación y Psicopedagogía. 11, (19), 35 – 50. 2000.

[16] Mateo,M. *La motivación, pilar básico de todo tipo de esfuerzo*. Proyecto social: revista de relaciones laborales. - N.9 (nov. 2001), p. 163-184.

[17] Page,R., *Formal Methods Education and Programming Effectiveness,* Beseme Project, http://www.cs.ou.edu/~beseme/ (2000-2003)

[18] Wittrock,M.C*., Procesos del pensamiento de los alumnos*. En autor (ed). La investigación de la enseñanza III: Profesores y alumnos. 544 - 585. Barcelona: Paidós.1986.

Verificación de Autómatas y Gramáticas

Jairo Rocha
Dept. de Matemàtiques i Informàtica
Universitat de les Illes Balears
jairo@uib.es

Resumen

La verificación de autómatas de estados finitos y de pila, y de gramáticas libres de contexto puede ser materia de enseñanza en una asignatura de Lenguajes Formales usando las metodologías descritas en este artículo. Se intenta mostrar que para los autómatas y gramáticas la verificación es posible en la práctica diaria. Las metodologías se basan en la asociación de *significados* (lenguajes) a los símbolos que aparecen en la solución de un problema (estado, contenido de la pila o variable de una gramática). Los significados corresponden a ciertas propiedades que son invariantes a través de las configuraciones. Los estudiantes de segundo curso de Ingeniería Informática han sido capaces de aplicar las metodologías y aumentar sus elementos de juicio a fin de asociar los significados de los símbolos que aparecen en la solución de un problema.

1. Introducción

Los costes de producción de software son tan altos, en parte, por la falta de corrección de los sistemas que se diseñan e implementan. Los errores son un resultado natural de la alta complejidad de estos sistemas desarrollados y, para intentar reducirlos, se hace necesario especificar rigurosamente los subsistemas y sus relaciones.

Este artículo trata de metodologías que permiten asegurar ls corrección de soluciones a problemas de especificación usando autómatas y lenguajes formales que son fácilmente asimilables por estudiantes de segundo año.

La mayor parte de las soluciones a problemas propuestos en una asignatura de Lenguajes Formales son el resultado de modificaciones que se han producido durante el intento de demostración de soluciones erróneas. Es decir, la solución correcta no se encontró hasta que la demostración exigió completar todos los casos y cubrir aquella combinación de situaciones que no se pensó durante la descripción de la solución.

Por el contrario, en la práctica diaria de un programador no se verifican los programas, por lo que los casos en los que las soluciones propuestas no funcionan aparecen durante las pruebas y experimentos posteriores. Esta situación es comprensible dada la imposibilidad teórica de verificar los programas de manera automática, y la imposibilidad práctica de escribir demostraciones para programas de longitud normal.

El objetivo primordial de este artículo es argumentar que se deben aumentar los elementos de juicio y afinar la intuición de los programadores con el fin de que se produzcan programas más correctos. El programador deberá descubrir en las metodologías de verificación que cada símbolo que aparece en la solución de un problema tiene un significado. Y que los significados se relacionan lógicamente de una manera muy precisa para satisfacer las especificaciones o definiciones de las soluciones.

El razonamiento que se propone para la verificación de autómatas se basa en la identificación de ciertas propiedades que en una configuración (estado actual, contenido de la pila, lenguaje generado por una variable de una gramática, etc.) del autómata son invariantes; es decir, que se puede mostrar que no cambian (siempre son ciertas) cuando el autómata llega a la misma configuración, aunque haya pasado por diferentes pasos de cómputo, tales como cambio de estados, contenido de pilas, de cintas o de variables. Por ejemplo, el estudiante debe saber

que todas las palabras que en un autómata de estados finitos determinista llegan a un estado fijo satisfacen una propiedad que no satisfacen palabras que llegan a otro estado; y que hay propiedades que relacionan una configuración de la pila con la palabra leída, o una variable de una gramática con sus palabras generadas.

Nuestro objetivo quedará cumplido si el uso de las técnicas descritas trae como resultado el hábito duradero (y casi inconsciente) de escribir soluciones, y programas en particular, respetando estrictamente las relaciones entre los significados intuitivos de los símbolos usados, sean estos estados, nombres de variables en una gramática, vectores, nombres de variables y posiciones dentro de un programa, nombres de procedimientos, de clases, etc.

Nuestra experiencia docente, y en particular con la asignatura de Autómatas y Lenguajes Formales, nos ha convencido de que la gran mayoría de los detalles técnicos de los métodos, definiciones o teoremas se olvida con facilidad y prontitud. Por lo tanto, si los estudiantes se concentran en estos detalles, ellos y los docentes verán sus esfuerzos difuminarse rápidamente.

El énfasis de la asignatura debería estar en la justificación. Se deben dar herramientas para responder a la pregunta de por qué cierto autómata reconoce un lenguaje previamente definido formalmente, con la intención de que el estudiante cree el hábito de preguntarse sobre la fiabilidad absoluta de las transiciones, reglas o instrucciones que decide incluir en una solución. En resumen, el contenido de la asignatura de Lenguajes Formales debería ayudar a mejorar al alumno sus habilidades de justificación en el área de la computación.

Esta actitud se contrapone al aparente énfasis que algunos estudiantes dan a las recetas relacionadas con estos temas; es fácil que un estudiante simplifique el temario de un curso de teoría de lenguajes formales y autómatas a recetas para, por ejemplo, encontrar un autómata de estados finitos determinista equivalente a uno no determinista, minimizar autómatas de estados finitos, encontrar la forma normal de Chomsky de una gramática, o escribir máquinas de Turing que solucionen problemas concretos simples o no. La importancia de evitar que una asignatura cualquiera se resuma al dominio de recetas no sobra que sea enfatizada. El estudiante fácilmente olvida las recetas y, por lo tanto, el resultado final de producir un cambio de actitud en el estudiante y enseñar conceptos importantes y duraderos no se alcanza.

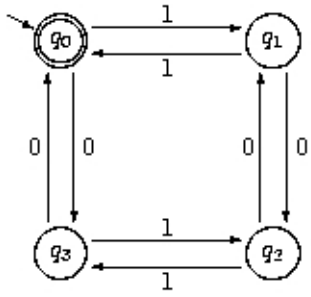
Así, el estudiante debería conocer los métodos completamente automáticos que permiten cambiar una solución de un simbolismo a otro, pero el objetivo no es que el estudiante aprenda esos métodos a la perfección, sino que sepa de su existencia y utilidad para generar soluciones fiables. En particular, el estudiante no se debería evaluar sobre el desarrollo de una máquina de Turing para un lenguaje dado, cuando en la práctica nadie usa máquinas de Turing y usando otros formalismos obtendría soluciones más fáciles; las máquinas de Turing se enseñan con el objetivo de definir formalmente un programa de manera simple y estudiar los límites últimos de las máquinas universales, más comúnmente llamadas ordenadores. Igualmente, el estudiante se debería evaluar sobre la verificación de que un autómata de pila es correcto respecto a un lenguaje dado, y no pidíendole que escriba el autómata asociado a una gramática, problema que puede ser solucionado mejor por un ordenador.

En este artículo se describen temas que no se tratan, o sólo de manera tangencial, en los textos usuales (por ejemplo, [1, 2]): verificación de autómatas finitos, de gramáticas y de pila. La verificación de máquinas de Turing no se cubre dado que existen textos que incluyen metodologías de demostración de lenguajes de programación generales (por ejemplo, las *wp* en [4]).

Sugerimos dar prioridad a la verificación sobre temas más tradicionales como minimización de autómatas, los lemas de bombeo y las formas normales de gramáticas porque la verificación ayuda más a crear el hábito de justificar que lo que ayuda el conocimiento de aquellos temas más tradicionales.

2. Autómatas de Estados Finitos

Explicaremos la metodología para cada tipo de verificación con un ejemplo. En este caso,



supongamos que el problema es verificar que el siguiente autómata A reconoce el lenguaje

$L=\{w \in (0+1)^* \mid w$ tiene par de 0's y de 1's $\}$.

2.1. Ecuaciones

El primer paso es definir las ecuaciones asociadas al autómata en términos de los lenguajes asociados a cada estado:

$$l_i = \{w \mid (q_0,w) \Rightarrow^* (q_i,w)\}$$

Es decir, l_i es el conjunto de palabras que en cero o más pasos del autómata llegan a q_i partiendo del estado inicial. Las ecuaciones son:

$$l_o = l_1 1 + l_3 0 + \lambda$$
$$l_1 = l_0 1 + l_2 0$$
$$l_2 = l_1 0 + l_3 1$$
$$l_3 = l_0 0 + l_2 1$$
$$L(A) = l_0$$

en las que el lenguaje del autómata es el asociado a q_0 porque es el único estado final. Notese que las ecuaciones por sí solas describen totalmente el autómata.

2.2. Significados

El siguiente paso es el más delicado de la metodología. Consiste en definir un lenguaje asociado a cada estado en términos similares a los del lenguaje L. En este caso, se hace necesario adivinar el conjunto de palabras asociadas a cada estado en términos de la paridad de los símbolos, independientemente del autómata. Este paso es el menos mecánico de todos y se realiza normalmente por prueba y error. El lector debe descubrir los siguientes lenguajes:

$L_0 = \{w \mid w$ tiene par de 0´s y 1´s$\}$
$L_1 = \{w \mid w$ tiene par de 0´s e impar de 1´s$\}$
$L_2 = \{w \mid w$ tiene impar de 0´s y 1´s$\}$
$L_3 = \{w \mid w$ tiene impar de 0´s y par de 1´s$\}$.

2.3. Partición

Si el autómata es determinista se hace necesario mostrar que los lenguajes del paso anterior forman una partición de Σ^*. En la mayoría de los autómatas, este paso se hace al mismo tiempo que se ensayan varias definiciones de significados pues es una manera de saber que estas definiciones van por buen camino. En este ejemplo, es obvio que estos lenguajes forman una partición.

2.4. Una Solución de las Ecuaciones

Si en las ecuaciones de arriba se reemplaza l_i por L_i, se debe comprobar que las ecuaciones siguen siendo ciertas. Si este es el caso, entonces por unicidad de la solución del sistema, debe ser cierto que $l_i = L_i$. La unicidad es una consecuencia de la aplicación iterativa del Lema de Arden [1].

Si el autómata es determinista es suficiente mostrar que las ecuaciones valen en la dirección de derecha a izquierda, pues las otras direcciones valen automáticamente si los significados forman una partición [3]. Si el autómata no es determinista se hace necesario mostrar cada ecuación en ambas direcciones.

En este ejemplo, hay que demostrar cada una de las siguientes afirmaciones:

$L_o \supseteq L_1 1 + L_3 0 + \lambda$
$L_1 \supseteq L_0 1 + L_2 0$
$L_2 \supseteq L_1 0 + L_3 1$
$L_3 \supseteq L_0 0 + L_2 1$

Todas estas comprobaciones hacen la verificación larga, pero en la mayoría de las veces, cada comprobación es simple, como en este caso.

2.5. El Paso Final

Debido a que $l_i = L_i$, para cada estado q_i, y, suponiendo que los estados finales son $q_{f1},..., q_{fk}$,

$$L(A) = l_{f1} \cup ... \cup l_{fk} = L_{f1} \cup ... \cup L_{fk},$$

por lo que la verificación termina mostrando que

$$L_{f1} \cup ... \cup L_{fk} = L.$$

En este ejemplo basta mostrar que $L_0 = L$, lo que es obvio.

3. Gramáticas Libres de Contexto

Nos interesa demostrar que el lenguaje generado por la gramática G

$I \rightarrow \lambda \mid aB \mid bA$
$A \rightarrow aI \mid bAA$
$B \rightarrow bI \mid aBB$

es
$L = \{ w \in (a+b)^* \mid w$ tiene el mismo número de a'es que de b'es$\}$.

La metodología es muy similar a la de la sección anterior, y como antes, lo que la hace simple es un teorema de unicidad de las soluciones de un sistema de ecuaciones.

3.1. Ecuaciones

El primer paso es definir las ecuaciones asociadas a la gramática:

$X_I = \lambda + aX_B + bX_A$
$X_A = aX_I + bX_AX_A$
$X_B = bX_I + aX_BX_B$

en las que X_I, X_A y X_B son variables de lenguajes. Los lenguages generados por las variables,

$Gen(I) = \{w \mid I \Rightarrow^* w\}$
$Gen(A) = \{w \mid A \Rightarrow^* w\}$
$Gen(B) = \{w \mid B \Rightarrow^* w\}$

son solución de las ecuaciones. Las condiciones de unicidad de la solución son ciertas para estas producciones porque las partes derechas de las producciones son constantes o tienen un símbolo terminal[3].

3.2. Significados

El siguiente paso es nuevamente el más difícil de la metodología. Consiste en definir un lenguaje asociado a cada variable de la gramática en términos similares a los del lenguaje L.

Después de varias derivaciones, debe estar claro que lo más es seguro es que los lenguajes generados por las variables sean

$L_I = \{ w \mid |w|_a = |w|_b \}$
$L_A = \{ w \mid |w|_a = |w|_b + 1 \}$
$L_B = \{ w \mid |w|_b = |w|_a + 1 \}$.

donde $|w|_\sigma$ representa el número de veces que σ aparece en w.

Nos interesa ver que L_I, L_A y L_B son también solución de las ecuaciones, por lo que, por unicidad $L_I = Gen(I)$, $L_A = Gen(A)$ y $L_B = Gen(B)$.

3.3. Una Solución de las Ecuaciones

Basta demostrar que (L_I, L_A, L_B) satisfacen cada ecuación. La demostración de que cada ecuación se satisface necesita dos direcciones (no hay atajos como en la sección anterior). La demostración de cada uno de los casos en cada dirección normalmente es simple, aunque todo junto resulta ser largo.

En este ejemplo concreto, la demostración de

$L_A \subset aL_I + bL_AL_A$

requiere demostrar que si una palabra de L_A comienza por *b*, el resto de la palabra se puede dividir en dos subpalabras, cada una con una *a* más; para esto se hace necesario mostrar cómo dividir efectivamente la palabra, contando de izquierda a derecha la diferencia entre el número de *a*'es y el de b'es.

3.4. El Paso Final

Debido a que $L_X = Gen(X)$ para cada variable X de la gramática, $L(G) = Gen(S)$, donde S es la variable inicial. Entonces, la verificación termina mostrando que

$$L_S = L.$$

En este ejemplo, basta mostrar que $L_I = L$, lo que es inmediato.

4. Autómatas con Pila

La verificación de autómatas de pila deterministas se realiza por inducción en el tamaño de la palabra procesada. Esta estrategia implica una complicación simbólica que se había evitado en las dos metodologías anteriores usando teoremas de unicidad sobre un sistema de ecuaciones. Pero para este caso no hay un teorema de unicidad que conozcamos, aunque sí hay lenguajes asociados a configuraciones (estado,pila) del autómata.

4.1. Lenguaje Asociado a una Configuración de la Pila

Como en las metodologías anteriores, la definición de ciertos lenguajes intermedios asociados a significados de símbolos (estados, variables, configuraciones, etc.) es decisiva para conseguir el objetivo.

Consideremos el siguiente ejemplo. Sea

$$L = \{ w \mid w \in (a+b)^* \text{ y } |w|_a = |w|_b \}.$$

El autómata de pila M que reconoce $L\#$ por pila vacía es:

$$\begin{array}{ll} \delta(q,a,Z_0) = (q, aZ_0) & \delta(q,b,Z_0) = (q, bZ_0) \\ \delta(q,a,a) = (q, aa) & \delta(q,b,b) = (q, bb) \\ \delta(q,a,b) = (q,\lambda) & \delta(q,b,a) = (q, \lambda) \\ \delta(q,\#,Z_0) = (q, \lambda) & \end{array}$$

Los lenguajes intermedios (o significados) corresponden a conjuntos de palabras de entrada con las cuales se llega a un estado y a un contenido de la pila dados. Dado un autómata de pila $(Q,\Sigma,\Gamma,\delta, q_0,Z_0,F)$, se define el siguiente lenguaje para cada $q \in Q$ y $\gamma \in \Gamma$:

$$L(q,\gamma) = \{w \in \Sigma^* \mid (q_0, w, Z_0) \Rightarrow^* (q_0, \lambda, \gamma) \}.$$

En nuestro ejemplo, $L(M) = L(q, \lambda)$.

Demostraremos las siguientes igualdades:

$$L(q,a^kZ_0) = \{w \mid |w|_a = |w|_b + k \},$$
$$L(q,b^kZ_0) = \{w \mid |w|_b = |w|_a + k \}.$$

La pregunta en este instante debería ser: ¿de dónde se han obtenido estas ecuaciones? La respuesta es: corresponden a la formalización de las ideas intuitivas de los significados de los estados y del contenido de la pila que se dieron antes de definir el autómata. Por ejemplo, la primera ecuación dice que para llegar al estado q con k *a*es encima de Z_0, se debe haber leído una palabra que tiene k *a*es más que *b*es.

4.2. Inducción

La demostración de los significados de los lenguajes se hace por inducción simultáneamente sobre las dos igualdades. Se deben hacer simultáneamente porque se puede pasar de la configuración de una ecuación a la otra leyendo un símbolo, es decir, las ecuaciones se relacionan entre sí, como es lo normal en autómatas y gramáticas.

Las ecuaciones corresponden a demostrar por inducción en $|w|$ que para todo k:

$$(q,w,Z_0) \Rightarrow^* (q, \lambda, a^k Z_0) \Leftrightarrow |w|_a = |w|_b + k \quad \text{y}$$
$$(q,w,Z_0) \Rightarrow^* (q, \lambda, b^k Z_0) \Leftrightarrow |w|_b = |w|_a + k.$$

Cuando se considera $w\sigma$ en el paso de inducción, se calculan todas las posibles configuraciones (q, σ, γ) antes de que la entrada quede vacía. Los diferentes casos para q, σ y dependen de las transiciones del autómata y pueden ser varios, lo que hace la demostración larga. Cada caso requiere usar la hipótesis de inducción sobre la variable k apropiada, lo que no es difícil. Sin embargo, los estudiantes deben hacer varias demostraciones antes de aprender a hacer un manejo simbólico correcto.

4.3. El Paso Final

Para terminar veamos que $L(M) = L$. Sabiendo que $L(M) = L(q, \lambda)$, y observando que la única forma de vaciar la pila es eliminando el símbolo Z_0 al leer #, se tiene que

$(q, w\#, Z_0) \Rightarrow^* (q, \#, Z_0) \Rightarrow (q, \lambda, \lambda)$
ssi $(q, w, Z_0) \Rightarrow^* (q, \lambda, Z_0)$
ssi (según lo demostrado para $a^0 Z_0$)
$|w|_a = |w|_b + 0$
ssi $w \in L$,

lo que muestra que el autómata es correcto.

4.4. Experiencia Docente

Los estudiantes de segundo año aprenden que un ejercicio que requiera dar un autómata o gramática no estará completo hasta que no se dé la verificación correspondiente. Es obvio que la mayoría de los estudiantes preferiría dar soluciones a los ejercicios sin dar las verificaciones como se hace en los libros de lenguajes formales tradicionales y en la mayoría de las asignaturas de programación. Esto hace aún más importante que en al menos esta asignatura se mantenga como objetivo la necesidad de justificar cada parte de una solución presentada. Sin embargo, no disponemos de ningún elemento objetivo que permita medir si los estudiantes saben dar, a largo plazo, mejores justificaciones.

Las metodologías son perfectamente accequibles a los estudiantes. Es decir, más de un 50% de los estudiantes son capaces de dar verificaciones correctas en los exámenes. Solamente la verificación de automatas de pila requiere varias semanas hasta que los estudiantes reemplazan los símbolos correctamente en los pasos de las demostraciones por inducción de cada uno de los casos. Esta estrategia es sólamente un poco más compleja que los casos que se consideran en el uso de los lemas de bombeo, sólo que los resultados son más prácticos: la garantía de la corrección de las soluciones.

Referencias

[1] Rafel Casas, Lluís Màrquez. *Llenguatges, Gramàtiques i Autòmats: Curs Bàsic.* Universitat Politècnica de Catalunya, 1997.

[2] Dean Kelley. Teoría de Autómatas y Lenguajes Formales. Prentice Hall, 1995.

[3] Francesc Roselló, Jairo Rocha. *Autòmats i Llenguatges: Verificació, Implementació i Concurrència.* Materials Didàctics. Universitat de les Illes Balears, 2003 (En imprenta).

[4] Robert Sebesta.. *Concepts of Programming Languages*, Addison Wesley, 2002.

Tema estratégico

Educar/formar profesionales informáticos

Situaciones y Problemas: cómo preparar a los alumnos para lo que se les avecina

José Miró
Dept. de Matemàtiques i Informàtica
Universitat de les Illes Balears
07071 Palma de Mallorca
e-mail:
joe@ipc4.uib.es

Resumen

Preparamos a nuestros alumnos usando problemas claros, concisos, que se resuelven con lo que se ha enseñado en la asignatura, mientras que en su vida laboral lo que se van a encontrar son *situaciones* que son ambiguas, confusas y que no se sabe con qué conocimientos y técnicas deben resolverse. Para preparar mejor a nuestros alumnos para lo que se van a encontrar en su vida laboral, es necesario enseñarles a enfrentarse con situaciones, por dificultoso que sea hacerlo.

Never tell people 'how' to do things. Tell them 'what' to do and they will surprise you with their ingenuity. --General George S. Patton

1. Introducción

La misión docente de la universidad es la de preparar a nuestros alumnos para enfrentarse con los retos que se encontrarán durante su vida laboral [21,23]. ¿Realmente les preparamos para ello, o simplemente les impartimos conocimientos y esperamos que sepan valerse? Esta es una pregunta muy pertinente y que todo profesor debe formularse repetidamente.

El método habitual de impartir nuestras asignaturas consiste en explicar unos conocimientos teóricos y después practicar y profundizar estos conocimientos a través de problemas, sea en la misma asignatura o en otra relacionada. El uso de los problemas es esencial para anclar la sabiduría obtenida [19] y un buen profesor prepara siempre una buena colección de problemas bien definidos, claros y concisos, con los que los alumnos se vean forzados a reflexionar y practicar la materia que se acaba de impartir. El profesor concienzudo ayuda a los alumnos que tienen dificultades, encaminándoles en la buena dirección para que ellos puedan llegar a la solución, y, finalmente, resuelve o entrega soluciones de los problemas propuestos ya que el poder saber rápidamente lo que se ha hecho bien y mal refuerza el conocimiento.

En suma, el método expuesto supedita los problemas a la teoría. Es muy útil y efectivo para adquirir conocimientos y por eso el buen profesor lo usa extensamente. Pero el uso exclusivo de este tipo de problemas no prepara directamente a los alumnos para la vida profesional.

En su vida profesional nuestros alumnos no se van a encontrar problemas bien definidos, claros y concisos que se resuelven con 'lo que se acaba de explicar', sino con algo distinto, que llamaré *situaciones*. Las situaciones son ambiguas e imprecisas, a menudo explicadas verbalmente en lenguaje no técnico, con incongruencias y contradicciones. Para desenvolverse ante estas situaciones no está claro qué conocimientos se necesitan, ni tan siquiera si existen. El profesional, quizá con un grupo, será el único que trabaje en el problema y no podrá comparar con otros su trabajo para saber si va por buen camino. Tampoco se van a encontrar con un mentor que les guíe hacia la solución o que les pueda asegurar que lo que han hecho es correcto.

Un buen profesional de la informática va a ser valorado por lo bien que sepa reaccionar ante estas situaciones. Es cierto que también ha de saber resolver problemas más sencillos y de solución directa, pero esto raramente es

gratificante ni proporciona reconocimiento profesional. Incluso puede argumentarse que la resolución de estos problemas repetitivos es labor de programadores u otros técnicos y que lo que verdaderamente distingue al profesional informático es su capacidad de valerse ante situaciones.

La resolución de problemas es imprescindible para adquirir buenos y profundos conocimientos, pero si queremos preparar a nuestros alumnos para la vida profesional, debe complementarse explicándoles y haciéndoles practicar cómo desenvolverse ante situaciones. Pero enseñar esto es difícil. Inherentemente la situación sobrepasa el concepto de materia, o área de conocimiento, siendo además el aprendizaje de una habilidad, y por tanto un aprendizaje experiencial. Esto crea dificultades de todo tipo, desde políticas a técnicas. Probablemente por eso se han ideado soluciones híbridas en donde se usan problemas con algunas características de las situaciones. Esto es evidentemente positivo, pero no llega a preparar a los alumnos tan bien como lo haría el uso de situaciones en el aula.

Desde el curso 2001-2002 imparto una asignatura donde se explica un método de resolución de situaciones. La experiencia ha sido positiva, ha demostrado que se puede hacer, ha mostrado que para hacerlo bien aún tengo mucho que aprender, pero sobre todo ha hecho evidente que asignaturas con este tipo de contenido son necesarias. Paso a exponer lo que he hecho y lo que he aprendido en esta experiencia.

2. Problemas y situaciones

2.1 Problemas

Siendo ayudante de una asignatura de circuitos digitales, uno de los primeros problemas que diseñé tenía un texto larguísimo que ocupaba una página entera. El profesor encargado de la asignatura me sugirió que lo hiciera más corto y conciso. A mí ese problema me gustaba mucho, ya que era 'realista' y lo encontraba muy interesante. El profesor aceptó el problema (estoy seguro que para darme una lección) y apareció en la lista. Los alumnos lo odiaron: se perdían en el texto sin saber qué es lo que se les pedía ni qué se suponía que debían hacer.

Aprendí que para que los alumnos se centren rápidamente en la tarea a realizar y no gasten esfuerzos inútilmente en cuestiones derivadas de la redacción, los problemas debían tener unas ciertas características [9,13,19,20]. Un buen problema debe ser claro, conciso, directo, sin ambigüedades, con todos los datos necesarios, y ha de ser tal que se deba resolver utilizando los conceptos y técnicas en las que se desea que profundicen.

Ahora bien, como suele ocurrir en ingeniería, nada sirve para todo, y una característica adecuada a un objetivo es inadecuada para otro. Los problemas redactados con las características que acabo de exponer son muy buenos para aplicar y profundizar en los conocimientos adquiridos, pero promueve un procedimiento de trabajo que no prepara a los alumnos para la situaciones que se encontrarán en su vida laboral.

Los alumnos son inteligentes y se adaptan al tipo de tarea que se les propone para obtener el máximo rendimiento con el mínimo esfuerzo. Por ello han adoptado un método muy eficiente para resolver los problemas que se les plantean. Naturalmente, este método tiene infinitas variaciones, pero consiste en una mezcla de las siguientes acciones:

- Identificar los datos que se proveen y se piden.
- Buscar en el último tema explicado conceptos y fórmulas que relacionan estos datos.
- Buscar problemas similares que estén resueltos e intentar adaptarlos al problema propuesto.
- A falta de método mejor, juguetear con los datos y las fórmulas hasta llegar al valor que se pide.
- Si se llega al resultado sin utilizar alguno de los datos, sospechar que algo se ha escapado.
- Evaluar la corrección de las soluciones obtenidas comparándolas con las de otros compañeros.
- Comparar la solución obtenida con *la* solución del profesor como prueba final de la corrección del ejercicio.

Este método da resultado en el aula, aunque su utilidad profesional es escasa. Además promueve dos vicios:

- No desarrollar formas de comprobación de la corrección de los resultados obtenidos.

- Creer que existe *la* solución [2] y que el profesor es su guardián sagrado.

Estas carencias han sido identificadas en la literatura [7] e incluso se menciona en el currículum ACM/IEEE 91. Siendo así, sorprende, aunque es muy ilustrativo, que a pesar de todo, a los problemas se les denomina con el título favorable de *well-defined problems* (problemas bien definidos) mientras que las situaciones, a las cuales se les reconoce su importancia, reciben el término algo peyorativo de *ill-defined problems* (problemas mal definidos). Quizá esta imagen desfavorable de las situaciones sea el motivo por el que las estrategias que se usan para dotar a los problemas de algunas características deseables sean parciales y ataquen más a los síntomas que al meollo.

Una primera estrategia es la de usar problemas abiertos [18], es decir, problemas sin una respuesta y un final definidos. Este tipo de problemas obliga al alumno a ser curioso y creativo y en general no es posible resolverlos utilizando el método descrito anteriormente. Pero en el mundo laboral los problemas suelen ser bastante cerrados: se requiere una solución que se ajuste al objetivo de la empresa y se requiere para una fecha dada. Incluso se desconfía de los que consideran los problemas como un reto personal y se olvidan de las necesidades y hábitos de la empresa. Los problemas abiertos son los típicos en investigación, y quizá por eso gustan en la universidad a pesar de su utilidad limitada.

Una segunda vía es el uso de problemas de gran tamaño [7], en forma de proyectos o trabajos de final de curso. Los problemas de gran tamaño suelen exigir integrar el uso de la mayor parte de los conocimientos impartidos durante el curso, lo cual es muy positivo, y además pueden poseer alguna otra característica de las situaciones, aunque esto es más accidental que por diseño. Desgraciadamente, el uso de problemas de gran tamaño presenta dos inconvenientes. El primero es que están relacionados con una asignatura, y por lo tanto con unos conocimientos y técnicas concretas; el segundo, es que se pueden hacer pocos.

Una variante de los problemas de gran tamaño son los 'problemas del mundo real' [6,8,12]. Como su nombre indica, son problemas que se importan al mundo universitario típicamente desde empresas. La diferencia esencial entre este tipo de problemas y los otros problemas de gran tamaño es que la descripción del trabajo a hacer y la resolución de dudas la hace un 'cliente', que suele provenir de la empresa, y no el profesor. El cliente no sabe cuál es la situación del aula y presenta el problema y responde preguntas desde el punto de vista de la empresa, y no de la asignatura. Se consigue así un factor de realismo interesante. Naturalmente, los problemas del mundo real tienen los mismos inconvenientes, ya mencionados, de los problemas de gran tamaño, y desgraciadamente, en las experiencias que he visto descritas de experiencias de este tipo, los profesores, posiblemente con la intención de ayudar a los alumnos, le quitan buena parte del realismo a los problemas [19]. Por ejemplo, en un caso se usa como cliente a un ex-alumno [22]. Los autores se felicitaban de esta elección y de lo mucho que este cliente había podido ayudar a los alumnos gracias a los conocimientos que tenía de la asignatura, sin percatarse que habían convertido su problema del mundo real en algo bastante irreal.

También cabe abogar por el uso del proyecto final de carrera como lugar idóneo para introducir las situaciones [4]. Evidentemente, el proyecto final de carrera se puede asemejar mucho a una situación, y el alumno dispone aquí de tiempo y recursos para trabajar en este nuevo ambiente. Pero el proyecto final de carrera es inadecuado como lugar donde enseñar los métodos de resolución de situaciones por varios motivos. Por un lado es ineficiente, ya que habría que enseñar el método alumno a alumno; además el alumno sólo podría practicar ante las situaciones una vez, lo que es insuficiente; y sobre todo, tiene lugar demasiado tarde: para cuando el alumno llega al proyecto ya está fuertemente sesgado hacia el método de resolución de problemas descrito anteriormente. Como en todo, para dominar las situaciones hay que practicar su resolución a menudo y desde el principio. El proyecto final de carrera es ideal para ser una última práctica de resolución de situaciones, es inadecuado para ser la primera.

Estas y otras extensiones a los problemas [1,3,14] resuelven parcialmente algunas de las carencias que se han percibido en la preparación de nuestros alumnos. Pero, algunas carencias son inherentes al uso de problemas. El simple hecho

que los problemas deban resolverse dentro de una asignatura donde se imparte una materia obliga al alumno a utilizar una técnicas determinadas, mientras que en su vida laboral, al enfrentarse a una situación no sabrá a priori si debe abordarlo con técnicas de programación, algoritmia, sistemas operativos, o hardware. Un ejemplo servirá para ilustrar esta idea.

2.2 Situaciones

Veamos una situación típica.

La empresa Qjo S.A. ha reestructurado completamente su infraestructura informática. Ha cambiado todos los ordenadores por otros mucho más potentes, y ha decidido aprovechar la situación para cambiar el programa que más usan –un programa de planificación de rutas— por otro, utilizando un algoritmo más eficiente y aprovechando mejor los nuevos recursos. La empresa de un joven informático recién salido de la universidad ha sido la encargada de escribir el nuevo programa. A los pocos días de haberlo instalado nuestro joven informático recibe un mensaje de su jefe diciéndole que los de Qjo S.A. están muy descontentos con el programa y que fuera a ver qué pasa. No le sorprende demasiado —siempre es difícil acostumbrarse a un nuevo sistema— y está seguro que el problema es que no se han leído el manual. Hace una visita y se encuentra con la gran sorpresa que la queja principal es que el nuevo programa es muy lento.

Si este texto se entregara en una asignatura de programación, el alumno buscaría la solución en una mala estructura de datos; si en una asignatura de algoritmia, en un mal algoritmo; si en una asignatura de sistemas operativos, en una mala planificación de procesos o en un mal uso de la memoria virtual; si en una asignatura de arquitectura en un conflicto entre periféricos o un mal uso de la cache, si en una asignatura… Pero en el entorno laboral, ¿qué debe hacer nuestro joven informático? La situación no es ni clara ni directa, no se sabe qué técnicas utilizar, ni siquiera si las técnicas existen. Por no saber, no se sabe siquiera si hay un problema informático: podría ser que Qjo S.A. tuviera un problema de liquidez y se han inventado las quejas para retrasar el pago. Y desde luego, no puede ir al jefe y decir que no sabe qué pasa.

Podría parecer que ante situaciones que son infinítamente variadas no existe método alguno que pueda ayudar al profesional, y que todo depende de su genio e ingenio innatos. Pero estos métodos sí que existen y podemos y debemos preparar a nuestros alumnos para enfrentarse con lo que se encontrarán cuando quieran ejercer su profesión.

3. Enseñando a desenvolverse ante situaciones

Hemos visto que es importante que los alumnos sepan desenvolverse ante situaciones. Pero enseñarlo es difícil.

La primera dificultad es la de disponer de una asignatura donde hacerlo. Por su propia esencia, ha de ser una asignatura donde no haya ningún cuerpo de teoría a enseñar, y que no esté relacionada con ninguna otra asignatura concreta, y por lo tanto no debe estar relacionada con ningún departamento ni área de conocimiento. Evidentemente, es muy difícil introducir una asignatura así en un plan de estudios.

Aparte de esta primera dificultad política, existen otras dificultades técnicas. Dado que lo que se desea enseñar es un proceso y no conocimientos, es difícil establecer unos objetivos concretos y un método de evaluación. Esto es difícil para el profesor, ya que la evaluación pasa a tener una fuerte componente subjetiva, y también lo es para el alumno, acostumbrado a que se le evalúe en función de lo cerca que su resultado esté con 'el' resultado correcto. El alumno puede considerar que la evaluación es arbitraria y esto puede dar lugar a conflictos desagradables.

Otra dificultad es que, como se busca un cambio de mentalidad del alumno que es necesariamente muy lento, es difícil constatar si hay un avance. Esto es duro tanto para el profesor –que lo suple con fe— como para el alumno, que cree que está trabajando y sufriendo para nada. No es la primera vez que me enfrento con una asignatura de estas características [15,16], y, por suerte, he tenido la grata recompensa de ver que antiguos alumnos, que en el momento no veían la utilidad de lo que les explicaba y exigía, años

después han agradecido las enseñanzas recibidas. Por ello hay que tener fe en lo que haces, y continuar avanzando con la seguridad de la importancia de lo que se enseña.

3.1. Preparación del curso

En el curso 2001-02 tuve la suerte de que se me asignara un trimestre de la asignatura Métodos Matemáticos del segundo ciclo de la licenciatura de matemáticas. Con el apoyo y beneplácito del jefe de estudios utilicé esta asignatura para explicar a los alumnos el concepto de situación e introducirlos en los métodos de resolución de las mismas. Aunque esta asignatura es de la licenciatura de matemáticas, las ideas generales que expongo son perfectamente aplicables a una asignatura de la licenciatura de informática, si existiera. Tuve la suerte de contar con la ayuda del profesor emérito José Miró Nicolau, que había impartido conocimientos parecidos en el pasado, que me ayudó mucho en la preparación de la asignatura e impartió algunas clases.

Antes de empezar el curso se tomaron varias decisiones importantes.

- Como lo que se buscaba era un procedimiento de trabajo, exigimos a los alumnos que todo lo que hicieran —no sólo los resultados finales, sino los razonamientos, pruebas, errores… todo— se plasmara en un cuaderno de trabajo. Este cuaderno debía entregarse a final de curso y era evaluado.
- Como queríamos enfatizar que no existía 'la' solución del profesor, nunca resolvimos ni publicamos solución a ninguno de los ejercicios y situaciones planteadas en clase. Se les dijo a menudo que no había más solución que la que ellos produjeran.
- Los ejercicios y situaciones a utilizar no iban a ser en general los de tema matemático que estaban acostumbrados a resolver, sino que provendrían de ciencia e ingeniería. La redacción de los problemas no era formal y no teníamos especial cuidado en eliminar ambigüedades o en asegurarnos de que no faltaran o sobraran datos. Más bien lo contrario.

Existen varios métodos para enfrentarse a situaciones [5,10,11], pero consideramos el *professional method* de ver Planck y Teare [21] el más completo y nos basamos en él. En breve, el método consiste en aplicar los siguientes pasos:

1. A partir de la situación, definir un problema específico.
2. Planificar cómo atacar el problema, haciendo explícitas las simplificaciones pertinentes
3. Ejecutar el plan hasta llegar a una decisión o resultado
4. Comprobar la corrección de lo obtenido
5. Evaluar lo que se ha aprendido de la experiencia.

Naturalmente, estos pasos no necesariamente se dan en orden, y se debe iterar a través del método las veces que sea necesario.

Por ejemplo, ante la situación expuesta en la Sección 2.2, el joven informático empieza por recabar información, hablar con los usuarios y los responsables, observar el funcionamiento del programa y formularse una pregunta concreta a la que crea puede dar respuesta. Dado ahora el problema concreto que se ha propuesto a sí mismo, está preparado para actuar: decide qué simplificaciones debe hacer, sabe qué nuevos conocimientos puede necesitar, qué preguntas hacer a los desarrolladores, qué pruebas ha de hacer al programa. Lleva a cabo este plan de actuación, comprobando los resultados parciales que va obteniendo, asegurándose que no se ha cometido ningún error. Una vez respuesta su pregunta, evalúa lo que ha aprendido y está en condiciones de hacerse otra pregunta más pertinente, de suavizar alguna de las restricciones que se ha hecho, o de decidir cambiar completamente de línea de actuación.

3.2. Marcha del curso

Se matricularon ocho alumnos. Procuramos que la clase fuese eminentemente práctica, enseñando a aplicar el método sobre todo a base de ejemplos, aunque cuando era pertinente dábamos explicaciones teóricas. Desde la segunda semana los alumnos tuvieron a su disposición listas de ejercicios y situaciones simples para que fueran trabajando. La segunda mitad del trimestre se dedicó a una situación ya de tamaño y complejidad mayores, supuestamente real. Los alumnos debían entregar un informe final de su solución. Este informe debía ser breve y no iba dirigido al profesor, sino al jefe de la empresa, que no era ni científico ni ingeniero, y por lo tanto el

lenguaje a utilizar no podía ser técnico. Aún así, en el informe debía constar una recomendación de actuación, y debía convencer al jefe de la bondad de la misma. La nota final del curso dependía del cuaderno de trabajo y de este informe. Mientras los alumnos trabajaban en esta situación final, procuré abandonar el papel de profesor y convertirme en 'cliente', aunque algunas veces me vi forzado a volver al papel de tutor al ser aparente el desconcierto de los alumnos.

3.3. Resultados

Estoy seguro que los alumnos aprendieron mucho durante el curso, pero probablemente yo aprendí más. El verlos trabajar, y sobre todo el estudio de los cuadernos de trabajo fue muy ilustrativo. Más que explicar lo que hice en clase, qué en la segunda iteración va a cambiar considerablemente, creo más interesante mostrar lo que aprendí y lo que es importante para alguien que quiera montar un curso similar.

Este tipo de asignaturas son experienciales. Los conocimientos que los alumnos tengan simplemente a nivel mental sirven de poco. Puede que sepan responder a la pregunta '¿Por qué es importante comprobar tu trabajo?' pero después no lo aplican: lo saben pero no lo viven, no lo experimentan. El conocimiento experiencial es de absorción lenta [17] y para conseguirla hay que trabajar mucho y resolver muchas situaciones durante el curso. La estrategia de hacerles resolver una única situación de tamaño moderado no fue una buena idea.

Los alumnos tienen cincelados en sus mentes que existe 'la' solución, que es la del profesor, y que cualquier desvío de esta solución es errónea. Es decir, o piensan como el profesor o creen estar equivocados. Esto se vio en lo desconcertados que se encontraban al no tener una lista de soluciones. También les descolocaba mi respuesta cuando en clase o en tutorías preguntaban si lo que habían hecho estaba bien: les repetía que no había más solución que la que ellos crearan, y les daba pautas para que ellos mismos pudieran hacer comprobaciones. A pesar de que en clase se dedicó un tiempo considerable a enseñarles a comprobar sus resultados y a mi insistencia, en los cuadernos comprobé que casi nadie casi nunca intentó siquiera comprobar su trabajo. Casi se podía leer entre líneas: 'No sé si es correcto o no. No importa: ya me lo dirán'.

El no comprobar resultados tiene como efecto secundario una gran falta de seguridad en el trabajo producido. Cualquier comentario dicho de pasada en clase que les hiciera pensar que yo estaba considerando una vía de solución diferente de lo que ellos estaban haciendo, les llevaba a abandonar su enfoque de la situación. En un caso, en un cuaderno se podía leer 'Esto está mal porque el profesor ha dicho [un comentario que no necesariamente se aplicaba a su línea de trabajo]' para dos páginas después leerse 'Pues al final estaba bien porque el profesor ha dicho [otro comentario similar]'. El alumno ni siquiera se plantea que el resultado está bien o mal por sí mismo, y no por los comentarios casuales que diga o deje de decir el profesor, y, aún más importante, que puede comprobar por sí mismo si lo que ha hecho está bien o no.

Los alumnos consideran que las asignaturas son cotos cerrados sin relación entre ellas. Baste un ejemplo: en un momento dado tenían una variable estocástica y no sabían qué hacer con ella. No sabían dónde encontrar información y al final tuve que ser yo quien les sugiriera que mirasen los apuntes de la asignatura de probabilidad que habían cursado el año anterior. No se les había ocurrido. Realmente, entiendo un poco a los alumnos: los profesores también consideramos que nuestras asignaturas son cotos cerrados sin relación unas con otras.

En suma, una asignatura de este tipo debe ser eminentemente práctica, los alumnos deben resolver situaciones (para nota) repetidamente, debe ponerse mucho énfasis en la comprobación de la corrección de los resultados, debe hacérseles buscar información en lo aprendido en asignaturas anteriores y muchas otras fuentes, y el profesor debe de alguna manera procurar quitarse toda la autoridad que sea posible. Esto es lo que pienso aplicar la segunda vez que imparta la asignatura.

4. Conclusiones

Tras enseñar una asignatura donde se ha mostrado a los alumnos como desenvolverse ante situaciones he aprendido que problemas y situaciones son dos formas distintas de aprender la praxis de la profesión de informático. Los problemas son adecuados para anclar y

profundizar los conocimientos teóricos y por ello están supeditados a la teoría. Las situaciones son más cercanas a lo que el profesional se va a encontrar en su vida laboral y no se resuelven con las técnicas típicas de resolución de problemas. Enseñar a desenvolverse ante situaciones es difícil. Es difícil encontrar una asignatura donde se pueda hacer y es difícil porque lo que se enseña es una forma de trabajar, una experiencia. Estas dificultades se pueden obviar y podemos y debemos enseñar a nuestros alumnos a enfrentarse a situaciones ya que esto les dará una ventaja al iniciar su vida laboral.

Referencias

[1] T. Beaubouef, R. Lucas, y J. Howatt.. The *unlock* system: Enhancing problem solving skills in CS-1students. *SIGCSE Bulletin*, 33(2):43—46, Junio, 2001.

[2] Ewa Z. Bem. Experiment-based project in undergraduate computer architecture. *SIGCSE Bulletin*, 34(1):171—175, Marzo 2002.

[3] Grant Braught y David Reed. Disequilibration for teaching the scientific method in computer science. *SIGCSE Bulletin*, 34(1):106—110, Marzo 2002.

[4] Agustín Cernuda del Río. El modelo de desarrollo para un proyecto fin de carrera en ingeniería técnica en informática. *Actas de las VIII Jornadas de Enseñanza Universitaria de Informática, Jenui 2002*, pp. 559—566, Cáceres, Julio 2002.

[5] Peter Chalk. Scaffolding learning in virtual environments. *SIGCSE Bulletin*, 33(3):85—88, Septiembre 2001.

[6] A.T. Chamillard y Kim A. Braun. The software engineering capstone: structure and tradeoffs. *SIGCSE Bulletin*, 34(1):227—231, Marzo 2002..

[7] Tony Clear. "Programming in the large" and the need for professional discrimination. *SIGCSE Bulletin*, 33(4):9—10, Diciembre 2001.

[8] Tony Clear, Frank H. Young, Michael Goldweber, Paul M. Leidig, y Kirk Scott. Resources for instructors of capstone courses in computing. *SIGCSE Bulletin*, 33(4):93—113, Diciembre 2001.

[9] James B. Fenwick, Jr., Cindy Norris, y James Wilkes. Scientific experimentation via the matching game. *SIGCSE Bulletin*, 34(1):326—330, Marzo 2002.

[10] Center for Problem-Based Learning. Checking the fit. Página web, 2000. <http://www.imsa.edu/team/cpbl/whatis/Bisonproj/chkfit.html> (Última visita: Nov. 2002).

[11] Center for Problem-Based Learning. An introduction to problem-based learning. Página web, 2000. <http://www.imsa.edu/team/cpbl/whatis/whatis/slide1.html> (Última visita: Nov. 2002)..

[12] Ángel García Crespo. Mejora de la docencia mediante proyectos de investigación: Planificación estratégica de sistemas de información. *Actas de las IV Jornadas de Enseñanza Universitaria de Informática, Jenui 98*, pp. 113—120, Sant Julià de Lòria, Andorra, Julio 1998.

[13] David Ginat. On varying perspectives of problem decomposition. *SIGCSE Bulletin*, 34(1):331—335, Marzo 2002.

[14] Amruth N. Kumar. Learning the interaction between pointers and scope in C++. *SIGCSE Bulletin*, 33(3):45—48, Septiembre 2001.

[15] José Miró Julià. La enseñanza de la comunicación escrita en las carreras técnicas. Una experiencia. *Congreso universitario sobre innovación educativa en las enseñanzas técnicas, vol. I*, pp. 171—179, Zaragoza, 1996.

[16] José Miró Julià. Aprendizaje a través de la escritura. *Actas de las V Jornadas de Enseñanza Universitaria de Informática, Jenui 99*, pp. 205—210, La Almunia de Doña Godina, Zaragoza, Octubre 1999.

[17] Donald A. Norman. *The Design of Everyday Things*. MIT Press, 1998.

[18] Nick Parlante. Nifty assignments. Página web, 2002. <http://nifty.stanford.edu> (Última visita: Dec. 2002).

[19] Ángel Perles, Carlos Domínguez, Juan M. Martínez, y Houcine Hassan. Enseñanza de la informática industrial mediante proyectos reales simplificados. *Actas de las V Jornadas de Enseñanza Universitaria de Informática, Jenui 99*, pp. 55—60, La Almunia de Doña Godina, Zaragoza, Octubre 1999..

[20] Molecular Biology Teaching Pilot Program. Writing problem sets and questions. Página web, 2000. <http://www.princeton.edu/

bioteach/f-writing.html> (Última visita: Nov. 2002).

[21] D.W. Ver Plank y B.R. Teare, Jr. *Engineering Analysis: An Introduction to Professional Method.* Wiley and Sons, 1954.

[22] E.E. Villarreal y Dennis Butler. Giving computer science students a real-world experience. *SIGCSE Bulletin*, 30(1):40—44, Marzo 1998.

[23] Gloria Zaballa Pérez y Asunción Barredo Fuentes. Nueva metodología docente para adaptarse a las necesidades de la sociedad actual: adquisición de habilidades y destrezas. Un caso práctico: calidad en ingeniería del software. *Actas de las V Jornadas de Enseñanza Universitaria de Informática, Jenui 99*, pp. 89—96, La Almunia de Doña Godina, Zaragoza, Octubre 1999.

El futuro de la formación de los profesionales informáticos

Javier Oliver
Facultad de Ingeniería ESIDE
Universidad de Deusto
48080 Bilbao
e-mail: oliver@eside.deusto.es

Resumen

La actualización de los contenidos y los métodos de enseñanza en las titulaciones de informática debe llevarse a cabo de forma planificada y con unos objetivos claros para que sea una auténtica innovación pedagógica. En este trabajo se presentan algunos de los aspectos que conviene tener en cuenta en la formación de los profesionales informáticos de los próximos años. La inevitable convergencia de las universidades hacia un espacio europeo de educación superior supondrá una diversificación de las metodologías docentes que empleamos, aunque cada universidad seguirá manteniendo su independencia y determinando la forma concreta en la que realiza la innovación pedagógica. La elaboración de un perfil de los titulados en informática permite clarificar los objetivos que se quieren alcanzar en el proceso de enseñanza-apredizaje, pero conviene tener en cuenta las dificultades con las que se puede encontrar la puesta en marcha de un plan de innovación pedagógica y cuidar la formación del profesorado en metodologías pedagógicas.

1. Introducción

Los procesos formativos van cambiando a lo largo del tiempo, adaptándose a los medios disponibles y a los objetivos que se desean alcanzar. Desde los tiempos de la Grecia clásica, pasando por la transmisión de conocimientos en los monasterios medievales y llegando hasta nuestros días, hemos recorrido un largo camino. Las modificaciones en los procesos pedagógicos pueden ser triviales o significativas, y pueden consistir en la utilización de métodos importados de otros lugares, con lo que la innovación no tiene que ser necesariamente una invención, pero el aspecto clave que diferencia la innovación de un mero cambio sin rumbo es la planificación [5]. En los siguientes apartados se describen algunos de los aspectos que deben tenerse en cuenta a la hora de planificar y llevar a cabo el proceso de innovación pedagógica en las titulaciones de informática.

2. La convergencia universitaria europea

La demanda de los estudiantes europeos de una mayor movilidad entre las universidades de los distintos países y la voluntad de la Unión Europea de fortalecerse en todos sus ámbitos ha dado lugar a un debate sobre la actualización de las estructuras académicas de las universidades europeas [1].

Con esta actualización se pretende converger hacia unos principios básicos comunes que permitan la creación de un espacio europeo de educación superior por el que los estudiantes europeos se puedan mover más fácilmente, preparándose para un futuro laboral de ámbito europeo, y creando lazos profesionales y personales que aumenten la cohesión entre los países de la Unión.

La convergencia de las universidades hacia ese espacio europeo de educación superior deberá mantener la independencia de las instituciones académicas y huir de planteamientos rígidos y uniformizadores. No se pretende en ningún caso reducir la riqueza que supone la gran diversidad cultural europea, desarrollando planes de estudios únicos o controlando los contenidos, la docencia o la filosofía de los centros universitarios.

En este escenario que combina la convergencia y la autonomía docente, una de las ideas básicas es un sistema de créditos europeos (ECTS) que permita a los estudiantes ir acumulando en un registro sus actividades de aprendizaje, y poder transferir ese registro a la

universidad de su elección, promoviendo así una auténtica libertad de circulación de los estudiantes en Europa. El sistema de créditos europeos se basa no tanto en el número de horas de asistencia a clases magistrales como en los resultados del proceso de aprendizaje. Se estima que un crédito europeo equivaldrá a entre 25 y 30 horas de trabajo del alumno, incluyendo no solo las horas de clase tradicional sino también el tiempo empleado en realizar prácticas, trabajos, estudio personal o en grupo, evaluaciones, etc.

Este esquema va a promover el uso de metodologías docentes alternativas en las que las tecnologías de la información van a jugar un papel fundamental, permitiendo la distribución de contenidos vía Web, los foros de debate virtuales, las videoconferencias, la búsqueda rápida de información... Pero esto no significa que se proponga la desaparición de la clase magistral. En unas reflexiones preliminares acerca de los métodos de enseñanza y aprendizaje en el futuro espacio europeo de educación superior [1], se reconoce que la clase magistral puede ahorrar mucho tiempo al alumno presentándole de manera clara y sistemática los contenidos fundamentales de un tema. Pero se insiste en la necesidad de que el alumno realice también otras actividades como por ejemplo:

- Clases de problemas.
- Proyectos individuales o en pequeños grupos.
- Presentación de resultados en formato oral y escrito.
- Búsqueda de información relevante de manera eficiente, etc.

Por lo tanto, y desde el punto de vista pedagógico, el espacio europeo de educación superior va a suponer un énfasis en la diversificación de las técnicas docentes.

3. Directrices de cada centro

Todo proceso de innovación docente se lleva a cabo en un contexto determinado. En nuestro caso, una parte importante de ese contexto es el Plan Estratégico vigente en la Universidad de Deusto, que marca las líneas prioritarias de actuación para los próximos años. De entre esas líneas prioritarias, se puede destacar la innovación pedagógica, entendida como un esfuerzo planificado para adecuar nuestros métodos docentes a las demandas de los alumnos y de la sociedad [9].

Las directrices de innovación pedagógica son una guía de principios generales, para orientar el trabajo de modernización docente en la Universidad de Deusto. Cada facultad y departamento deberá adaptarlas a sus objetivos y características específicas. Los elementos fundamentales del Plan Estratégico de la Universidad de Deusto en cuanto a la innovación pedagógica son:

- Desarrollo en el alumno de *actitudes* positivas hacia su propio proceso de aprendizaje.
- Adquisición de *competencias* y valores por parte del alumno.
- Utilización de un *modelo* de aprendizaje autónomo y significativo.

El mundo de la informática es uno de los que evoluciona con más rapidez en la actualidad por lo que los futuros profesionales informáticos tendrán que asumir una *actitud* de formación continua durante toda su vida laboral. Por ello, además de una serie de contenidos cuidadosamente seleccionados, hemos de transmitir a los alumnos el gusto por el estudio y las técnicas para llevarlo a cabo con eficiencia.

En cuanto a los contenidos de los planes de estudios, tradicionalmente se ha hecho hincapié en los de tipo teórico, y últimamente en los de tipo práctico (*competencias*) específicos de cada asignatura. Pero las empresas cada vez exigen más a los titulados en informática, incluyendo competencias de tipo general (trabajo en equipo, dominio de idiomas, liderazgo, capacidad de negociación...), e incluso determinadas actitudes o valores. Es necesario debatir acerca de cuáles de estos contenidos se pueden o se deben transmitir en una titulación de informática e incorporarlos en su caso a la lista de objetivos a conseguir.

El aprendizaje no puede conducir solamente a repetir de memoria lo expuesto por otro. A través de una actitud activa y responsable, se pretende que el alumnado interiorice y haga realmente suyo lo aprendido. El *modelo* de aprendizaje significativo es un proceso cíclico que se divide en cuatro partes:

1. Contexto experiencial. En esta fase se pretende que lo que se va a aprender se vincule con algo de lo que ya se sabe. Hay que conseguir que el alumno construya su conocimiento relacionando sus ideas y conceptos previos con la estructura y contenidos de la materia que se presenta. También hay que motivar al alumno y despertar su interés, resaltando los aspectos más importantes del tema que se va a abordar.
2. Observación reflexiva: el alumno debe hacerse preguntas acerca del material que se le va a presentar: ¿cómo me afecta a mí, personalmente? ¿para qué me va a servir? ¿por qué (o por qué no) me interesa? Estas preguntas suponen una reflexión y un primer paso para la interiorización.
3. Conceptualización: es la asimilación por parte del alumno de los contenidos teóricos. No se trata de aprender de memoria, sino de aprender entendiendo. Tras el estudio individual, pueden trabajarse los contenidos teóricos en grupo.
4. Experimentación activa: actividad, frecuentemente realizada en equipo, que promueve la aplicación de los contenidos teóricos con el fin de comprenderlos mejor, resolver un problema o hacer un diseño o un desarrollo.

Teniendo en cuenta estas líneas maestras del Plan Estratégico en cuanto a innovación pedagógica, cada centro, titulación, departamento y en última instancia cada profesor en su asignatura puede ver cómo adaptar las líneas generales a su caso particular. La metodología que se aplique para conseguir los objetivos de innovación depende del contexto, pero se pueden citar algunas tareas como:

- Realización de encuestas para determinar el perfil que se quiere lograr en los titulados.
- Identificación de las asignaturas idóneas para el aprendizaje y evaluación de las competencias más generales.
- Búsqueda en cada asignatura de alguna actividad alternativa a la clase magistral que permita alcanzar de igual manera los objetivos docentes establecidos.
- Reconocer adecuadamente a los profesores que lleven a cabo acciones de innovación pedagógica.

4. Perfil del titulado

La innovación pedagógica exige tener claros los objetivos que se desean conseguir tras el proceso de enseñanza-aprendizaje, es decir, el perfil que deberá tener el futuro titulado en informática. Los titulados, al incorporarse a su puesto de trabajo, deberán satisfacer las necesidades de la empresa y de la sociedad en general, pero desde la universidad, demasiado a menudo no tenemos una visión clara de cuáles son esas necesidades. Por ello, en la definición del perfil del titulado deberán intervenir tanto los profesores de los departamentos universitarios como los exalumnos de las facultades, y las empresas que les darán trabajo y cualquier otro estamento social con el que los titulados puedan tener relación.

Todos los indicios apuntan a que las nuevas demandas de los estudiantes y de la sociedad, van a estar orientadas hacia un mayor énfasis en las competencias (conocimiento práctico) tanto específicas de un área de conocimientos como generales. El hecho de que los recursos empleados en el proceso de enseñanza-aprendizaje son finitos, obliga a revisar con espíritu crítico los esfuerzos que se dedican a la transmisión de conocimientos. En la era de la información y las telecomunicaciones, los conocimientos son cada vez más accesibles para los estudiantes y cada vez se quedan obsoletos más rápidamente. Por ello, en los próximos años tendremos que hacer más hincapié en desarrollar en los alumnos la capacidad de aprender a lo largo de toda su vida y de aplicar en la práctica lo aprendido, quizá a costa de reducir algo la cantidad de contenidos teóricos.

El alumno tendrá que adoptar un papel más activo en su aprendizaje, consiguiendo que éste sea más significativo y menos memorístico. Vamos hacia un escenario en el que los profesores, más que enseñar tendremos que facilitar, motivar y dirigir el aprendizaje de los alumnos. La atención se centrará en la actividad del alumno (aprender) más que en la del profesor (enseñar).

En este perfil o descripción de las características de los titulados hay que incluir:

- *Conocimientos*: cuáles son las áreas y los conocimientos teóricos con los que el titulado

deberá estar familiarizado, es decir, qué deberá saber.

- *Competencias*: qué deberá saber hacer. Los conocimientos teóricos son una base necesaria que en muchos casos hay que aplicar a la realización de tareas o resolución de problemas. Pero además de la aplicación práctica de conceptos de las distintas áreas de conocimiento, las competencias incluyen capacidades más generales, como hablar en público, manejar bibliografía o liderar un equipo de trabajo. Si queremos que nuestros alumnos desarrollen estas habilidades no podemos confiar en el azar. Hay que incluirlas en el perfil del titulado y en el programa de estudios, y hay que evaluarlas.
- *Actitudes*: son formas de pensar, prioridades, escalas de valores que sirven como herramienta y como guía para actuar de un forma determinada ante una situación concreta de la vida. ¿Cuáles deberán ser estos valores? Eso dependerá, entre otras cosas, del ideario de la institución universitaria y de las demandas de la sociedad. Pero si queremos que nuestros titulados sean respetuosos con el medio ambiente o sean tolerantes con el que es diferente o solidarios con el desfavorecido, alguna actividad habrá que llevar a cabo para intentar conseguirlo.

5. Dificultades para la innovación

Una vez tomada la decisión de llevar a cabo un proceso de innovación educativa, el primer paso es informar y motivar al profesorado, que es uno de los agentes fundamentales del cambio. La motivación deberá basarse en la constatación de que exalumnos, empresas y los propios profesores reclaman una actualización no sólo en los contenidos, sino también en los métodos de enseñanza.

Este proceso de información y motivación del profesorado para el cambio puede encontrarse con muchos impedimentos [4]:

- La falta de espíritu de equipo y la ausencia de objetivos y proyectos comunes.
- Las tensiones y los enfrentamientos interpersonales.
- La falta de planificación.
- La falta de coordinación.
- La aplicación excesivamente uniformizada de los planes de innovación sin tener en cuenta las especificidades de cada centro, departamento, asignatura o profesor.
- Formación pedagógica insuficiente y poco actualizada del profesorado.
- Falta de estímulos positivos que animen a realizar el esfuerzo que supone la innovación.
- Asignación insuficiente de recursos.
- Análisis erróneo de la situación de partida y falta de identificación de los puntos débiles
- Inercia al cambio y actitud defensiva de los profesores.
- Falta de una conexión clara entre la innovación en teoría y los métodos para llevarla a la práctica...

Uno de los obstáculos que hay que vencer a la hora de poner en marcha la innovación pedagógica, es la *inercia*, la tendencia que casi todos tenemos a ir acomodándonos a la realización de tareas conocidas y a rechazar la incertidumbre que supone cualquier cambio. A menudo se utiliza como argumento lo que en realidad es la constatación de una incapacidad de adaptación: “es que eso siempre se ha hecho así”. La repetición rutinaria de contenidos y metodologías transmite al profesor una sensación de seguridad y estabilidad. Sin embargo los profesores tenemos que llegar al convencimiento de que una actitud de permanente cuestionamiento de nuestras actividades nos va a permitir la incorporación gradual de innovaciones que vayan mejorando constantemente nuestros resultados.

Otra dificultad importante que hay que salvar para que la innovación pedagógica funcione es la ausencia de *espíritu de equipo*. La innovación es un trabajo de grupo en el que hay que involucrar al mayor número posible de personas. En ocasiones un individualismo extremo, o una libertad de cátedra entendida en su acepción más negativa puede llevar a algún profesor a actuar sin tener en cuenta el contexto en el que se encuentra, dificultando la consecución de los objetivos de las demás asignaturas. En otros casos no solo existe la desconexión sino una franca rivalidad, debida a enfrentamientos personales o a una competencia por recursos académicos o fondos financieros. El éxito de la innovación pedagógica depende de la creación de una conciencia de equipo entre todos

los profesores, de una serie de objetivos comunes que dejen en un segundo plano las legítimas discrepancias.

Para poder acometer con éxito una innovación pedagógica hay que partir de una situación en la que el profesorado se encuentre a gusto con su trabajo y con sus *condiciones laborales*. Si se dan situaciones de descontento, de excesiva carga de trabajo, con cambios constantes en los contenidos que provocan una sensación de no poder llegar a todo, interminables sesiones de corrección, reuniones, casos de amiguismo o de auténtica corrupción no se puede contar con el profesorado para llevar a cabo un proceso de innovación.

Una reforma educativa rara vez tiene éxito y cala de verdad en los agentes que tienen que desarrollarla si viene impuesta desde la dirección o se trata de una ley de obligado cumplimiento. La excesiva burocratización y rigidez de un plan de reforma es uno de los mayores enemigos para su realización. Es necesario *compaginar* la coordinación y la dirección del proceso con una cierta flexibilidad y adaptación a cada individuo, de manera que la dinámica del cambio vaya tanto de abajo a arriba como de arriba abajo en la organización.

6. Algunas técnicas docentes alternativas

La innovación pedagógica supone en algunos casos, la utilización de técnicas docentes con las que los profesores no están suficientemente familiarizados. Por este motivo, los profesores debemos tener una actitud de aprendizaje y actualización permanentes, no solamente en cuanto a los conocimientos de las asignaturas que impartimos, sino también en cuanto a las técnicas pedagógicas.

No se trata de hacer desaparecer la clase magistral de toda la vida, sino de complementarla con otras técnicas docentes que aprovechen mejor los distintos estilos de aprendizaje de los alumnos y que les ayuden a alcanzar ese perfil de conocimientos, competencias y actitudes que las empresas buscan y la sociedad en general les demanda.

Las empresas que contratan a los titulados en informática les van a pedir que sean capaces de buscar información y procesarla de manera crítica, hablar en público y expresarse con corrección y claridad, analizar problemas y proponer soluciones, etc. El modelo docente basado en clases magistrales no desarrolla estas capacidades, por lo que si se pretende que el alumno las adquiera, debe ser complementado con otras técnicas pedagógicas. A continuación se presentan algunas de estas técnicas que en función de factores como el tipo de asignatura, número de alumnos y experiencia del profesor, pueden servir para conseguir alguno de los objetivos del perfil docente de los titulados.

El *método del caso* consiste en presentar a los alumnos ejemplos de la vida real para que sirvan de base al aprendizaje [2]. El caso se presenta en forma de narración detallada, y su análisis suele incluir la participación activa de los alumnos. Es fundamental que los casos que se presentan a los alumnos se elijan cuidadosamente para conseguir los objetivos pretendidos. Algunas de las características de esta técnica docente son:

- El estudio de los casos se presta a utilizar el trabajo en grupo, con lo que el estudiante se familiariza con una forma de trabajo y toma de decisiones que será muy habitual en su vida laboral.
- Al tratarse de situaciones complejas, se fomenta la capacidad de análisis y la identificación de los aspectos fundamentales de un problema.
- Los puntos de vista diferentes pueden generar discusiones con lo que el profesor puede trabajar objetivos como la tolerancia, el respeto mutuo o la negociación.
- Es muy importante la capacidad del profesor para moderar la discusión, llevándola hacia los aspectos que más le interese destacar.

El *trabajo cooperativo* consiste en que los miembros de un grupo colaboren entre sí para obtener un objetivo común. Hay que destacar que no todo el trabajo en grupo es necesariamente trabajo cooperativo. En ocasiones se establecen relaciones de competencia entre los miembros de un grupo, con lo que cada individuo quiere destacar con respecto a los demás, llegando incluso a obstaculizar su trabajo. Para que un equipo de personas llegue a realizar un trabajo cooperativo de aprendizaje deben darse una serie de requisitos mínimos [7]:

- Interdependencia positiva entre los miembros del equipo, es decir, la percepción de que no se puede tener éxito en la tarea si no lo tienen también los demás miembros del equipo.
- Ayuda mutua. Una buena parte de la tarea de cada individuo es ayudar a los demás a que realicen la suya, y llevar a cabo una crítica constructiva del trabajo de los demás.
- Cada individuo debe hacerse responsable de la parte de trabajo que le corresponde, y no rehuir las tareas confiándoselas a los demás.
- Después de una sesión de aprendizaje cooperativo cada individuo debe estar mejor preparado para llevar a cabo esa misma tarea solo. De esta manera, el trabajo en equipo debe servir como potenciador del aprendizaje, dejando al individuo en mejores condiciones para el trabajo individual.
- Una buena comunicación interpersonal. Para que los miembros de un grupo puedan llevar a cabo una tarea en común, deberán conocerse y tener confianza mutua, ser capaces de comunicarse con precisión y sin ambigüedades, aceptarse y apoyarse entre sí y resolver los conflictos que puedan presentarse de manera eficaz.

Otra técnica docente que puede resultar de utilidad es el *aprendizaje basado en proyectos* [3]. Los alumnos tienen que crear un producto o desarrollar un servicio que satisfaga una necesidad real, y mientras trabajan de manera autónoma van elaborando su propio conocimiento. En este contexto, la exposición de contenidos por parte del profesor es solo un método más para adquirir información. El objetivo no es memorizar unos apuntes, sino llevar a cabo un proyecto que tiene una finalidad fuera del contexto del aula [6, 8]. Este tipo de actividad obliga a desarrollar capacidades muy valoradas en las empresas como la búsqueda y procesamiento crítico de la información, el trabajo en grupo y la resolución de problemas. Es importante elegir proyectos cuya realización permita alcanzar los objetivos de aprendizaje deseados.

La utilización de éstas y otras técnicas alternativas a la clase magistral no es sencilla, requiere un mínimo de experiencia del profesor, y puede no adaptarse a todas las situaciones de enseñanza-aprendizaje.

7. Conclusión

La actualización de los contenidos y métodos docentes de las titulaciones de informática es un proceso que debe estar planificado y tener unos objetivos claros.

La convergencia universitaria hacia un espacio europeo de educación superior significará un acercamiento entre las instituciones académicas y una diversificación de los métodos de enseñanza para la que debemos comenzar a prepararnos.

Se han presentado las directrices de innovación pedagógica del Plan Estratégico de la Universidad de Deusto, que se centran en desarrollar en los alumnos actitudes positivas hacia el aprendizaje, hacer hincapié en la adquisición de distintas competencias y basar el aprendizaje en un modelo autónomo y significativo.

La definición de un perfil de los titulados sirve para clarificar los objetivos que se pretenden conseguir. La puesta en marcha de la innovación se enfrenta a dificultades que conviene prever y conocer, y exige una formación continua del profesorado en aspectos de innovación pedagógica.

Referencias

[1] Área de Relaciones Internacionales, Universidad de Deusto. *Tuning Educational Structures in Europe.* 2002 (`http://www.relint.deusto.es/TUNINGProject/index.htm`)

[2] Asopa, V.N. y Beye, G. *Management of Agricultural Research: The case method.* 1997 (`http://www.fao.org/docrep/W7500E/w7500e0b.htm`)

[3] Buck Institute for Education. *An overview of project Based Learning.* 2002 (`http://www.bie.org/pbl/`)

[4] Carbonell, J. *La aventura de innovar.* Morata, 2001.

[5] Huberman, A.M. *Cómo se realizan los cambios en la educación: una contribución al estudio de la innovación.* Unesco: OIE, 1973.

[6] Kearsley, G. y Shneiderman B. *Engagement theory: a framework for technology based teaching and learning.* 1999

(http://home.sprynet.com/~gkearsley/engage.htm)

[7] Roger, T. y Johnson D.W. *An overview of cooperative learning.* Sin fecha (http://www.clcrc.com./pages/overviewpaper.html)

[8] Shneiderman, B. *A teaching/learning philosophy for the cyber-generation.* Computers & Education, 31, 1, 1998.

[9] Universidad de Deusto. *Plan Estratégico*. (http://www.deusto.es/infgeneral/presentacion/planestrategico/innovación/).

Repercusiones del futuro espacio europeo de educación superior sobre las titulaciones universitarias de Informática en España

Fermín Sánchez
Dept. de Arquitectura de Computadors
Universitat Politècnica de Catalunya

e-mail: fermin@ac.upc.es

María-Ribera Sancho
Dept. de Llenguatges i Sistemas informàtics
Universitat Politècnica de Catalunya

e-mail: ribera@lsi.upc.es

Resumen

Este artículo pretende ser una reflexión sobre las repercusiones que el proceso de convergencia europea en educación superior tendrá sobre las titulaciones universitarias de informática en España. En la sección 1 se revisa brevemente la historia de las principales reuniones mantenidas en la UE sobre el tema. La sección 2 describe cómo será el nuevo espacio europeo de educación superior. En la sección 3 se presentan diferentes opciones para los futuros planes de estudio de informática dentro del nuevo marco europeo. La sección 4 analiza el coste que supondrá implantar los nuevos planes de estudios. La sección 5 presenta algunas de las posibles acciones que se pueden realizar en el futuro inmediato y, finalmente, la sección 6 concluye el artículo.

1. Introducción

En Mayo de 1998, el ministro de Educación de Francia invitó a sus homónimos de Alemania, Italia e Inglaterra a los actos del 800 aniversario de la Universidad de la Sorbona. El 25 de Mayo, los cuatro ministros firmaron la *declaración de la Sorbona* [1], documento en el que se proponía la creación de un Espacio Europeo de Educación Superior (EEES) para promocionar la movilidad de los ciudadanos de la UE, aumentar la capacidad de creación de empleo y contribuir al desarrollo general de Europa. Este documento marcó el inicio del proceso de homologación de los estudios universitarios de los cuatro países firmantes, que adoptaron una estructura común de titulaciones basada en el modelo anglosajón de *bachelor*, *master* y *PhD*. En el documento se propone un modelo basado en un primer ciclo y un *master* con cursos semestrales y un sistema de créditos fácilmente convalidables, similar al ECTS (*European Credit Transfer System*) [2]. Se propone también animar a los estudiantes a cursar, al menos, un semestre fuera de sus países.

La siguiente reunión se realizó en Bolonia (Italia), y a ella fueron invitados todos los países europeos dispuestos a realizar reformas en su educación superior. Asistieron los representantes de 29 países. El 19 de Junio de 1999, los ministros europeos de educación firmaron una declaración conjunta, conocida como la *declaración de Bolonia* [3], donde se proponía la creación, antes de 2010, de un EEES orientado a conseguir dos objetivos estratégicos: el incremento de empleo en la UE y la captación de estudiantes y profesores de otras partes del mundo. Se definieron para ello los siguientes objetivos:

- Adoptar un sistema de titulaciones fácilmente comprensible y comparable en todos los países mediante la implantación de un *suplemento del diploma* [4].
- Tener un sistema basado en dos ciclos principales, denominados en el documento *bachelor* y *master*.
- Establecer un sistema de créditos compatible, similar al ECTS, que promocione la movilidad al permitir que sean transferibles y acumulables, independientemente de la institución que los acredite (que no tiene por qué ser universitaria).
- Promover la movilidad de estudiantes, profesores y personal administrativo de las universidades y otras instituciones de enseñanza superior.

- Promover la cooperación europea para garantizar la calidad de la enseñanza superior definiendo criterios y metodologías comparables.
- Promover una dimensión europea en la educación superior, haciendo particular énfasis en el desarrollo curricular, la cooperación entre instituciones, los esquemas de movilidad y planes de estudio y la integridad de la formación y la investigación.

La *declaración de Bolonia* contiene un plan de acción, procedimientos para su seguimiento e implementación y establece fases bienales de realización. Al final de cada fase se propone una conferencia ministerial para evaluar los logros conseguidos y establecer las directrices futuras.

Los países firmantes se comprometieron a cumplir los objetivos definidos y formaron dos grupos para realizar el seguimiento de la declaración: un primer grupo, constituido por representantes de todos los países firmantes, la Comisión Europea, la Confederación de Conferencias de rectores de la UE, la Asociación de Universidades Europeas, el ESIB-*The National Union of Students in Europe*, el Consejo de Europa y la *European Association of Institutions in Higher Education*, y un segundo grupo, más reducido, formado por los representantes de los países que sucesivamente ostentasen la presidencia de la UE hasta la realización de la próxima reunión (Finlandia, Portugal, Francia, Suecia y Bélgica), la República Checa –como organizadora del siguiente encuentro-, la Comisión Europea, la Confederación de Conferencias de rectores de la UE y la Asociación de Universidades Europeas.

Los dos grupos organizaron reuniones en Helsinki (Noviembre de 1999), Lisboa (Enero y Junio de 2000), París (Octubre y Diciembre de 2000), Leira (Noviembre de 2000) y Estocolmo (Abril de 2001). En Marzo de 2001 se celebró en Göteborg una convención de estudiantes y, en Salamanca, la convención preparatoria de la siguiente reunión ministerial, que se realizaría en Praga en Mayo de 2001. En la convención de Göteborg [5] los estudiantes declararon que el sistema debe garantizar la igualdad de todos los alumnos en el acceso a la educación superior y la necesidad de que los programas sean compatibles e intercambiables. En la convención de Salamanca [6] se permitió a las universidades, por primera vez en todo el proceso, participar activamente en las discusiones. Se remarcó que las universidades deben ejercer con responsabilidad su autonomía, que la educación es un servicio público, que la enseñanza superior se apoya en la investigación, que la diversidad debe ser considerada en el EEES y que la calidad es un factor esencial. Esta reunión se centró en regular el primer ciclo de los estudios y en ella se constató que todos los países estaban desarrollando su propio sistema de acreditación.

A Praga asistieron representantes de 32 países. El objetivo fundamental era evaluar los progresos conseguidos hasta entonces y definir las prioridades del futuro. Los asistentes firmaron el 19 de Mayo de 2001 un documento conocido como el *Comunicado de Praga* [7]. En él se manifiestan las divergencias existentes en los distintos países para obtener títulos equivalentes, la necesidad de adaptar la nueva estructura cíclica a las peculiaridades académicas y laborales de cada institución y la necesidad de definir un marco común de calificación, acreditación y certificación. Se constata la gran dimensión social que ha originado la *declaración de Bolonia*, se promueve el uso de las nuevas tecnologías en la enseñanza, se insiste en el desarrollo de mecanismos que garanticen la calidad de la educación y la investigación y se promueve la creación de planes de estudio que conduzcan a un perfil profesional o académico. Se hace especial énfasis en la importancia del aprendizaje a lo largo de toda la vida, en que las universidades y los estudiantes deben implicarse más en el proceso de creación del EEES y en aumentar el atractivo del sistema para estudiantes de fuera de la UE.

En la reunión de Praga se modificó la estructura de los grupos de trabajo, formando un grupo de seguimiento y otro de preparación. El grupo de seguimiento lo forman representantes de todos los estados firmantes, de la Comisión Europea y de los nuevos participantes, presididos por el país que presida la UE. El grupo de preparación está formado por representantes de los países que han organizado reuniones ministeriales previas, el que organizará la siguiente, la Comisión Europea, el país que presida la UE, dos estados miembros de la UE y dos estados no miembros. Los últimos cuatro representantes son nombrados por el otro grupo.

Antes de la próxima reunión ministerial está prevista la realización de distintos seminarios

internacionales sobre acreditación de la calidad, reconocimiento y uso de créditos en el proceso de Bolonia, desarrollo de títulos conjuntos y sobre la dimensión social del proceso, poniendo especial énfasis en los obstáculos de movilidad, formación continua y participación de los estudiantes.

La próxima reunión se realizará en Berlín los días 18 y 19 de Septiembre de 2003 [8,9]. A esta reunión asistirán, además de los ministros, los rectores de las universidades europeas, y se dedicará más atención al doctorado y a la acreditación de la calidad. Se espera que esta reunión aclare definitivamente el camino hacia un EEES que requiere, además de voluntad política, la implicación decidida de las universidades y demás asociaciones académicas.

2. El nuevo EEES

En los documentos mencionados en la sección anterior y en las reuniones de trabajo de los grupos de seguimiento se perfilaron las características fundamentales del nuevo EEES. La declaración de Bolonia se convirtió en el pilar fundamental del proceso de convergencia y en el manual de referencia que las universidades han utilizado para avanzar en este proceso.

Cada país de la UE decidirá si opta por la coexistencia temporal de las nuevas titulaciones con las antiguas (como se está haciendo en Alemania) o adopta de forma única el nuevo sistema (como se ha hecho en Italia).

2.1. Enseñanza y aprendizaje

La formación superior del futuro difiere considerablemente de la que se ha venido realizando hasta nuestros días. Tradicionalmente, el aprendizaje se hacía de forma intensiva durante unos pocos años y era suficiente para trabajar el resto de la vida. El puesto de trabajo y ocasionales y breves cursos de formación bastaban para mantenerse al día en la profesión. El vertiginoso crecimiento de las TIC en los últimos años ha destrozado ese paradigma, haciendo prácticamente imprescindible una formación continua en cada vez más profesiones. Los nuevos tiempos requieren que los alumnos "aprendan a aprender". Este aspecto es especialmente notable en las carreras de informática.

El cambio de paradigma educativo conlleva un cambio en la mentalidad del profesor: su objetivo pasa del "qué debo enseñar" al "qué debe aprender el alumno". Este cambio implica que el estudiante ha de participar de forma mucho más activa en el proceso de aprendizaje de lo que lo hace actualmente.

La estructura promovida por el EEES asume dos ciclos consecutivos, ambos con capacitación laboral. Dado que el primer ciclo proveerá a las empresas europeas de profesionales cualificados, debemos plantearnos si ha llegado el momento de realizar un cambio de modelo. En el modelo clásico, los cursos con mayor dificultad están situados en el nivel más bajo. En opinión de Benjamín Suárez [10], vicerrector de la UPC, el grado de dificultad de las asignaturas en el nuevo modelo educativo debería ser creciente a medida que se avance en el itinerario académico.

2.2. Elementos clave del EEES

Existen cuatro elementos clave en el EEES: la valoración mediante ECTS, la estructura cíclica, el suplemento del diploma y la evaluación, acreditación y certificación de los estudios.

El ECTS es la unidad de valoración de la actividad académica. Integra estudios teóricos y prácticos, otras actividades académicas dirigidas y el trabajo personal del estudiante. Es una valoración del trabajo del estudiante, no del profesor. Se trata de un valor numérico entre 1 y 60, donde 60 créditos representan el volumen de trabajo de un alumno durante un año académico completo. Los créditos ECTS pueden obtenerse también a través de la experiencia profesional. Un crédito ECTS es equivalente a una dedicación del estudiante de entre 25 y 30 horas [2].

Las titulaciones del EEES están divididas en dos ciclos cuya nomenclatura aún no se ha estandarizado. El primer ciclo es público y comprende entre 180 y 240 ECTS, cursados en tres ó cuatro años a tiempo completo. El título otorgado al acabar el primer ciclo tendrá un valor específico en el mercado laboral europeo. Además, en España este título dará acceso al nivel A de la función pública y, por lo tanto, al doctorado [11]. Para acceder al segundo ciclo es preciso haber completado con éxito el primero. El segundo ciclo permite la obtención de un *master* o de un doctorado.

El *suplemento del diploma* es un documento que acompaña a un título de educación superior y describe, en la lengua nacional correspondiente y en inglés, la naturaleza, el nivel, el contexto, el contenido y el estatus de los estudios que se han cursado y completado con éxito.

Finalmente, es preciso desarrollar mecanismos de evaluación, acreditación y certificación para garantizar la calidad de la enseñanza. Parece obvio que la evaluación y la certificación se realizarán en cada institución. Cabe preguntarse si de la acreditación se encargarán agencias estatales o independientes.

3. Las futuras titulaciones de informática

Debemos plantearnos cómo afectará el nuevo panorama educativo europeo a las titulaciones universitarias de informática del estado español.

La primera consecuencia clara de este cambio es la desaparición de las dos Ingenierías Técnicas de Informática de Gestión y de Sistemas, dado que se plantea la definición de un único primer ciclo que dará lugar a una titulación de ingeniero informático o similar.

Desde nuestro punto de vista la Ingeniería Informática también desaparece como tal, puesto que la adaptación al nuevo marco requiere mucho más que una simple reestructuración de esta titulación si se pretende garantizar la capacitación laboral al final del primer ciclo.

Por lo tanto, es oportuno preguntarse cómo estructurar los estudios de informática en dos ciclos. Entendemos que la forma en que se defina esta ciclicidad dependerá de las características de cada titulación y puede diferir de una a otra.

Para el caso concreto de la informática existe, que nosotros sepamos, un solo texto de referencia: el informe que la CODDI (Conferencia de Decanos y Directores de Informática) emitió como resultado de su reunión del 16 de Mayo de 2002 [12]. Como complemento, podemos citar la ponencia de Recober en las jornadas de reflexión que la UPC organizó sobre el tema [13].

Las conclusiones de los decanos indican que un primer ciclo de tres años parece insuficiente para adquirir la capacitación laboral esperada en un ingeniero informático, sobre todo si se pretende que el estudiante realice un proyecto final de carrera como parte de su formación. Esta opinión se sustenta, básicamente, en los resultados de las actuales ingenierías técnicas en cuanto a rendimiento de los estudiantes. Por otra parte, si este primer ciclo debe dar acceso al nivel A de la función pública española y al doctorado, la legalidad vigente establece un mínimo de 4 años para estas titulaciones.

Consecuentemente, para el caso de la informática se plantea como modelo más adecuado un primer ciclo de 4 años de carácter "generalista", que incluya la realización de un proyecto final de carrera, y un segundo ciclo de entre 1 y 2 años de duración, en función de la especialización que se pretenda conseguir en cada caso.

Otro aspecto interesante que plantea Recober [13] es el de la puesta en marcha del nuevo modelo en cada centro. Se pregunta si resulta más conveniente comenzar experimentalmente por el primer ciclo o por el segundo y recuerda que algunas de las titulaciones actuales de informática comenzaron como un segundo ciclo. Desde nuestro punto de vista, iniciar el segundo ciclo antes que el primero resulta menos arriesgado por el menor número de estudiantes involucrados y porque permite ajustar, según las conveniencias del centro, qué segundos ciclos se imparten. Por otra parte, opinamos que los beneficios del nuevo modelo educativo son más evidentes para el primer ciclo que para el segundo y que, por tanto, conviene poner en marcha los primeros ciclos cuanto antes mejor.

3.1. El primer ciclo

Una vez planteados los aspectos generales de la posible ciclicidad en los estudios de informática, expondremos algunos elementos propios de cada ciclo.

El primer ciclo se define como público en la LOU [11]. Parece claro que no corresponde a ninguna de las actuales ingenierías técnicas, ni tampoco al primer ciclo ni a una adaptación de la actual ingeniería informática. Para mayor dificultad, el informe de los decanos [12] indica que el mercado laboral español todavía no ha conseguido diferenciar entre un ingeniero técnico (de gestión o de sistemas) y un ingeniero informático. Hay que plantearse con urgencia, por lo tanto, qué tipo de ingeniero informático debe formarse en el primer ciclo.

Por otra parte, la desaparición de las titulaciones actuales y la creación de la titulación de primer ciclo tienen como consecuencia la necesidad de definir las nuevas troncalidades antes de diseñar los nuevos planes de estudio. Esto afecta muy especialmente a la titulación de primer ciclo porque es la que, de forma natural, debería tener una troncalidad mayor.

Para garantizar la capacitación laboral del estudiante al finalizar el ciclo parece adecuado diseñar el curriculum de arriba abajo: desde las competencias profesionales hasta los planes de estudio (contenidos, habilidades, etc.) [16]. La existencia de un proyecto final de carrera o de estancias en empresas también contribuye claramente a conseguir este objetivo.

Otro elemento a tener en cuenta en el diseño de los nuevos planes de estudio de primer ciclo es el hecho de que su plena implantación no llegará, presumiblemente, antes del 2012, momento en que los primeros titulados saldrán al mercado laboral. Aunque sea difícil, es preciso tener esta perspectiva temporal y prever, en lo posible, la evolución tecnológica y del mercado de trabajo.

En nuestra opinión, la calidad de la titulación de primer ciclo es fundamental porque determinará la futura movilidad internacional de nuestros estudiantes y su participación en la construcción del nuevo entorno laboral europeo.

3.2. El segundo ciclo

Pese a que tanto en la declaración de Bolonia como en el comunicado de Praga se insiste en que la educación es un bien público que ha de quedar bajo responsabilidad pública, el segundo ciclo no se define ni como público ni como privado en la LOU [11]. Este hecho alimenta la sospecha de que no recibirá subvención pública (o, al menos, no tanta como el primer ciclo).

La duda fundamental en el caso del segundo ciclo está, a nuestro modo de ver, en cuál ha de ser su orientación. ¿Debe entenderse como formación inicial (igual que los segundos ciclos actuales), como formación continua o como ambas cosas? Se habla, por un lado, de segundos ciclos de especialización profesional y por otro de segundos ciclos de orientación académica, destinados al doctorado. La CRUE está debatiendo, por ejemplo, sobre los estudios de matemáticas [14]. En el documento se hace una propuesta de *masters* con diferentes orientaciones: *masters* "profesionales", orientados a la integración del estudiante en la empresa y *masters* "científicos", con vocación más académica, enfocados como un inicio a la investigación y que pueden conducir a la elaboración de una tesis doctoral. Este modelo puede extrapolarse fácilmente a cualquier ingeniería. Para los segundos ciclos de especialización profesional debería considerarse la necesidad de potenciar el profesorado procedente del entorno empresarial. En definitiva, pensamos que es imprescindible una clarificación y posicionamiento institucional en este sentido.

Contrariamente a lo que sucedía con el primer ciclo, y a excepción del doctorado, opinamos que la calidad de los segundos ciclos no marcará de un modo tan determinante la movilidad internacional de los estudiantes pero, aún así, ¿hay que ofrecer lo que pida el mercado, o los segundos ciclos resultan el marco más adecuado para que la universidad influya en la sociedad del futuro?

Respecto al número de estudiantes que cursarán segundos ciclos, es razonable pensar que será muy inferior al actual, sobre todo si no recibe la adecuada subvención pública, y que muchos estudiantes deberán compatibilizar los estudios con su trabajo. Esto permitirá realizar la docencia en grupos reducidos, con alumnos más interesados en las materias que en la actualidad.

Por otra parte, pensamos que las técnicas de formación semipresencial se convertirán en una herramienta indispensable para permitir que los estudiantes puedan compaginar adecuadamente los estudios de segundo ciclo con sus actividades profesionales.

3.3. Homologación de los títulos actuales

Nos parece muy prematuro adelantar nada sobre este tema, dado el presente estado de indefinición. Es difícil aventurar si los actuales títulos de ingenieros técnicos son homologables al futuro primer ciclo o si los actuales títulos de ingeniero informático son homologables a alguno de los futuros *masters*.

Desde nuestro punto de vista, parece razonable que los actuales ingenieros técnicos puedan acceder al segundo ciclo directamente o con muy pocos complementos de formación, y tampoco es descabellado que un ingeniero informático actual, en función de su perfil

académico, de su experiencia profesional y de los *masters* que imparta cada centro pueda convalidar su título por uno de estos *masters*.

4. El coste del cambio

Es indiscutible que el proceso de convergencia hacia el EEES tiene asociado un coste, tanto económico como humano, que no puede ser obviado. El coste será menor si sólo el primer ciclo es público y los recursos actuales, como mínimo, se mantienen. En esta sección intentaremos analizar algunos de los factores más importantes que repercutirán sobre dicho coste.

Desde el punto de vista estrictamente económico, el descenso de la natalidad producido en España durante los últimos años ha empezado ya a provocar una disminución del número de alumnos que acceden a la universidad. Este descenso continuará presumiblemente hasta 2007, año a partir del cual se verá compensado con el aumento de población debida a la emigración [13]. Un descenso en el número de alumnos de las titulaciones puede conducir a una reducción en el número de grupos de las asignaturas –y de profesores-, especialmente de primer curso, a una reducción del número de alumnos por grupo o a ambas cosas. Estos datos parecen avalar que el descenso de natalidad reducirá el coste económico futuro de la mayoría de las titulaciones. No obstante, en el caso de la informática parece que todavía no se ha notado el impacto de este efecto. A modo de ejemplo, la Facultat d'Informàtica de Barcelona es el único centro de la UPC en el que el número de solicitudes de entrada de alumnos en primera opción en el curso 2002-2003 no ha disminuido. Además, la nota de corte ha aumentado manteniendo el mismo volumen de entrada que el curso anterior [15] lo que apunta a que, al menos en la UPC, la informática esquiva de momento el efecto de la disminución de la natalidad. Si el efecto comienza a notarse en informática en los próximos años, y los recursos no disminuyen, se plantea una buena oportunidad para mejorar las técnicas docentes.

Pero es el coste humano probablemente el más preocupante. En primer lugar, para asignar adecuadamente los ECTS a cada asignatura es preciso recopilar y procesar información detallada sobre el tiempo y trabajo que un estudiante medio dedica a asimilar la materia del curso y a realizar las distintas tareas. Por otra parte, el hecho de que la dedicación del estudiante no deba superar las 1600 horas/año -40 horas/semana-, implica un esfuerzo suplementario de coordinación y planificación de las diferentes asignaturas para que, a diferencia de lo que ocurre en la actualidad, la entrega de prácticas y la realización de exámenes no se acumule en un corto espacio de tiempo, normalmente al final del curso.

El cambio de paradigma educativo, pasar de la "enseñanza del profesor" al "aprendizaje del alumno", requiere disponer de una buena documentación de las asignaturas -en la mayoría de los casos mucho mejor que la que se ofrece actualmente-, la adaptación del profesorado a nuevos métodos docentes –sensiblemente distintos de las clases magistrales- y el aprovechamiento de las oportunidades que ofrecen las nuevas tecnologías. Este último factor no resultará complicado de superar en las titulaciones de Informática, pero ofrecerá grandes problemas en las carreras típicamente "de letras". Sin embargo, la reducción de las horas "presenciales" de clase obligará a la creación de mecanismos que permitan evaluar y valorar el trabajo de preparación de las clases y atención de los alumnos por parte del profesor, trabajo que seguramente será mayor que en la actualidad.

Cabría preguntarse si lo expuesto en el párrafo anterior implica realmente un coste adicional. Nuestras asignaturas deberían tener una documentación adecuada, y nuestros métodos docentes y aprovechamiento de las TIC deberían estar acorde con los tiempos que corren. Lamentablemente, la forma en que se valora hoy en día la labor docente del profesor –exclusivamente las "horas de pizarra"- hace que aspectos tan importantes como los citados anteriormente se hayan descuidado en la mayoría de los casos. Por ejemplo, las "horas de atención a los alumnos", que suponen casi la mitad de la carga docente de un profesor, se desaprovechan en muchos casos debido a que los alumnos no acuden a consultar dudas con los profesores. Estas horas desaprovechadas podrían dedicarse, por ejemplo, a subsanar las deficiencias encontradas en cada asignatura, contribuyendo así a una mejora global del sistema y, probablemente, del rendimiento académico de los alumnos.

El nuevo modelo, basado en el aprendizaje, requiere una atención mucho más personalizada a

los alumnos, y sólo puede conseguirse con grupos mucho más reducidos que en la actualidad. En grupos de tamaño medio puede ensayarse un modelo basado en la evaluación continua, pero la presión a la que se ven sometidos los estudiantes, que "continuamente" son evaluados de las distintas materias matriculadas, repercute negativamente en su rendimiento. En cualquier caso, el viejo tópico del profesor impartiendo una clase magistral a un grupo de 200 ó más alumnos, clásico en algunas carreras, podría ser desterrado definitivamente.

Pero, ¿hasta qué punto está dispuesto el estado español a aumentar los recursos de las universidades públicas para afrontar los cambios que se avecinan? Si tenemos en cuenta la historia más reciente, nos declaramos bastante pesimistas al respecto.

5. Adelantándonos al futuro

Es posible adelantar parte del trabajo a realizar poniendo en marcha algunas acciones mientras no esté elaborada la definición de materias troncales.

La primera acción a tener en cuenta consiste en actualizar los créditos de los actuales planes de estudio a los nuevos ECTS. Esto podría hacerse mediante una traducción directa, pero creemos que eso sería un grave error. Los actuales créditos valoran el número de horas lectivas del profesor, que no guarda relación necesariamente con el número de horas de dedicación que se exige a un estudiante para que asimile la materia. Por lo tanto, parece razonable que la realización de este cambio pase por hacer una estimación real del trabajo que se exige al alumno en cada asignatura, lo cual no es una tarea sencilla ni rápida, y exige una reflexión profunda sobre los objetivos que se persiguen y los conocimientos y habilidades que el alumno debe adquirir.

Una segunda acción consistiría en implantar, cuanto antes, la existencia de un *suplemento del diploma* para los planes de estudio vigentes. La nueva estructura *bachelor-master* permite la existencia de varios *mastes* distintos, cada uno de ellos con su propio suplemento. El suplemento del *bachelor*, por el contrario, parece que sería único. Con el objeto de avanzar en la definición de la estructura cíclica, y especialmente de los *masters*, podría ser de gran ayuda estructurar los actuales planes de estudios de forma que se definiesen "perfiles" que reflejasen las competencias profesionales que adquieren los titulados que siguen un determinado itinerario académico [16], como por ejemplo se está haciendo en la Facultat d'Informàtica de la UPV. Podría asociarse un *suplemento del diploma* distinto a cada uno de estos "perfiles", en el cual se reflejasen las competencias profesionales adquiridas.

Por otra parte, es importante comenzar a reconocer y valorar cuanto antes el trabajo docente "no lectivo" del profesor, ya que en la nueva estructura la docencia presencial perderá peso frente a las tareas no directamente presenciales del profesor, como pueden ser la dirección y participación en tribunales de proyectos de final de carrera y otros trabajos de los alumnos, la coordinación de asignaturas, la elaboración de material docente (apuntes, libros, prácticas, colecciones de problemas, etc.), la preparación de asignaturas nuevas o el cambio de temario de asignaturas ya existentes, la participación en jornadas de trabajo relacionadas con la docencia (como las JENUI, por ejemplo), las tutorías de los alumnos, etc. La mayoría de las tareas citadas no se encuentran suficientemente valoradas en la actualidad en las universidades españolas. Es preciso, por ello, detallar cuanto antes lo que se espera, desde el punto de vista docente, de un profesor en la universidad española, y exigir y valorar adecuadamente su cumplimiento.

Un gran número de universidades han puesto manos a la obra y han comenzado ya a adaptar sus titulaciones pensando en el futuro marco europeo. A modo de ejemplo, la UAB ha adaptado su diplomatura de turismo al nuevo paradigma educativo basado en el aprendizaje del alumno en lugar de en la enseñanza del profesor [17]. Para ello, han elaborado una lista detallada de los objetivos de cada titulación, de cada curso y de cada una de las asignaturas, tal como se proponía en [18]. A partir de ellos han desarrollado los temarios de las asignaturas, definiendo los objetivos específicos para cada tema, la metodología docente a utilizar, los recursos y herramientas necesarios para el correcto aprendizaje y la documentación de referencia y el tiempo estimado de dedicación del alumno para cada uno de los temas. Además, han clasificado los temas entre imprescindibles, importantes y secundarios.

Finalmente, dentro del ámbito de las TIC hay que destacar los trabajos desarrollados en el marco de la comunidad europea por *Career Space*, un consorcio formado por once grandes compañías de las TIC y la Asociación Europea de Industrias de TIC, para definir el currículo y capacidades profesionales de los futuros ingenieros [19,20].

Creemos que es muy importante reflexionar en profundidad sobre los aspectos planteados en este artículo y comenzar a pensar y actuar cuanto antes para intentar influir en las decisiones del estado español sobre este tema, en lugar de intentar adaptarnos una vez hayan sido tomadas.

En [21] y [22] puede encontrase información actualizada sobre muchas de las cuestiones que se han discutido en este artículo.

6. Conclusiones

En este artículo se ha hecho una reflexión sobre el efecto que el nuevo EEES tendrá sobre las titulaciones universitarias de informática en España. Para ello, en primer lugar se ha descrito el marco de referencia: las diferentes declaraciones de los ministros de la UE al respecto del tema. A continuación se ha descrito el nuevo EEES y se ha conjeturado sobre cómo podrían ser las titulaciones de Informática dentro del nuevo marco. Finalmente se ha disertado sobre el coste económico y humano que supondrá el cambio y se han detallado algunas de las acciones que pueden tomarse de forma inmediata para acercarnos cuanto antes al nuevo marco europeo.

Agradecimientos

Queremos agradecer a la Facultat d'Informática de Barcelona y al Departament d'Arquitectura de Computadors de la UPC su soporte a este trabajo.

Referencias

[1] http://www.universia.es/contenidos/universidades/documentos/Universidades_docum_Sorbona.htm
[2] http://www.ucm.es/info/vestud/Convergencia/Etcs.htm
[3] http://www.universia.es/contenidos/universidades/documentos/Universidades_docum_Bolonia.htm
[4] http://www.ucm.es/info/vestud/Convergencia/Suplemento.htm
[5] http://www.ucm.es/info/vestud/Convergencia/Student_documents_ESIB.pdf
[6] http://www.ucm.es/info/vestud/Convergencia/salamanca_convention.pdf
[7] http://www.ucm.es/info/vestud/Convergencia/Prague_communiquTheta.pdf
[8] http://www.ucm.es/info/vestud/Convergencia/from_prague_berlin.pdf
[9] http://www.bologna-berlin2003.de/en/prague_berlin/index.htm
[10] *La UPC fa Europa*, Jornades de treball Octubre Noviembre 2002, http://www.upc.es/upcfaeuropa/
[11] Ley Orgánica de universidades, BOE 24-12-2001, Sección 1, http://www.boe.es/boe/dias/2001-12-24/seccion1.html
[12] *Informe sobre la adaptación de los estudios de las ingenierías en informática a la declaración de Bolonia*, Conferencia de Decanos y Directores de Informàtica, CODDI02, Barcelona 16-17 /5/2002
[13] *Alternatives per a l'adopció del model bachelor-master: el cas de les TIC*, M.M.Recober, Jornades de Treball "L'UPC fa Europa", Octubre 2002, http://www.upc.es/upcfaeuropa/catala/documents/presentacions/presentacions.htm
[14] J.M.Bayod et al., *Documento de trabajo sobre la integración de los estudios españoles de matemáticas en el espacio europeo de enseñanza superior,* CRUE, Octubre 2002, http://www.upc.es/upcfaeuropa/catala/documents/referencies/matematicas.pdf
[15] *Dades estadístiques i de gestió de la UPC, Mayo 2002.*
[16] M.Valero-García y J.J.Navarro. *Niveles de competencia de los objetivos formativos de las ingenierías.* VII Jornadas sobre la enseñanza Universitaria de la informática, JENUI'2001, Julio 2001, pag. 149-154.
[17] G. Roselló, *La Universitat de Barcelona i l'espai europeu d'educació superior*, Jornades de Treball "L'UPC fa Europa", Octubre 2002, http://www.upc.es/upcfaeuropa/catala/documents/presentacions/presentacions.htm
[18] J.J.Navarro, M.Valero-García, F.Sánchez y J.Tubella. *Formulación de los objetivos de una asignatura en tres niveles jerárquicos*. VI Jornadas sobre la enseñanza Universitaria de la informática, JENUI'2000, Sept. 2000, pag. 457-462.
[19] *Directrices para el desarrollo curricular*, Career Space, CEDEFOP. www.cedefop.eu.int
[20] *Perfiles de capacidades profesionales genéricas de TIC*, Career Space, CEDEFOP. www.cedefop.eu.int
[21] http://www.mec.es/consejou/
[22] http://www.univ.mecd.es/

Enseñando Inteligencia Emocional a Ingenieros en Informática

Esperanza Marcos
Grupo Kybele
Universidad Rey Juan Carlos.
28933 Móstoles (Madrid)
Email: cuca@escet.urjc.es

José María Cavero Barca
Grupo Kybele
Universidad Rey Juan Carlos.
28933 Móstoles (Madrid)
Email: jmcavero@escet.urjc.es

Resumen

Cada día más, se está solicitando de los profesionales informáticos que, además, e incluso antes, de conocimientos técnicos, posean habilidades emocionales. Habilidades para el trabajo en equipo o para hablar en público, capacidad de liderazgo o de adaptación al cambio, son requisitos importantes, quizá, en cualquier trabajo. Sin embargo, este tipo de habilidades, propias de lo que se viene denominando *Inteligencia Emocional*, se convierten en características imprescindibles para el perfil de un Ingeniero en Informática. Este tipo de profesionales deberá, con toda seguridad, trabajar en grupo, enfrentarse a la dirección de proyectos, impartir charlas y, lo que puede resultar aún más difícil, mantenerse al día en una tecnología cuyos nuevos avances estarán obsoletos en apenas cinco años. Sin embargo, y a pesar de la importancia de este tipo de habilidades, la Inteligencia Emocional continúa siendo la asignatura pendiente en los currículos de Ingenierías Informáticas. En este artículo se presenta un experimento en el que se trata de incentivar, mediante un concurso, la Inteligencia Emocional de alumnos de Ingeniería Informática, dentro de una asignatura de Bases de Datos.

1. Introducción

El sistema de enseñanza clásico que venimos utilizando en la Universidad española parece que no es el más adecuado para transmitir al alumno un tipo de conocimiento que cada vez se considera más importante en la formación de nuestros futuros profesionales. Se trata, por una parte, de invertir el papel pasivo del alumno, haciendo que sea él el que aprenda. Hay que tener en cuenta que *enseñar no puede ser sólo transmitir conocimientos*, ya que esto implicaría que quien aprende representa un papel pasivo y esto no es deseable, máxime teniendo en cuenta el volumen actual de información en cualquier área del saber y, aún más si cabe, en Informática. Por otra parte, y además de incentivar la participación del alumno en su propio aprendizaje, se trata de potenciar en él una serie de habilidades (a las que podemos referirnos como habilidades emocionales) que le permitirán manejar con más destreza los conflictos y problemas profesionales, la toma de decisiones, mejorar sus habilidades de trabajo en equipo o de liderazgo, etc.

En este sentido van también las directrices europeas de enseñanza universitaria. La universidad española comienza a invertir grandes esfuerzos en la adaptación a la Declaración de Bolonia, [17]; uno de los principales cambios consiste en el concepto de *crédito ECTS*. En la actualidad, un crédito se mide por el número de horas impartidas por el profesor, en las que el alumno no es sino mero receptor pasivo. El crédito ECTS, sin embargo, es una medida del esfuerzo del alumno que incluye no sólo las horas de clase lectivas, sino también las horas que el alumno requiere para estudiar la asignatura, horas de búsqueda bibliográfica, seminarios, etc.

En beneficio del alumno, debemos atender al desarrollo en aspectos no sólo intelectuales, sino también interpersonales y afectivo-motivacionales, dado que estas dimensiones concurren en el rendimiento del alumno y en su motivación personal, además de afectar el clima social de la clase. Es lo que las nuevas tendencias de la psicología denominan *inteligencia emocional*, [5,8,9,18], y cuyo cociente, según los expertos, unido al cociente intelectual, determina los alumnos que realmente triunfarán en el desempeño de su profesión. Goleman en su libro "La Práctica de la Inteligencia Emocional" explica cómo *"El aprendizaje académico solo sirve para diferenciar a los trabajadores "estrella" en unos pocos de los quinientos o seiscientos trabajos en los que hemos llevado a cabo estudios de*

competencia. [...] Lo que realmente importa para el desempeño superior son las habilidades propias de la inteligencia emocional", ([9], pág. 39). Además, de los cientos de experimentos realizados por Goleman, se puede concluir que las habilidades sociales son especialmente relevantes para los programadores: *"Uno de los campos en los que curiosamente más incide la inteligencia emocional es el de la programación informática, un campo donde la eficacia de la elite que ocupa el 10% superior es un 320% mayor que la de los programadores promedio, algo que, en el caso de la "rara avis" que conforman el 1% más elevado ¡llega a alcanzar el 1.272%!"*, ([9], pág. 62).

Además de formar al alumno en el curriculum propio de cada asignatura, un objetivo general de la enseñanza universitaria y, por tanto, común a todas las asignaturas, debería ser el de fomentar la inteligencia emocional del alumno, ya que ésta parece ser una de las mayores carencias en los técnicos informáticos [9]. Goleman indica que esta enseñanza es fundamental en las universidades donde deben de formarse estos técnicos. *"Todas estas evidencias han espoleado a las universidades a asegurarse de que los nuevos ingenieros y científicos que accedan al mundo laboral sean más competentes en el campo de la inteligencia emocional....Hasta el momento, la formación de los ingenieros ha ignorado esta clase de habilidades pero ya no puede seguir permitiéndose ese lujo"*, ([9], pág 74). Como dato significativo de entre los muchos proporcionados por Goleman para apoyar su tesis, podemos destacar las palabras de John Seely Brown, director de I+D de Xerox Corporation, en Silicon Valley quien afirma: *"En todos los años que llevo aquí, nunca he mirado el historial universitario de nadie porque las dos competencias que más valoramos son la intuición y el entusiasmo. Buscamos personas atrevidas. Osadas, pero que, al mismo tiempo, se sientan seguras de sí mismas"*, ([9], pág. 74).

En la Northwest Missouri State University se ha implantado un sistema de evaluación del curriculum en informática y en él se destacan las habilidades que se esperan de los titulados en informática [13]: sólidos conocimientos técnicos en informática; buenas habilidades de comunicación; capacidad de trabajar eficazmente en grupo; buenas habilidades para la resolución de problemas; poseer un nivel alto de satisfacción con el programa de informática de la universidad en la que se formaron.

La necesidad de este tipo de habilidades se pone también de manifiesto en el reciente informe *"Future skills for tomorrow's world"* [7], en el que las principales compañías relacionadas con las TIC (Tecnologías de la Información y de las Comunicaciones) europeas[1] plantean los perfiles profesionales que requieren. El proyecto, patrocinado por la Comisión Europea, pone de manifiesto la necesidad de profesionales que sean creativos y artísticos; entusiasmados por la tecnología; con conocimientos matemáticos y científicos; que posean buenas habilidades de comunicación; capaces de tratar con la gente; que quieran trabajar como parte de un grupo. Detalla también las habilidades técnicas que requieren (entre las que se destacan los conceptos de BD), así como las habilidades de comportamiento (que corresponden a las emocionales de Goleman): Analítica, Atención al detalle, Compromiso con la excelencia, Comunicación, Iniciativa, Creatividad, Orientación al cliente, Liderazgo, Manejo de riesgos, Negociación, Persuasión, Planificación y Organización, Capacidad de Relacionarse, Estrategia y Planificación, Trabajo en Equipo y Orientación Técnica e Interés.

Por todo ello, consideramos un objetivo prioritario fomentar en el alumno este tipo de habilidades y destrezas emocionales que serán definitivas a la hora de introducirse, mantenerse y triunfar en el mundo laboral. Sin embargo, la mayor parte de los currículos internacionales [1, 10], no proponen asignaturas que formen de modo explícito en tales competencias. Particularmente, los currículos de las universidades españolas tampoco lo hacen. Es por ello, que consideramos imprescindible incorporar en la metodología docente de nuestras asignaturas, prácticas didácticas que permitan formar al alumno en este tipo de habilidades.

El resto del artículo se estructura del siguiente modo: el apartado 2 describe el sistema de prácticas usado en la asignatura de Bases de Datos de Ingeniería Informática de la Universidad Rey Juan Carlos, especificando qué tipo de habilidades se pretendía fomentar con este sistema; el

[1] IBM Europa, Nokia Telecommunications, Philips Semiconductors, Thomson CSF, Siemens AG, Microsoft Europa, British Telecommunications PCL.

apartado 3 es el resultado de la evaluación del método mediante encuesta a los alumnos y al profesor de la asignatura; por último, en el apartado 4, se resumen las principales conclusiones, así como las mejoras que podrían realizarse en el método.

2. Aprendiendo Bases de Datos e Inteligencia Emocional

Parece, pues, clara, la necesidad de entrenar a nuestros alumnos en unas habilidades propias de la inteligencia emocional. Dado que en nuestros currículos no existen materias específicas para este tipo de aptitudes, parecería conveniente incluir su formación dentro de la metodología docente de asignaturas puramente técnicas. De un modo u otro, muchos venimos haciéndolo aunque quizá no siempre de un modo totalmente consciente, incluyendo en nuestras asignaturas, por ejemplo, prácticas en grupo, realización de trabajos con exposición oral, seminarios, etc. Nosotros hemos tratado de incorporar este tipo de formación, de un modo más directo, dentro de una asignatura de Bases de Datos (BD), cuyo contenido, a priori, podría considerarse puramente técnico. Para ello, hemos planteado las prácticas de la asignatura como un concurso, en el que el alumno tiene que superar una serie de pruebas. Cada prueba trata de fomentar un tipo de habilidades y permite obtener una puntuación. La puntuación obtenida al final del curso servirá como parte de la nota final del alumno.

En el siguiente apartado, resumimos las principales habilidades emocionales que, según Goleman, deberían fomentarse, incidiendo en las que nosotros consideramos de especial relevancia para Ingenieros Informáticos. Posteriormente, en el apartado 2.2, se detallan las características del concurso, así como el sistema de evaluación.

2.1. Habilidades emocionales deseables en un Ingeniero Informático

Goleman [9] divide el marco de la competencia emocional en dos tipos de competencias, las *personales*, que determinan el modo en que nos relacionamos con nosotros mismos, y las *sociales*, que determinan el modo en que nos relacionamos con los demás. Para el desempeño ejemplar de nuestra profesión sólo es necesario ser diestro en algunas de las competencias enumeradas (en unas 6 de un total de 25) y dependiendo de la profesión serán más necesarias unas u otras.

En cuanto a las *competencias personales*, Goleman las divide en:

- Conciencia de uno mismo: conciencia emocional, valoración adecuada de uno mismo y confianza en uno mismo.
- Autorregulación, que engloba competencias de autocontrol, confiabilidad, integridad, adaptabilidad e innovación.
- Motivación: motivación de logro, compromiso, iniciativa y optimismo.

No cabe duda de la importancia de fomentar todas estas aptitudes en los alumnos. En un primer momento para que finalicen con éxito sus estudios universitarios y también para que posteriormente desempeñen con éxito su labor profesional. Teniendo en cuenta la profesión para la que se les está formando, creemos especialmente necesario fomentar en los alumnos las capacidades de *adaptabilidad* e *innovación*, dentro de las de autorregulación, ya que durante toda su vida profesional van a tener que enfrentarse a fuertes cambios tecnológicos. Así mismo, y debido a que generalmente deberán trabajar en grupo, es importante fomentar sus aptitudes *compromiso* e *iniciativa*, dentro de las de motivación. Es igualmente importante que el alumno desarrolle una *autoestima* positiva, aprendiendo a valorarse.

Creemos que un modo de fomentar las capacidades de innovación y adaptabilidad es explicar los conceptos independientemente de los productos o de la sintaxis concreta de un determinado lenguaje. Deberán ser ellos quienes, mediante la utilización de manuales y manejo de las herramientas, aborden el aprendizaje de los productos y lenguajes concretos que habrán de utilizar. Además, se propondrán prácticas que incluyan el manejo de las características más innovadoras de los productos, de modo que sean ellos los que tengan que enfrentarse a problemas tecnológicos aún no resueltos y, en general, poco o incluso nada documentados.

En cuanto a las *competencias sociales*, Goleman las divide en:

- Empatía, entre las que se encuentran: comprensión de los demás, orientación hacia el servicio, aprovechamiento de la diversidad y conciencia política.
- Habilidades sociales: influencia, comunicación, liderazgo, catalización del

cambio, resolución de conflictos, colaboración y cooperación y habilidades de equipo.

De entre las habilidades sociales enumeradas, consideramos de especial interés fomentar la *comprensión de los demás y orientación hacia el servicio*, entre las competencias de empatía, lo que permitirá, entre otras cosas, un mayor entendimiento con el usuario, uno de los problemas claves en el desarrollo de la profesión informática. Entre las habilidades sociales son necesarias la de *colaboración* y *cooperación* así como las *habilidades de equipo*, también destacadas como prioritarias para los titulados en informática [13] y las de *comunicación* [14, 13].

Goleman destaca que las habilidades que marcan la diferencia en el campo de la programación informática *"no son estrictamente técnicas sino que tienen que ver con la capacidad de trabajar en equipo. [...] Son personas, en suma, que no compiten, sino que colaboran."*, ([9], pág. 62).

Por ello, creemos oportuno combinar, con formas individuales, diversas aproximaciones instruccionales grupales (exposición, proyectos de grupo). Mediante esas mismas actividades se fomentará el desarrollo de habilidades sociales: organización del trabajo en grupo, expresión pública de ideas, escucha activa y empática, saber preguntar y pedir ayuda, negociar e integrar ideas. Además, consideramos importante evaluar no sólo el resultado final obtenido sino también este tipo de habilidades más subjetivas, como, por ejemplo, la capacidad de trabajo en grupo.

2.2. Un experimento de educación emocional

A fin de fomentar algunas de las habilidades emocionales consideradas deseables para todo Ingeniero Informático, en la asignatura de BD de cuarto curso de Ingeniería Informática de la Universidad Rey Juan Carlos, hemos planteado, a modo de prueba, un nuevo sistema para la realización de ejercicios y prácticas. Se trata de un concurso en el que los alumnos, a través de una serie de pruebas, tanto teóricas como prácticas, irán obteniendo puntos. En función de la puntuación obtenida al finalizar el curso, y del examen, se calculará la nota final del alumno.

2.3. Características de la asignatura

Se trata de una asignatura cuatrimestral, obligatoria, de 6 créditos (3 teóricos y 3 prácticos). Es importante destacar que los alumnos han cursado previamente una asignatura de *Diseño de BD*, también de carácter obligatorio, en la que han estudiado los conceptos básicos de BD: concepto de BD y de Sistema de Gestión de BD, modelos relacional y entidad/interrelación (E/R), diseño de BD relacionales, teoría de la normalización, seguridad en BD. Igualmente, han manejado herramientas CASE para el diseño de BD relacionales.

Por tanto, el temario de esta asignatura comprende conceptos avanzados de modelos de BD: modelo objeto-relacional (OR), incluyendo BD activas y diseño orientado a objetos (OO) de BDOR y relacionales con UML (Unified Modeling Language), [2, 3, 4, 6, 11, 12, 15, 16]. Los alumnos trabajan con Oracle, utilizando siempre las últimas versiones del producto (el curso 2002/03, en el que se ha realizado este experimento, con Oracle 9*i*). Dada la rapidez con que avanza la tecnología de BD, el objetivo principal de esta asignatura no es tanto que los alumnos aprendan un modelo concreto de BD, sino más bien que aprendan a manejar conceptos nuevos y nueva tecnología. Que sean capaces de razonar y diseñar en diferentes modelos, potenciando en el alumno la capacidad de pensar a un mayor nivel de abstracción lo que le facilitará, en el futuro, la adaptación a los nuevos modelos y tecnologías que, con toda seguridad, surgirán.

Además, las prácticas no serán tan guiadas y deterministas como lo son en cursos anteriores, sino que se les pedirá que se enfrenten a problemas no resueltos en clase (ni en libros de texto o bibliografía básica de la asignatura). De este modo, se verán obligados a probar la sintaxis y buscar las soluciones en manuales, la Web, etc.

2.4. Dinámica del concurso

Aquellos alumnos que deciden participar se agrupan en equipos de 6 o 7 personas. La participación en el concurso es libre y a decisión de cada alumno. Aquellos alumnos que no entren en concurso, se presentarán al examen sin ningún punto acumulado y su nota dependerá exclusivamente de su examen, que puntuará del modo habitual (entre 0 y 10 puntos).

Se realizarán una serie de pruebas, que se detallan a continuación. Cada prueba tendrá un valor. En función de las puntuaciones acumuladas, cada equipo tendrá un resultado final que se utilizará, junto con la nota obtenida en el examen, para la calificación final del alumno. Es decir,

NOTA FINAL = PC + NE*(10-PC)/10

Siendo NE la nota del examen individual y PC los puntos obtenidos en el concurso. Estos puntos estarán entre 0 y 5 (el grupo ganador 5 puntos y el resto una puntuación proporcional a su resultado).

2.5. Descripción de las pruebas

La siguiente tabla muestra un resumen de las principales pruebas realizadas, incluyendo: descripción de las mismas, sistema de puntuación y principales objetivos emocionales perseguidos con la realización de las mismas. Son objetivos generales del concurso y, por tanto, de todas las pruebas:

- Entrenar al alumno en el trabajo en equipo
- Entrenar al alumno en capacidades como la adaptación al cambio
- Entrenar al alumno en el diseño de sistemas basado en modelos (independientemente de los modelos concretos utilizados).

Prueba	**Funcionamiento**	**Puntuación**	**Objetivos**
P1. Teoría: BDOO-Objetos-BDOR	Se enunciarán una serie de preguntas. Se dará un tiempo de 45 segundos para contestar a cada una de ellas, por escrito.	Grupo 1: 3 ptos Grupo 2: 2 ptos Grupo 3: 1 pto	• Comprobar la comprensión de los temas teóricos por parte de los alumnos. • Consolidar conceptos Descubrir lagunas y fallos en las explicaciones.
P2. Teoría: Modelado Conceptual OO-Modelo OR-Activas	Se enunciarán una serie de preguntas. El grupo que primero conteste suma dos puntos. Si se equivoca resta 1 punto.	Grupo 1: 5 ptos Grupo 2: 3 ptos Grupo 3: 1 pto	Además de los objetivos de la prueba anterior: • Potenciar la agilidad y compenetración del grupo.
P3. Ejercicios: Modelado conceptual UML	Se entrega un enunciado. Cada grupo debe entregar una solución de diseño en UML en un máximo de 30 min. Solo puntúan los tres primeros grupos que entreguen una solución correcta. Si se entrega una solución que no es válida, este grupo queda eliminado.	Grupo 1: 5 ptos Grupo 2: 3 ptos Grupo 3: 1 pto	• Comprobar la comprensión de UML y el modelado conceptual OO por parte de los alumnos. • Entrenar al alumno en el diseño de alto nivel OO.
P4. Exposición: BD/WEB/XML	Cada grupo preparará una exposición de 5 minutos sobre un tema propuesto por el profesor (en este caso, se propuso cualquier tema relacionado con BD, Web y/o XML – *Extensible Markup Language*-). El grupo decide el representante para exponerlo. La exposición debe basarse en, al menos, la consulta de un LIBRO. La puntuación se somete a votación de los alumnos y del profesor. Se dan dos calificaciones, una de alumnos y otra de profesor a fin de corregir desviaciones producidas por afinidades entre alumnos..	Alumnos: Grupo 1: 4 ptos Grupo 2: 3 ptos Grupo 3: 2 ptos Grupo 4: 1 pto Profesor: Grupo 1: 4 ptos Grupo 2: 3 ptos Grupo 3: 2 ptos Grupo 4: 1 pto Todos los que presenten: 1 pto	• Entrenar al alumno en búsqueda y consulta de documentación • Entrenar al alumno en capacidad crítica, al tener que seleccionar la bibliografía y enfocar el tema. • Entrenar al alumno en capacidad de síntesis (exponen 5 minutos). • Acostumbrar al alumno a hablar en público. • Entrenar al alumno en el espíritu crítico al tener que juzgar el trabajo de otros compañeros; los fallos de sus compañeros les servirán de espejo. Ser juzgados por compañeros estimula su interés.
P5. Prácticas Práctica 1: Tipos de objetos Práctica 4: Disparadores	Se entrega un enunciado. Cada grupo tiene una única oportunidad de entrega. Habrá un tiempo máximo. Puntúan todos los grupos que entreguen una solución correcta.	Grupo 1: 4 ptos Grupo 2: 3 ptos Grupo 3: 2 ptos Resto: 1 pto Grupo con solución incorrecta: -1	Además de los objetivos de aprendizaje de los conceptos propios del contenido de la práctica: • Acostumbrar a enfrentarse a un producto que no conoce. • Acostumbrar a buscar y consultar documentación para resolver problemas sintaxis y programación.

P6. Prácticas Práctica 2: Tipos REF Práctica 3: Tipos colección	Se entrega un enunciado. Cada grupo tiene una única oportunidad de entrega. Habrá un tiempo máximo. Solo puntúan los tres primeros grupos que entreguen una solución correcta.	Grupo 1: 5 ptos Grupo 2: 3 ptos Grupo 3: 1 pto	Además de objetivos de aprendizaje de los conceptos propios del contenido de la práctica y de los dos objetivos de la práctica anterior: • Potenciar la agilidad y compenetración del grupo (reparto de tareas, etc.)
Ejercicio completo	**Los alumnos deberán llevar a cabo un ejercicio en el que se resuma e integre el temario. Se realizará en varias pruebas que se detallan a continuación.**		**• Repasar, consolidar e integrar conceptos** **• Entrenar al alumno en el diseño e implementación de una BDOR, desde la etapa de captura de requisitos.**
P7. Requisitos	Los alumnos deberán redactar una especificación de requisitos a partir de los requisitos de usuario (el profesor actuará de usuario). Tendrán toda la clase para realizar esta prueba (2h.) Se entregarán las especificaciones y sólo la mejor puntuará (en caso de empate pueden puntuar varias).	Grupo 1: 3 ptos	• Acostumbrar al alumno a enfrentarse con un problema no delimitado. • Entrenar al alumno en el trato con el usuario.
P8. Modelado conceptual	A partir de la especificación ganadora de la prueba anterior, deberán realizar un modelo conceptual en UML. Para ello se dejarán 45 minutos para que cada grupo obtenga una solución. Cada grupo se partirá en dos subgrupos y se mezclarán entre sí, de forma que cada uno de estos grupos estará compuesto por integrantes de dos de los grupos iniciales. Los nuevos grupos deberán consensuar una solución a partir de las dos propuestas que aporten. Para esto tendrán un tiempo máximo de 45 minutos. Se establecerán tres franjas de calificación y cada solución caerá en una de ellas.	SG- SubGrupo • Los 2 SG caen en Franja 1: 5 ptos • 1SG. en Franja 1 y 1SG. en Franja 2: 3 ptos • 2 SG en franja 2: 1 pto	Además de las propias del modelado conceptual: • Fomentar el trabajo de TODO el grupo. • Potenciar las habilidades de negociación así como la capacidad de consensuar soluciones. • Acostumbrar al alumno a defender sus posiciones frente a otras contrarias, así como a ser capaz de adoptar otras mejores si éstas le convencen.
P9-P10: Diseño e Implementación OR y relacional	A partir del modelo conceptual mejor, tomado de la prueba anterior, se realizará el diseño OR en UML para Oracle y su compilación. Se establecerán franjas temporales de 30 minutos. Cada grupo tiene una sólo posibilidad de entrega y sólo puntúan las que sean correctas.	• G en Franja 1: 3 ptos • G Franja 2: 2 ptos • G en Franja 3: 1 pto	Además de las propias del diseño lógico e implementación: • Adquirir agilidad y destreza en el diseño OR y en el manejo de un producto OR.
Comparativa	Se redactará una comparativa entre la solución relacional y la OR, indicando ventajas e inconvenientes de una y de otra así como la oportunidad de una u otra solución o de realizar un diseño mixto.	La mejor (o mejores) comparativa: 5 ptos	• Entrenar al alumno en la capacidad crítica y de toma de decisiones de diseño y tecnología.

3. Resultados y lecciones aprendidas

La experiencia se puso en marcha y, a medida que transcurría el curso, se iba refinando. Para ello, nos basamos en las impresiones del profesor, así como en el diálogo, comentarios y sugerencias de los alumnos.

Además, para obtener una valoración más completa, al finalizar el curso se pasó una encuesta a los alumnos. Esta encuesta era anónima y de carácter voluntario. Se trataba de conocer, según la impresión del alumno, en qué medida había sido positivo este sistema de cara al aprendizaje, así como de mejorarlo según los comentarios y sugerencias. Por supuesto que esta encuesta serviría fundamentalmente para completar la impresión del profesor de la asignatura.

La encuesta constaba de 12 preguntas, en la que se pedía a los alumnos su opinión acerca de cómo ha afectado en el desarrollo de la asignatura el nuevo sistema de prácticas en aspectos tales como la asistencia y participación en clase, el seguimiento continuo y la mejor comprensión de la asignatura, la realización de prácticas y ejercicios y la agilidad en resolverlos, la mejor comprensión del temario, el interés general por la asignatura, el trabajo en equipo, la búsqueda de documentación y el enfrentarse con nuevas tecnologías y productos, etc.

Los alumnos debían responder en una escala de 7 a 1 (de más a menos positivo). A grandes rasgos, y dividiendo las puntuaciones en 7, 6, 5 (el efecto ha sido positivo) 4 (no ha afectado), y 3, 2, 1 (poco o nada positivo), el resultado general ha sido bastante satisfactorio. Casi todos los aspectos han sido considerados positivos. Los aspectos mejor valorados, muy por encima del 4, han sido, en este orden, los relativos a que el sistema de prácticas ha hecho que la asignatura sea más *amena*, que ha contribuido a incentivar la *realización de prácticas y ejercicios* de clase y que ha contribuido a incentivar y mejorar el *trabajo en equipo*. También han estado entre los aspectos más valorados los relativos a una mayor participación en clase en todos los aspectos (asistencia, participación en las clases y seguimiento de la asignatura). La Figura 1 muestra la valoración general del método.

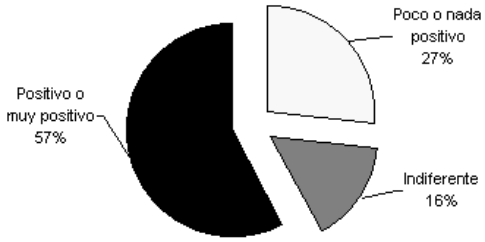


Figura 1. Valoración general del método

Según el profesor, aumentó la participación de los alumnos en clase de un modo muy significativo, haciendo, además, que éstas fueran mucho más amenas y dinámicas. Quizá éste es el beneficio más destacable de la experiencia.

La asistencia a clase fue más homogénea a lo largo del curso. Hubo un incremento importante en el número de alumnos que hacían los ejercicios y prácticas propuestas en clase: en cursos anteriores, cuando al alumno se le proponía un ejercicio en clase, el profesor percibía que era un grupo muy reducido de ellos los que realmente hacían el esfuerzo de pensarlo; sin embargo, al premiar la agilidad, la mayor parte de los alumnos se esforzaban en pensar soluciones. Esto también hizo que, a pesar de la sensación percibida por algunos alumnos de que se perdía tiempo, se realizaran más ejercicios que en años anteriores.

El hecho de que algunos alumnos del grupo no trabajaran, beneficiándose del esfuerzo del resto, es un inconveniente que no tiene solución y que ocurre siempre que realicemos trabajos en grupo (independientemente de que se haga o no dentro de un juego). Es también un aprendizaje de algo que ocurrirá en la vida real. El factor corrector de este tipo de problema es la nota del examen, que es individual.

Otro aspecto, criticado por algunos alumnos, es el carácter competitivo de las pruebas. Estimamos que la competitividad en cierto grado es, y así lo hemos notado, positiva. Estimula al alumno a trabajar y esforzarse al tener que quedar bien delante de sus compañeros y así también lo han hecho notar algunos alumnos. La competitividad, además, era entre grupos, y no individual, lo cual contribuye a mejorar el trabajo en equipo, que era uno de los objetivos perseguidos. Sin embargo, es posible que el hecho de haber sólo un grupo ganador incite a una competitividad excesiva. Estimamos que este problema puede resolverse fácilmente: se trataría de que los grupos acumularan puntos y establecer franjas no excluyentes (como ya se hizo en las últimas pruebas). De este modo, todos los grupos que estuvieran en una determinada franja, sumarían los mismos puntos de cara al examen.

En cuanto a la crítica expresada por algunos alumnos de premiar la rapidez, también creemos que es bueno mantener el factor agilidad, si bien es posible reducir su peso en las pruebas. Por ejemplo, incluyéndolo sólo en alguna de ellas o bien puntuando también por franjas.

Otro aspecto que consideramos positivo es que el sistema permite realizar una evaluación continuada en clases con un número elevado de alumnos (hablamos de en torno a 100 alumnos).

4. Conclusiones

En este artículo se ha resumido un experimento llevado a cabo en la asignatura de BD de

Ingeniería Informática de la Universidad Rey Juan Carlos. El experimento trataba de fomentar en el alumno ciertas habilidades emocionales, además de las intelectuales propias de la materia. Para ello, los ejercicios y prácticas de la asignatura se han realizado mediante un concurso que permitía a los alumnos acumular puntos para el examen.

Las principales ventajas encontradas en este sistema son: se fomenta la participación del alumno y un seguimiento continuado de la asignatura; permite, de algún modo, una evaluación a lo largo del curso, algo que no se obtiene con el sistema de examen; ha fomentado la asistencia a clase, así como el trabajo en equipo.

Sin embargo, a través de una encuesta a los alumnos, se han detectado algunos aspectos que sería necesario ajustar, entre los que cabe destacar la excesiva competitividad con la que se ha planteado el concurso, así como el peso que se le ha dado al factor de agilidad.

El próximo curso, trataremos de aplicar el experimento introduciendo las modificaciones expuestas en el apartado 3 para eliminar estos problemas.

Agradecimientos

Queremos agradecer su colaboración a los alumnos de Bases de Datos de cuarto curso de Ingeniería Informática de la Universidad Rey Juan Carlos del curso 2002/2003. Este trabajo ha sido parcialmente financiado por el MCYT (TIC 2002-04050-C02-01).

Referencias

[1] ACM (2001) Computing curricula 2001. Computer Science (final report) ACM-IEEE

[2] Atzeni, P., Ceri, S., Paraboschi S. and Torlone, R., 1999. Database Systems. Concepts, Languages and Architectures. McGraw-Hill.

[3] Bertino, E. and Marcos, E., 2000. Object Oriented Database Systems. En Advanced Databases: Technology and Design, O. Díaz and M. Piattini (Eds.). Artech House.

[4] Booch, G., Rumbaugh, J., Jacobson, I., 1999. The Unified Modeling Language User Guide. Addison Wesley.

[5] Boyatszis, R.E., Cowen, S.S., Kolb, D.A.1995 Innovations in Professional Education: Steps on a Journey from Teaching to Learning. Jossey-Bass

[6] Eisenberg A. y Melton J., SQL:1999, formerly known as SQL3. ACM SIGMOD Record, Vol. 28, No. 1, pp. 131-138, Marzo, 1999.

[7] European Centre for the Development of Vocational Training (Cedefop) and ICEL Career Space, 2002. "Generic ICT skills profiles. Future skills for tomorrow's world" http://europa.eu.int

[8] Goleman, D. Inteligencia Emocional. Kairós, 1996

[9] Goleman, D. La práctica de la Inteligencia Emocional. Kairós, 1999

[10] Informatics Curriculum Framework 2000 (ICF-2000), Technical Committee 3 IFIP-UNESCO

[11] Marcos E., Vela B. y Cavero J. M., Extending UML for Object-Relational Database Design. UML 2001, Toronto, LNCS 2185, Springer Verlag, pp. 225-239, 2001.

[12] Marcos, E. , Vela, B. y Cavero J.M. A Methodological Approach for Object-Relational Database Design using UML. Journal on Software and System Modeling (SoSyM). Vol. 2. Issue 1. Ed. Springer Verlag. Editores B. Rumpe y R. France. ISSN: 1619-1366, Marzo 2003.

[13] McDonald, M. y McDonald, G., "Computer Science Curriculum Assessment". Thirtieth SIGCSE Technical Symposium on Computer Sciece Education. New Orleans, 24-28 de marzo, 1999. En SIGCSE Bulletin, Vol. 31, Num. 1, marzo, 1999, pp. 194-197.

[14] Norris, C. y Wilkes, J. (1999), "Computer Systems "Conference" for Teaching Communications Skills". Proceedings of the Thirtieth SIGCSE Technical Symposium on Computer Sciece Education. New Orleans, 24-28 de marzo, 1999. En SIGCSE Bulletin, Vol. 31, Num. 1, marzo, 1999, pp. 189-193.

[15] Silberschatz, A., Korth, H. F., Sudarshan, S. 2002. Fundamentos de Bases de Datos. Mcgraw Hill

[16] Stonebraker and Brown, 1999. Object-Relational DBMSs. Traking the Next Great Wave. Morgan Kauffman.

[17] The Bologna Declaration on the European space for higher education: and explanation. http://europa.eu.int

[18] Weisinger, H. 2001 La Inteligencia Emocional en el trabajo. Punto de lectura

Retos en la formación de profesionales de las tecnologías de la información: perfiles y aplicación en la formación universitaria

Luis Fernández Sanz
Departamento de Programación e Ingeniería del Software
Universidad Europea de Madrid
28670 Villaviciosa de Odón (Madrid)
e-mail: lufern@dpris.esi.uem.es

Resumen

Actualmente, una de las quejas habituales de las empresas y círculos profesionales de cualquier disciplina sobre la labor de la universidad se basa en el divorcio entre la formación impartida y las necesidades reales del mercado laboral. Este trabajo muestra datos de los diversos estudios de requisitos para el empleo en tecnologías de la información y de las comunicaciones (TIC) realizados en la Universidad Europea de Madrid (UEM). También aporta algunas reflexiones sobre la aplicación de dichos datos a la formación universitaria reglada en informática.

1. Introducción

Son muchas las palabras vertidas desde universidades y empresas sobre la necesidad de adaptar la formación de los estudiantes de grado a la realidad de las necesidades laborales de las empresas y organizaciones donde se ubicará su futuro laboral. Lamentablemente, en la mayoría de las ocasiones, las palabras no han dado lugar a acciones serias encaminadas a mejorar la conexión entre la formación universitaria y la realidad del empleo. Esta situación resulta aún más compleja en el caso de las TIC, especialmente en las titulaciones de informática donde se acumulan varios factores de influencia:

- La rápida evolución tecnológica en un campo que dista de ser maduro supone gran dificultad para elaborar planes de estudio con vigencia a largo plazo. También provoca la creación de perfiles profesionales que aparecen (y, a veces, desaparecen) a velocidad vertiginosa.
- La innovación tecnológica que domina el mercado informática, no se genera predominantemente en la investigación universitaria sino en las empresas (a diferencia de muchos años atrás[1]). Además, éstas fuerzan estrategias de competición en el mercado que provocan obsolescencia prematura[2] de versiones, técnicas, estándares, etc. que dificulta la enseñanza reglada.
- No existen estudios fiables y detallados disponibles para la comunidad universitaria sobre los requisitos reales exigidos a los candidatos a puestos de trabajo de TIC. Con frecuencia, estamos acostumbrados a fiarnos de "gurús" sin datos rigurosos y detallados.

Además, existen actitudes poco positivas en ambos lados de la confrontación dialéctica sobre la formación reglada universitaria en informática. De una parte, no son pocos los profesores universitarios en informática que consideran que no tienen obligación de enseñar conocimientos prácticos o de estar al tanto de lo que se requiere en la profesión, que el contacto con el mundo empresarial resulta perjudicial para un profesor universitario o que la verdadera tarea es la investigación mientras que la docencia es una "carga". En especial, ésta última afirmación, soportada por el baremo habitual para la promoción de los profesores en las universidades públicas[3], resulta muy dañina para el ánimo de averiguar qué es aquello que se demanda en el mercado laboral. La preocupación por obtener los méritos investigadores necesarios suele dejar poco margen de esfuerzo para la dedicación a la tarea educativa (denominada "carga" docente). Por

[1] Recordemos la contribución universitaria en la evolución de UNIX.
[2] En comparación con lo que ocurre en otras disciplinas.
[3] De manera tímida, la LOU ha incluido más méritos docentes en la acreditación de contratados frente a la tendencia tradicional de evaluación de los docentes.

último, es sabido, lamentablemente, cómo la creación de nuevos planes de estudios está muy influida por consideraciones de reparto político de cuotas de docencia a los actores influyentes ya existentes. También es cierto que, en la mayoría de los casos, este diseño se realiza sin la colaboración de estudios detallados de prospectiva y de análisis de situación que vayan más allá del cumplimiento de requisitos de normativa legal presente o futura.

En el otro lado, las empresas desearían que los titulados de informática tuvieran los conocimientos exactos para ser productivos al 100%, inmediatamente, en la configuración informática exacta que tienen instalada (a pesar de lo absurda que pueda ser en ciertas ocasiones). Dado que muchas empresas están atenazadas por la presión de plazos de tiempo poco realistas y de una permanente carencia de recursos[4] humanos o materiales y por las decisiones poco acertadas de muchos gestores, los métodos aplicados distan de parecerse a las buenas prácticas que aseguran calidad en los productos y servicios (véase, por ejemplo, [1]). Estas prácticas son las que, inicialmente, deberían ser objeto de enseñanza para los estudiantes de grado[5]. Por último, el ámbito laboral de las TIC ha contado con una definición muy difusa de perfiles (véase, por ejemplo, la figura 1), acrecentada por la existencia de profesionales con las más variopintas titulaciones, reconvertidos urgentemente a una actividad con creación intensiva de empleo. Esta situación hace que las titulaciones de informática no sean bien identificadas como la formación más adecuada para el ámbito laboral de las TIC[6].

Otro factor influyente es la consideración de la actividad de formación en general, que no cuenta con la mayor de las estimas tanto en el seno de las empresas como en la sociedad en general. Esta circunstancia se une al síndrome del seleccionador nacional de fútbol. Al igual que todos los habitantes de un país se consideran capacitados para ser los mejores entrenadores de la selección nacional, así prácticamente cualquier ciudadano/a no tiene reparo alguno en opinar temerariamente sobre la labor educativa universitaria, indicando que él/ella sería capaz organizar una formación impecable que capacite extraordinariamente a los alumnos, con un esfuerzo mínimo, para el mejor desempeño profesional. Conozco varias personas de empresa que han entrado como profesores a tiempo completo en una universidad y que se han sorprendido del trabajo que supone una docencia de una calidad suficiente.


IMPORTANTE EMPRESA NECESITA

INFORMÁTICO/ALMACENERO

PENSAMOS EN UN JOVEN QUE TENGA AMPLIOS CONOCIMIENTOS DE INFORMÁTICA Y QUE COMPATIBILICE LA JORNADA DE TRABAJO EN EL ORDENADOR (75%) Y EN ALMACÉN (25%).

LAS CONDICIONES SERÁN:

CONTRATO LABORAL, SEG. SOCIAL, JORNADA PARTIDA, INCORPORACIÓN INMEDIATA.

[redacted]

Figura 1. Oferta real de trabajo muy ilustrativa de la figura de "informático"

Otro ejemplo de esta disposición es la conocida creencia de que las asignaturas básicas como las matemáticas resultan innecesarias para el ejercicio profesional y, por lo tanto, deben ser excluidas de los planes de estudio a favor de conocimientos más prácticos. Este argumento se puede encontrar en foros de discusión sobre formación tecnológica, en conversaciones informales e, incluso, en artículos.

Un ultimo factor de explicación que cabe reseñar es la confusión y lucha entre las posiciones de los partidarios de la *Computer Science* y los de la ingeniería informática (*Informatics*). En un contexto como estas jornadas no es lógico incidir en la explicación de ambas concepciones. Sólo parece razonable señalar que la Ciencia Informática, por su propia filosofía próxima a la ciencia básica, no contempla el acercamiento a las aplicaciones prácticas o empresariales de las TIC.

[4] Por la constante tendencia de los gestores a recortar gastos manteniendo la misma productividad.

[5] No parece lógico enseñar a "hacer mal las cosas" aunque sí es recomendable fomentar una adecuada valoración entre costes y riesgos.

[6] De hecho, ciertas titulaciones, con la ventaja legal de un colegio profesional, han contribuido a esta confusión en beneficio propio.

Por el contrario, la preocupación por soluciones eficientes y útiles en la sociedad, es inherente a la idea de ingeniería.

2. Motivación

Dada la situación de desencuentro entre Universidad y Empresa descrita en el anterior apartado, en esta comunicación dirigida al profesorado de informática[7] reunido en JENUI resulta conveniente abordar los estudios para el mejor conocimiento del mercado laboral en TIC elaborados en el seno de la UEM desde 1998[8]. El propósito es presentar los datos obtenidos sobre requisitos técnicos y no técnicos exigidos en las ofertas de empleo publicadas para ayudar a todos los educadores en informática en el diseño docente de los estudios. También se ha obtenido información directamente de entrevistas con responsables de empresas emblemáticas, especialmente sobre competencias profesionales.

Por otra parte, presentaremos las consecuencias reales que ha tenido la utilización de estos datos en la práctica docente. También se incidirá en las dificultades que se han encontrado a la hora de implementar las acciones y las soluciones que pueden existir en cada caso.

3. Los estudios RENTIC y ERES

Los estudios RENTIC surgen de la constatación de que el empleo es la preocupación más importante de la población española en general (como se ha señalado en diversas estudios: por ejemplo, los antiguos CIRES) y, por supuesto, de los alumnos que cursan carreras universitarias. Los estudiantes universitarios y sus familias buscan preferentemente que los estudios elegidos, en su mayoría (73,3%) vocacionalmente por los alumnos [2], se desarrollen de tal forma que obtengan las mayores posibilidades de lograr el mejor empleo cualificado.

En España es complicado contar con fuentes de información fiables sobre la actividad en TIC como los propios especialistas en estadística reconocen debido a la rápida evolución del sector y los problemas para delimitar el ámbito de estudio [3]. Dentro del ámbito de las TIC, distintas instituciones han abordado la elaboración de estudios sobre el sector, su evolución comercial y económica o, incluso, sobre distintos aspectos del mercado laboral relacionado con estas tecnologías. Desde el punto de vista laboral, podemos ofrecer el siguiente resumen de las principales aportaciones[9]

3.1. Informes de empleo de ámbito general que ofrecen datos separados de TIC.

Un caso típico es el informe INFOEMPLEO [4] donde se analizan de manera general un grandísimo número de ofertas[10] para obtener una panorámica laboral genérica en España. Aquí se pueden obtener el número de ofertas procedentes de empresas del sector de TIC, el de las dirigidas al área funcional de informática en cualquier tipo de empresa, las cifras de ofertas para cada puesto de trabajo (incluidos los de informática), las dirigidas hacia cada titulación oficial de TIC, etc. No se analizan en detalle los requisitos específicos incluidos en cada oferta laboral. En otros casos, los estudios son de gran entidad (por el ámbito geográfico y temporal abordado) pero hacen pocas distinciones sobre el conjunto del empleo en TIC. Uno de los más comentados fue el informe de IDC [5] sobre perspectivas de empleo en TIC en Europa hasta 2004, emitido en plena euforia tecnológica, donde se presentaban grandes números en cuanto a déficit de profesionales cualificados[11]. También el informe del WEF [6] siguió una línea análoga.

[7] La información a profesionales de TIC que también pueden estar interesados en los datos de estudios (para mantener su empleabilidad) se ha cubierto con la próxima publicación del artículo "Requisitos para el empleo en nuevas tecnologías de la información: el estudio RENTIC" en la revista Novática de ATI.

[8] Existe una versión anterior con datos de 1993-95 realizada cuando el autor pertenecía a la Universidad Politécnica de Madrid.

[9] Dado que es muy complejo resumir el panorama completo de estudios, el autor pide disculpas anticipadamente por las omisiones que se detecten.

[10] El último informe del período 2001-02 incluía el análisis de 159.037 ofertas.

[11] Déficit de 4.000.000 profesionales en el mundo en 2004, 1,7 millones en Europa y 100.000 en España.

3.2. Informes específicos del sector de TIC.

La mayoría de los estudios realizados suelen abordar aspectos de utilidad práctica para los empleadores. Así, encontramos estudios salariales entre los que podemos mencionar los de periodicidad regular (por ejemplo, el de Oberthur) o puntuales (por ejemplo, el de SEDISI del año 2000 [7]). Entre estos informes, podemos encontrar algunos estudios puntuales, ligados a proyectos subvencionados sobre nuevos yacimientos de empleo, que analizan en detalle requisitos de ofertas [8]. Algunos de ellos tiene un cierto componente justificativo para ciertas acciones de formación o empleo como el proyecto PAFET[12], cuya primera edición [9] aportó una clasificación de puestos y requisitos para cada uno de ellos mientras que la segunda [10] se ha centrado más en los retos formativos asociados al sector electrónico más que de TIC[13]. En otras casos, existe un interés muy concreto ligado a reivindicaciones de titulación (por ejemplo, el estudio de ALI de 2001 sobre titulación solicitada en las ofertas publicadas en prensa). En este ámbito, los simples ejemplos de catalogación de perfiles sin datos contrastados no resultan útiles (como el del proyecto Career-Space en www.career-space.com o el más antiguo de CEPIS en http://www.cepis.org/prof/eiss.htm).

Dentro de este panorama, el estudio RENTIC, que desde 1998 elabora la UEM, aporta más datos sobre el empleo en TIC:

- Las ofertas no se limitan a una asociación patronal concreta (gran parte del empleo tecnológico está en empresas que simplemente usan las TIC). Aunque usa ofertas publicadas en prensa[14], los controles informales realizados confirman su representatividad.
- El número de ofertas analizado es más pequeño que el de los grandes estudios generales (miles de ofertas) pero se estudian con muchísimo más detalle, hasta llegar a los últimos requisitos expresados en cada una. Además se aporta la comparativa histórica de los resultados respecto de las anteriores ediciones, consiguiéndose una interesante perspectiva de evolución del mercado laboral.
- El objetivo principal es que tanto candidatos para el empleo como formadores y entidades educativas conozcan la evolución de los requisitos para el empleo en TIC. No obstante, empleadores y poderes públicos pueden obtener información útil sobre coyuntura laboral (o incluso de mercado) relacionada con las TIC.

Por otra parte, la UEM emprendió un estudio detallado (ERES) sobre las competencias personales no técnicas más valoradas por las empresas para los candidatos a un empleo. Dicho estudio fue realizado en 2001 por el Instituto PsicoPedagógico de la Universidad y en él intervinieron más de 60 empresas para determinar las 10 competencias más apreciadas. El objetivo fue sentar las bases para el plan de formación en competencias para los alumnos de la universidad.

4. Resultados de la 4ª edición de RENTIC

El estudio presenta los resultados obtenidos del análisis de las ofertas de empleo publicadas en los diarios de difusión nacional y que se orientan a cubrir puestos de trabajo destinados a especialistas en las TIC. Dicho análisis se ha centrado en el estudio de los conocimientos específicos de TIC solicitados en las ofertas de empleo así como en algunos aspectos complementarios exigidos a los candidatos en cuanto a idiomas, titulación y otros requisitos personales, especialmente competencias de comportamiento. También incluye datos de incentivos ofertados en las ofertas para atraer candidatos a procesos de selección y, por primera vez en esta edición, los puestos ofertados.

Otro de los atractivos del estudio es la posibilidad de observar la evolución de los requisitos solicitados en las ofertas de empleo para profesionales de TIC. El estudio se ha realizado sobre la totalidad de ofertas (249) publicadas entre

[12] Aunque también sorprende por una curiosa manera de presentar que una gran parte de los puestos en TIC están destinados a ser ocupados por ingenieros de telecomunicación, donde parece observarse la influencia del Colegio Oficial de Ingenieros de Telecomunicación, lo que fue objeto de crítica por ALI en un comunicado.

[13] Ya que se apoya en la experiencia recogida a través de ANIEL, patronal de electrónica y comunicaciones.

[14] En la 5ª edición, se incluirán datos de ofertas publicadas electrónicamente en portales de empleo.

el 7 de octubre de 2001 y el 22 de septiembre de 2002, con 461 perfiles profesionales diferenciados. Son ofertas publicadas en ediciones dominicales de los diarios El País y/o ABC.

Debido a la dificultad de comprobar el número real de personas que finalmente se contratan en cada oferta, se ha tomado el perfil (un puesto caracterizado individualmente en una oferta) como unidad de referencia para las estadísticas. Consideramos que el número de perfiles que solicitan un determinado requisito (por ejemplo, Java) es buen indicador de las oportunidades que un candidato tiene de optar a un empleo en TIC.

Tipo de empresa	Nº Perfiles 2001-2	%Perfiles 2001-2
Servicios informáticos	198	42,95%
Informática	45	9,76%
Entidad financiera	6	1,30%
Industrial	20	4,34%
No figura	21	4,56%
Formación	13	2,82%
Transporte	4	0,87%
Tecnología (aeroespacial, electrónica,...)	16	3,47%
Comunicaciones	29	6,29%
Administración pública	24	5,21%
Distribución/comercio	7	1,52%
Turismo/ocio	20	4,34%
Servicios/consultoría no informáticos	23	4,99%
Seguros	1	0,22%
Publicidad	2	0,43%
Editoriales/prensa/información (TV...)	9	1,95%
Internet	0	0,00%
Construcción/ingeniería	10	2,17%
Salud, farmacia	10	2,17%

Tabla 1. Distribución por tipo de solicitante

Las ofertas incluidas en el estudio corresponden a todo tipo de puestos que puedan desempeñar los titulados universitarios, superiores o medios, en informática, aunque no siempre se exija este grado académico en la oferta. Sólo se han excluido los puestos relacionados en exclusiva con la función comercial y/o de ventas, por sus especiales condiciones laborales. Tampoco se han considerado ofertas de empleo en prácticas, becas y aquéllas cuya única exigencia consiste en ser recién titulado (sin ninguna otra condición) por no aportar valor al objetivo del estudio.

Las ofertas estudiadas corresponden a todo tipo de empresas y organizaciones ya que se trata de recogerla mayor amplitud posible de destinos profesionales para los titulados en informática. Evidentemente, la mayoría de las ofertas se ubican entre las empresas de servicios informáticos y, a bastante distancia, las empresas de fabricación informática (hardware o software). De hecho, la distribución se puede ver en la Tabla 1.

En cuanto a puestos ofertados, que se incluyen por primera vez en esta edición, tienen la siguiente distribución de categorías y áreas (ver Tabla 2):

- Desarrolladores básicos (programadores, desarrolladores, etc.): 18,22%.
- Desarrolladores de nivel medio (analista programador, etc.): 14,32%.
- Desarrolladores de nivel alto (analistas, ingenieros, etc.): 16,92%.
- Jefes de proyecto: 5,42%.
- Consultores de todos los niveles: 15,62%
- Personal básico de sistemas (operadores, técnicos de soporte, etc.): 13,85%.
- Personal de sistemas de nivel no básico (administradores, ingenieros, etc.): 8,68%.
- Especialistas (todas las áreas): 8,46%.
- Directivos de todas las áreas: 8,68%.
- Puesto descrito por la titulación solicitada (diplomado, titulado, etc.): 3,47%.
- Puesto inespecífico (área funcional o nombre genérico: experto, informático, etc.): 3,47%.
- Formadores/profesores: 3,47%.

Área	%
Desarrolladores	44,0%
Técnicos	16,4%
Consultores	12,6%
Sistemas	8,8%
Dirección	6,8%
Otros	5,3%
Formación	3,2%
Especialistas	2,3%

Tabla 2. Distribución por áreas de actividad

Las categorías más solicitadas son las siguientes:

- Programador: 16,27%.
- Analista-programador: 13,45%.
- Analista: 14,97%.
- Técnico de sistemas: 7,81%.
- Jefe de proyecto: 5,42%.

4.1. Requisitos de titulación y formación

En el estudio, para cada perfil profesional solicitado en oferta de empleo, se contabiliza cada una de las titulaciones admitidas. Puede apreciarse cómo es habitual que no exista mención titulación alguna (cerca de la mitad: 54,63% en el período 2000-1 y 46,9% en el de 2001-2). Los datos obtenidos indican que la demanda específica de titulados superiores de informática ha crecido hasta el 18,4% en 2001-2 desde el 15,5% anterior. Si incluimos las variantes de titulación equivalentes (por ejemplo, ciencia físicas o matemáticas con especialidad de computación[15]), la cifra se eleva al 26% (si bien hay una disminución de 0,5% en las peticiones de estas variantes). Los titulados medios de informática pueden hacer valer específicamente su titulación en el 12,23%. Los otros requisitos de acreditación se refieren a certificados de fabricantes (Microsoft, Cisco, Oracle, etc.) pero en ningún caso llegan a pasar de un 2,5 %. La petición de titulación superior o media genérica se asocia a puestos muy específicos mientras que las de MBA se centra en ámbitos de responsabilidad directiva.

El conocimiento de idiomas refleja un abrumador dominio del requisito de inglés en distintos grados: aproximadamente un 34% indica expresamente esta necesidad (con un 17,5% de petición de dominio del idioma hablado y escrito). Sin embargo, resulta significativo que en muchas ofertas (un 62% más o menos) no se exija ningún nivel idiomático. No aparece nunca el requisito de títulos de acreditación de nivel idiomático. Los conocimientos de alemán y francés aparecen muy ocasionalmente en las ofertas (menos del 2,5%).

4.2. Competencias y situación personal

En los últimos años la evaluación de competencias personales ha tomado una gran importancia en el ámbito de la selección y gestión de recursos humanos[16]. Esta tendencia se ha trasladado a la redacción de las ofertas de empleo suponiendo un 33,16% de los perfiles ofertados. Las competencias más solicitadas son las de trabajo en equipo (un 5,86% de perfiles, un 14,52% de los perfiles que citan competencias), la de iniciativa/proactividad/dinamismo (3,69% y 10,22% respectivamente) y la de capacidad de comunicación y relación (3,58% y 9,68% respectivamente). Les siguen la capacidad de organización/gestión, de aprendizaje, dirección de equipos, orientación a cliente y autonomía e independencia (ésta ligada a puestos de dirección).

En cuanto a otras condiciones destaca la disposición para viajar que, en diversos modos, incide en el 10,69% de los perfiles (viajar en general en un 8,73%, específicamente para España en un 1,07% y fuera de España en un 0,89%).

4.3. Requisitos de conocimientos técnicos

En el estudio RENTIC se clasifican las necesidades técnicas en 8 categorías: bases de datos, comunicaciones y software de red, entornos de desarrollo y gestión, lenguajes, equipos, sistemas operativos, desarrollo e ingeniería de software y una miscelánea de otros requisitos.

Oracle domina las bases de datos con SQLServer como segundo. La administración de bases de datos ha crecido claramente (Tabla 3).

Base de datos	Nº Perfiles 2001-2	% 2001-2
ORACLE	85	18,44%
SQL Server	43	9,33%
DB2	29	6,29%
Administración BD	27	5,86%
SQL	22	4,77%
BD en general	20	4,34%
INFORMIX	14	3,04%
IMS	12	2,60%
PL/SQL	11	2,39%

Tabla 3. Conocimientos de bases de datos

En esta categoría (ver Tabla 4) hay que distinguir entre los requisitos de conocimientos de comunicaciones y redes y los correspondientes a software y entornos de red como servidores web y de aplicaciones, etc. En la primera subcategoría, dominan más los conocimientos genéricos de comunicaciones y redes (10,20 %), de Internet,

[15] En ningún caso se han considerado las titulaciones de telecomunicación.

[16] Aunque el autor cuenta con perfiles de cualidades intelectuales, de carácter, de aptitud y sociales (como antes de llamaba a las competencias) para perfiles informáticas creados unos 20 años atrás.

Intranet y web (6,07 %) y de TCP/IP (5,64%). Les sigue CICS (4,56%) como entorno de comunicación intrasistema, la administración general de redes y sus herramientas genéricas (3,47%), las redes locales (3,47%) y los routers (3,25%). En entornos de software relacionados con redes y comunicaciones (servidores web, de aplicaciones, etc.), el trío de cabeza está formado por Exchange y los servidores de aplicaciones WebSphere y Weblogic (todos con un 1,95%).

Conocimientos de comunicaciones	Nº Perfiles 2001-2	% 2001-2
Redes/Comunicaciones	47	10,20%
Internet/Intranet y Web	28	6,07%
TCP/IP	26	5,64%
CICS	21	4,56%
Administración de red y herramientas	16	3,47%
LAN	16	3,47%
Routers	15	3,25%
WAN	15	3,25%

Tabla 4. Conocimientos de comunicaciones y redes

En esta categorías distinguimos entre los entornos de gestión (ERP y similares) y los entornos y arquitecturas de software. En el caso de los ERP, el dominio absoluto corresponde a SAP (como conjunto incluidos sus módulos y ABAP) con un 10,41% de perfiles. A más distancia aparecerían el CRM de Siebel (con un 1,74% tiene una continua progresión desde 2000-1) y las aplicaciones de Oracle y JDEdwards con menos de un 1%. En entornos de desarrollo de software destaca Developer de Oracle (5,42%) seguido de herramientas de formularios e informes como Oracle o Cristal (2,6%). En arquitecturas generales, lidera Lotus Notes/Domino (3,47%), seguido por el cliente-servidor genérico (1,52%).

Entornos	Nº Perfiles 2001-2	%2001-2
SAP	48	10,41%
Developer 2000	25	5,42%
Lotus Notes/Domino	16	3,47%
Reports y/o Forms (Crystal, Oracle,...)	12	2,60%
Siebel	8	1,74%
Cliente-servidor	7	1,52%

Tabla 5. Conocimientos de entornos

Dentro de los lenguajes se han acumulado los datos tanto de lenguajes de programación en sentido amplio como de lenguajes de marcado. Existe una dura pugna entre el mundo de Java (Java, Javascript, J2EE, JSP y EJB: 19,95%) frente al mundo Microsoft (Visual Basic, VBScript, .NET y ASP: 18,65%). Consolidando la suma de referencias se ve un empate práctico entre los lenguajes Java y VB (10,41% frente a 9,98%) con ligera ventaja del primero, también reflejada en los ámbitos antes mencionados.

Lenguaje de programación	Nº perfiles	% 2001-2
Java	48	10,41%
Visual Basic	46	9,98%
C++	31	6,72%
HTML	24	5,21%
ASP	23	4,99%
COBOL	22	4,77%
Java Script	21	4,56%
C	21	4,56%
Visual C++	17	3,69%
XML	13	2,82%
Delphi	12	2,60%

Tabla 6. Conocimientos de lenguajes

En cuanto a equipos y hardware específico y actividades asociadas, lidera la acción de mantenimiento e instalación (5,64%) seguida por el mundo PC y el de AS/400, ambos con 3,69%.

En ingeniería del software, lo más demandado es el conocimiento funcional específico de sector o actividad (8,03%), el análisis de aplicaciones (6,51%) y la gestión de proyectos (5,86%). En herramientas, lidera Oracle Designer (3,04%) y, en metodologías y notaciones, ganan UML (1,95%) y MÉTRICA (1,73%) .

Sistema operativo	Perfiles	% 2001-2
Windows NT	66	14,32%
UNIX	63	13,67%
Windows2000	45	9,76%
Administración de sistemas	29	6,29%
Windows en general	24	5,21%
LINUX	23	4,99%
Solaris	22	4,77%
MVS	14	3,04%
OS/390	11	2,39%
HP-UX	11	2,39%

Tabla 7. Conocimientos de sistemas operativos

De nuevo existe un pugna entre Microsoft (Windows en sus distintas variantes: NT, 2000,

9x, etc.) y, en este caso, el mundo UNIX (Unix, Linux, Solaris, HP-UX, etc.). En los primeros puestos existe una lucha entre NT (14,32%) y Unix (13,67%) que en el período anterior (2000-1) era ganada por Unix con claridad. Aunque pudiera parecer que Linux progresa no lo hace tanto (4,99%) aunque mantiene posición destacada sobre los grandes ordenadores (OS/390: 2,34%). La actividad de administración de sistemas resulta especialmente solicitada y creciente (6,29%).

En la última categoría de requisitos técnicos que sirve de miscelánea, la seguridad (6,07%), la implantación de ERP (4,56%) y el soporte de MS-Office (3,47%) acaparan las primeras posiciones.

5. Algunas conclusiones

El conocimiento obtenido de los estudios RENTIC ha permitido que en la UEM se pueda ajustar mejor la oferta de asignaturas de libre configuración para paliar la rigidez legal y práctica de los planes de estudio. También la optatividad es más fácil de adaptar. Sin embargo, el mayor esfuerzo debe hacerse, primero, en la especialización de la universidad en la formación de titulados. La simple formación generalista de titulados no es apropiada para la competencia entre universidades, que deben definir su nicho de alumnos con intensificación en un perfil concreto.

En segundo lugar, es preferible la filosofía de incorporar, en las prácticas y créditos prácticos, aquellos entornos/equipos acordes con el perfil de especialización del centro y/o que sean más demandados laboralmente. Las dificultades surgen aquí en dos frentes: la resistencia del profesor a favor de entornos poco implantados que "teóricamente" son "mejores" y las dificultades económicas o logísticas para contar con dichos entornos/equipos para la docencia. En el caso de la UEM, además de las inversiones propias, ha dedicado gran esfuerzo a incorporarse a iniciativas económicas o gratuitas de cesión de licencias o equipos (OAI, MSDNAA, IBM SP, etc.).

También es muy importante resaltar que la empleabilidad en TIC no se puede basar en una simple colección de conocimientos inconexos (quizás como ciertas academias quieren vender), sin una sólida base de fundamentos técnicos. Hay habilidades que rara vez se incorporan como requisitos en las ofertas pero que son esenciales para el desempeño profesional: la capacidad analítica, el buen diseño, la autogestión del tiempo y de los recursos, la capacidad de aprendizaje, etc. suelen estar detrás de enunciados más sencillos. No obstante, otro acción de mejora de la empleabilidad son los seminarios donde se presentan estos datos y la compleja terminología asociada. La dificultad está en contar con profesores dispuestos a seguir con atención la evolución técnica y de mercado.

Por otra parte, en la UEM, se aplica un plan de desarrollo de competencias profesionales para los alumnos, se adaptan seminarios técnicos a los datos obtenidos de RENTIC y se difunden a profesores y alumnos las conclusiones del estudio (junto a otros informes con la colaboración de ponentes del sector con influencia en RRHH).

Referencias

[1] W.S. Humphrey, D. H. Kitson y T.C. Kasse, *Software Engineering Practice: Preliminary Report.CMU//SEI-89-TR-1*, Carnegie-Mellon University, Febrero, 1989.

[2] Accenture, *Expectativas de los universitarios españoles*, Accenture, 2001.

[3] Antonio Pulido, "El reto de la nueva información TIC", Revista Fuentes Estadísticas, nº 151, febrero, 2001.

[4] Círculo de Progreso, Informe INFOEMPLEO 2001, Círculo de Progreso, 2002 (www.circulodeprogreso.com).

[5] A. Milroy y P. Rajah, *Europe's Growing IT Skills Crisis*, IDC, 2000.

[6] WEF, *Worldwide ICT Professionals Market Situation Study*, World Electr. Forum, 2000.

[7] SEDISI, *Estudio sobre Salarios y Política Laboral en el Sector Informático 2001*. SEDISI, 2002.

[8] FORMAN, *Necesidades de formación profesional ocupacional en comunicaciones y TI*, FORMAN, 1999.

[9] Proyecto PAFET, *Propuesta de Acciones para la Formación de profesionales de la Electrónica, Informática y Telecomunicaciones*, 1ª edición, 2001.

[10] Proyecto PAFET, Propuesta de Acciones para la Formación .., 2ª edición, 2002.

Propuesta docente sobre Sistemas de Información Geográfica

Michael Gould
Dpto. de Lenguajes y Sistemas Informáticos
Universitat Jaume I
12080 Castellón
e-mail: gould@uji.es

José Poveda
Dpto. de Lenguajes y Sistemas Informáticos
Universitat Jaume I
12080 Castellón
e-mail: albalade@uji.es

Resumen

En el presente artículo se presenta una propuesta de contenidos para una primera asignatura sobre Sistemas de Información Geográfica dentro del currículo del Ingeniero en Informática. Inicialmente se realiza una descripción general de la misma en la que se plantean sus objetivos principales. Posteriormente se mencionan algunas características de la metodología empleada en cada una de las actividades que se realizan y un resumen de los programas de teoría y prácticas. En el siguiente apartado se profundiza en los detalles de cada unidad teórica indicando sus objetivos, contenidos, duración, ejercicios propuestos, prácticas relacionadas y bibliografía de consulta. A continuación se analiza el programa de prácticas planteando también sus objetivos, contenidos, duración, ejercicios propuestos, unidades teóricas relacionadas y bibliografía recomendada. Finalmente se mencionan aspectos relacionados con la evaluación de la asignatura y bibliografía.

1. Objetivos de la asignatura

En la Universitat Jaume I (UJI) la asignatura objeto de análisis es una asignatura optativa de primer ciclo, que se imparte durante el segundo semestre del curso académico y que tiene como prerrequisitos las asignaturas obligatorias o troncales Sistemas Operativos I y Estructuras de Datos, y la optativa Informática Gráfica.

La asignatura SIG en la UJI tiene un carácter especial con respecto a las de otras universidades en España, meramente por encontrarse en Informática en lugar de en Geografía o Ciencias ambientales (donde es obligatoria), por citar algunos ejemplos. El procesamiento de los datos espaciales o geoespaciales resulta ser un tema agradable para los alumnos, dadas sus múltiples aplicaciones del "mundo real", como por ejemplo estudios medioambientales, de gestión de tráfico o del seguimiento de taxis u otras entidades móviles hasta usuarios de teléfonos celulares y los nuevos servicios basados en la localización (LBS). Además el SIG engloba muchos de los conocimientos y técnicas adquiridos anteriormente: gráficos, bases de datos, estructuras de datos, programación, geometría computacional, etc. Según el actual plan de estudios (del año 1991), cuando llegan los alumnos a esta asignatura, en tercero, empiezan simultáneamente la asignatura troncal de Archivos y Bases de Datos, y ya han cursado temas como Programación I, Algorítmica, Estructuras de datos e Informática Gráfica.

Esta base de conocimientos tecnológicos permite que la parte introductoria de la asignatura (la que corresponde normalmente a los primeros capítulos de cualquier manual SIG) se enfoque directamente en los aspectos especiales de los datos espaciales, mientras en las disciplinas menos técnicas haría falta dedicar unas horas explicando los conceptos básicos tales como ficheros, sistemas operativos y componentes físicos del ordenador. Además, cuando en el temario se analizan los Sistemas Gestores de Bases de Datos como uno de los componentes fundamentales de un SIG, no es necesario empezar con los conceptos básicos, ya que han sido estudiados recientemente en la

asignatura troncal correspondiente. Esto permite que la asignatura avance directamente hacia los problemas que conlleva el uso del modelo relacional y las posibilidades que ofrecen los sistemas orientados a objetos (u objeto-relacionales).

El resultado es tal que, al final del semestre, los alumnos muestran conocimientos suficientes como para abordar un proyecto sencillo SIG incluyendo su programación a base de componentes, mientras en otras disciplinas los alumnos necesitarían una segunda asignatura, normalmente llamada "Técnicas de SIG" [1] antes de contemplar la programación SIG. En resumen, la naturaleza de los alumnos hace que en una sola asignatura, y de tan solo 5 créditos, el profesor pueda impartirles conocimientos y experiencia que en otras disciplinas requerirían dos o más asignaturas. Naturalmente la estructura de este tipo de asignatura SIG será diferente de la ofertada en muchas otras universidades españolas, y es el objetivo de este capítulo describir y justificar dicha estructura.

Los Sistemas de Información Geográfica son hoy en día una disciplina muy especializada (se forman pocos especialistas) y a la vez muy dispersa (los alumnos encuentran trabajo con relativa facilidad en un amplio rango de centros usuarios de SIG), debido fundamentalmente a su naturaleza. La información con referencias geográficas, o susceptible de ser georeferenciada, está presente en la gran mayoría de las bases de datos del sector público (algunos expertos afirman que hasta en el 80% de ellas [2]). Por otro lado, los avances recientes en la capacidad de los microordenadores para almacenar y procesar imágenes de satélite y datos vectoriales, ha multiplicado por un factor de 10 el número de usuarios de estos sistemas en los últimos 10 años, y las nuevas aplicaciones SIG basadas en la plataforma web prometen aún más. El mercado de trabajo actual tiene necesidad de especialistas capaces no sólo de ser superusuarios, programando aplicaciones en lenguajes de cuarta generación (macros), sino técnicos capaces de desarrollar nuevos sistemas integrados partiendo desde cero. Considerando este entorno laboral el desarrollo de los contenidos de la asignatura se plantea basado en el cumplimiento de los siguientes objetivos:

- Mostrar las particularidades de la información geográfica, desde sus raíces filosóficas hasta sus estructuras de datos más comunes.
- Dar a conocer, de forma básica, la funcionalidad de las dos clases de aplicaciones SIG: basadas en la manipulación de datos raster y vectorial.
- Experimentar con las posibilidades de desarrollo de aplicaciones geográficas mediante la utilización de librerías de componentes ActiveX.
- Iniciar al alumno en el conocimiento de la migración de sistemas monolíticos a sistemas integrados y más abiertos.
- Conseguir que el alumno conozca las diferentes etapas del proceso del diseño y puesta en marcha de un proyecto SIG.
- Situar al alumno en un nivel de conocimiento que le permita criticar, evaluar y decidir sobre el uso de una u otra técnica para el tratamiento de la información geográfica.

Finalmente, como corresponde a una asignatura técnica introductoria, el temario plantea sesiones prácticas que refuerzan directamente los temas teóricos implicados.

Carácter	Créditos	Créditos Teóricos	Créditos Práct.	Curso	Centro	Titulación
Optativa	5	3	2	3	Escuela Superior de Tecnología y Ciencias Experimentales	Ingeniería Informática

Figura 1. Características de la asignatura Sistemas de Información geográfica

2. Metodología Docente

Inicialmente, en la clase de presentación de la asignatura se sientan las bases de funcionamiento y la metodología de trabajo, tanto en las clases en el aula como en el laboratorio.

2.1. Clases en el aula

Debido al número medio de alumnos en la asignatura (en torno a 50), las clases se desarrollarán necesariamente en grupos numerosos. Esto impone que sea la lección magistral el método principalmente empleado, utilizándose en la medida de lo posible la lección dialogada. Este último método será el empleado fundamentalmente en las clases sobre aplicaciones de SIG que se incluyen en la programación de la asignatura, en las que la participación del alumno es más activa y en las que pueden incluir aportaciones personales sobre el tema. De esta forma, en este tipo de clase se puede conocer el nivel de los alumnos y hacer la clase más motivadora y amena, favoreciendo el acercamiento al alumno.

Es fundamental que estas clases de teoría estén preparadas con gran cuidado ya que la experiencia demuestra que el alumno acude a clase sin haber leído prácticamente nada de la materia que se le va a explicar por lo que el primer conocimiento que va a recibir procede de la explicación teórica del profesor, de aquí la importancia de la forma de impartir estas lecciones, que deben cumplir una serie de requisitos:

- Desarrollar el tema con vitalidad y entusiasmo.
- Desarrollar de forma ordenada y destacando los puntos fundamentales.
- Incluir, donde resulte relevante, ejemplos de uso en el "mundo real" de los conceptos objetivo de la lección.
- Incluir demostraciones de aplicaciones SIG en funcionamiento, con la ayuda del proyector.
- Hacer el máximo esfuerzo para que las ideas que se transmiten sean comprendidas por la mayor parte de los alumnos y no solamente por los más destacados.
- Mantener una actitud abierta fomentando el planteamiento de cuestiones que despierten la atención del alumno de forma que participe creativamente.

El alumno debe disponer del material adecuado para el seguimiento de las explicaciones del profesor (apuntes, fotocopias de las transparencias de gráficos, el libro de texto, diapositivas, videos, software multimedia, etc.).

Respecto al esquema de organización de la clase es recomendable estructurar ésta en tres fases:

La primera fase, de unos 5 a 10 minutos, se dedicará a presentar brevemente los puntos que se van a tratar de modo que el alumno se haga una idea de conjunto del tema que se explicará. Además se tratará de enlazar con lo que ya se había explicado en las lecciones anteriores de manera que esta misma idea global conduzca la clase como una parte de una unidad más amplia relacionando los contenidos con otros temas ya tratados o incluso con temas que serán tratados más adelante a fin de situarlo dentro del contexto general de la materia.

En segundo lugar se desarrolla el contenido propiamente dicho del temario haciéndolo de forma que se destaquen los aspectos más relevantes y relacionando los conceptos que se van introduciendo con los ya conocidos. Es conveniente de vez en cuando dar ideas intuitivas a los conceptos más complejos. La organización de la explicación y los medios que se utilizan para hacerlo suelen ser fundamentales, las definiciones, conceptos o algoritmos más importantes deben permanecer a la vista del alumno durante toda la clase en zonas perfectamente separadas y delimitadas de las que se utilizan como borrador para los ejemplos. Se fomentará la participación de los alumnos de manera que si estos no realizan preguntas, será necesario provocarlas para que se hagan o incluso hacerlas a fin de que nazca un espíritu de diálogo en el seno de la clase. Esta fase contiene la parte central de la lección, que puede durar entre 40 y 45 minutos, según la duración

establecida para dicha clase. La exposición ha de ser rigurosa en todos los sentidos. Al principio hay que sentar todas las hipótesis de partida en que se basa la explicación haciendo énfasis en lo que físicamente implica cada una de ellas y no utilizar ninguna que no haya sido introducida explícitamente.

En la tercera fase, se debe hacer una recapitulación, resaltando los puntos esenciales de la lección explicada y conectándolos con los contenidos de la próxima. También es el momento adecuado para señalizar la lectura correspondiente en la bibliografía. La duración aproximada de esta fase no debe exceder, por lo general, de diez minutos.

Del conjunto de 30 horas de teoría se dedican quizás un 5% a explicar como la materia cubierta en cada clase tiene aplicación en el campo SIG, y especialmente como enlaza con cada una de las sesiones prácticas futuras. En concreto se puede observar en el siguiente gráfico (Figura 2) la distribución del tiempo dedicado a clases teóricas y a las prácticas.

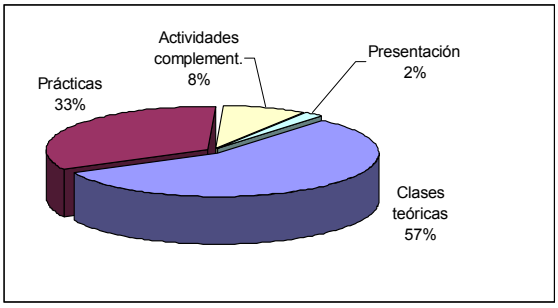


Figura 2. Características de la asignatura Sistemas de Información geográfica

2.2. Actividades en el laboratorio

En el laboratorio el alumno desarrollará fundamentalmente su trabajo práctico en contacto con la aplicación SIG objeto de estudio. Inicialmente se forman dos grupos de prácticas compuestos por un máximo de 25 personas por grupo, lo que permite que cada alumno pueda ocupar un ordenador. Se trata de ofrecer siempre un grupo con horario de mañana y otro de tarde, facilitando así la asistencia a todos alumnos que puedan estar trabajando fuera del campus universitario. Las prácticas se realizan individualmente, pero siempre permitiendo la comunicación entre los compañeros para la resolución de problemas. Al final de cada sesión de prácticas (o durante la misma semana) debe entregarse una memoria con los resultados de la práctica, incluyendo respuestas a las preguntas planteadas y los problemas desarrollados.

La metodología que se utiliza en prácticas es la de estudio programado. Para ello, los profesores involucrados en la asignatura han diseñado una serie de prácticas con documentos en los que se indica paso a paso como operar cada paquete de software o como programar unas aplicaciones simples. Aunque las sesiones han sido diseñadas para poderse realizar por completo durante la sesión de dos horas, todo el software utilizado se encuentra a la disposición del alumno, bien instalado permanentemente en los ordenadores de las salas públicas del campus o, en el caso del software de dominio público, disponible mediante disquete, CD-ROM o la página web de la asignatura. Se ha cuidado mucho este aspecto de disponibilidad del software, optando en determinadas ocasiones por un paquete de software quizás no de la última versión pero que sea disponible sin protección, o por una versión reducida “de alumno” de prueba durante 60 días, etc. Se ha demostrado durante los últimos cursos

que el hecho de que el alumno pueda disponer (o acceder con facilidad) del software aumenta su interés en el trabajo a realizar.

De las 20 horas de las que se dispone para la realización de las prácticas, se dedican 16 horas a las prácticas de estudio programado y las otras 4 a dos "visitas de campo". La primera es una visita fuera de la Escuela con receptores de señales GPS, recopilando puntos que posteriormente son volcados en un SIG en el laboratorio. La segunda es una visita técnica a la empresa mixta Facsa Sistemas (gestión de aguas de la provincia de Castellón), donde los alumnos asisten a una demostración del sistema de telemando y de las aplicaciones para la gestión de aguas basadas en un SIG orientado a objetos (*Smallworld GIS*). Como podía esperarse, los alumnos muestran un alto nivel de interés en estas practicas de campo.

3. Seminarios y conferencias

En los seminarios se deben tratar todos aquellos temas que por su naturaleza se apartan del desarrollo sistemático del programa de la asignatura. Su objetivo es el tratamiento con profundidad de algunos temas paralelos, adicionales o de actualidad, mostrándole al alumno que la temática de la asignatura no es algo cerrado y concluido, sino algo vivo en el que la comunidad científica está trabajando para ampliar los conocimientos que actualmente se tiene del tema.

Los seminarios deben servir también para iniciar al alumno en la investigación bibliográfica en determinadas áreas. Para ello se pueden proponer temas relacionados con la asignatura que pueden ser desarrollados por expertos en la materia. El tema será centrado por el profesor de la asignatura en clases anteriores y comentado con los alumnos.

Los seminarios propuestos están orientados a suplir las carencias en la formación debido al carácter único de la asignatura en el plan de estudios, mientras que las conferencias pretender abrir esos horizontes de los que se ha hablado anteriormente.

4. Programa teórico

De acuerdo con los objetivos fijados para la asignatura, y tenidas en cuenta las recomendaciones curriculares de los organismos internacionales [1] [3] para determinar los contenidos de una asignatura de estas características en un plan de estudios de Informática, la asignatura abarca todas las áreas de contenidos mencionadas incluyendo las tendencias actuales. En este sentido se incluyen en los temas, anexo I, conceptos que coinciden entre los varios currícula mencionados en el capítulo previo para un curso introductorio sobre SIG, además de incluir las últimas novedades de una disciplina en continua evolución (entre ellas los nuevos formatos gráficos para el web, la interoperabilidad a través de normas internacionales, integración mediante componentes, etc.).

Asimismo, en la elaboración del programa se han tenido en cuenta las características de los cursos introductorios sobre SIG de las universidades norteamericanas consultadas. También han influido notablemente los programas de las universidades españolas, en concreto los de las Universidades Politécnica de Madrid [4] y Autónoma de Barcelona [5]. Sin embargo, y debido a la limitación en cuanto al número de créditos de la asignatura (y al hecho de que sea la única asignatura sobre el tema durante la carrera de Ing. Informática) algunos conceptos (como por ejemplo aquellos relacionados con la programación y los formatos de datos espaciales) se introducen en la parte de prácticas de laboratorio.

Sobre el contenido de las prácticas, annexo II, se ha intentado también incorporar las recomendaciones de las fuentes citadas, intentando establecer una fuerte relación entre los contenidos prácticos y la evolución de los conceptos teóricos. El diseño de las prácticas se plantea de forma que a final de las 15 semanas los alumnos habrán tenido contacto "físico" con una buena gama de aplicaciones SIG, y habrán programado en el ambiente más puntero basado en componentes ActiveX y lenguajes visuales. Aunque está previsto ofrecer algunas prácticas empleando componentes del mundo Java, en este momento hay una carencia de software SIG, de

gama educativa, en Java; también es cierto que los alumnos de tercero en la UJI todavía no habrían visto Java como lenguaje de programación, ya que se enseña Python y C en los primeros cursos.

5. Programa práctico

El contenido del programa práctico gira en torno al aprendizaje de algunas de las aplicaciones SIG que en la actualidad se encuentran más extendidas. Sin embargo, se introducen también temas de más bajo nivel, que requieren el estudio y la modificación de código para resolver problemas específicos.

El programa de prácticas se estructura en 10 sesiones de prácticas de 2 horas cada una. Las prácticas siguen directamente al contenido de las clases teóricas, por lo que las dos primeras tratan del uso de un SIG raster, las dos siguientes de un SIG vectorial, etc.

Aunque fuera del tiempo programado de clases prácticas, se anima a los alumnos a asistir a las conferencias sobre gráficos en general y específicamente sobre SIG que se imparten dentro del ciclo de conferencias del departamento y a los seminarios introductorios impartidos por miembros del Grupo de Gráficos.

6. Programación temporal

Para la programación temporal, secuenciación, solapamiento y sincronización de las clases teóricas y prácticas de la asignatura a lo largo del semestre, hay que tener en cuenta los siguientes factores:

- El semestre consta de 15 semanas lectivas
- La asignatura tiene 3 créditos teóricos, equivalentes a 30 horas de clase (aproximadamente).
- La asignatura tiene 2 créditos prácticos, equivalentes a 20 horas de laboratorio.
- La distribución de la carga lectiva debería ser lo más equitativa posible a lo largo del semestre.
- Las clases prácticas tendrían que ir acordes a los conocimientos impartidos en teoría, situación que no siempre está de acuerdo con lo expuesto en el punto anterior.

Dados los aspectos anteriormente mencionados, la programación de las clases se ha realizado basándose en los siguientes criterios:

- Las clases teóricas se impartirán durante 15 semanas a razón de dos horas semanales.
- Las clases prácticas se impartirán durante 10 semanas a razón de una sesión de dos horas cada una.
- El proyecto (práctico) final es una continuación directa de la última práctica realizada en el laboratorio.

Las clases teóricas comienzan la primera semana del semestre. Las clases prácticas comienzan la semana siguiente para hacerlas corresponder con el inicio del segundo tema, introducción al SIG raster.

7. Conclusiones

Se ha desarrollado en el presenta artículo una propuesta de contenidos teóricos y prácticos para una asignatura optativa del currículo del Ingeniero en Informática, así como una metodología docente para la puesta en práctica de la misma y el desarrollo temporal de los contenidos fijados en los programas de teoría y prácticas.

Como complemento a las prácticas programadas para la asignatura, el alumno debe realizar un proyecto final de prácticas que es evaluado para obtener la nota final de la asignatura.

Al evaluar la asignatura se tendrán en cuenta las diferentes partes de la que esta se compone. Por un lado se evalúa la parte teórica mediante la calificación de una prueba escrita y por otro lado la parte práctica mediante la evaluación del proyecto final.

Para evaluar la asimilación de los conceptos introducidos en las clases de aula se propone la realización de una prueba escrita objetiva.

Con respecto a la parte práctica, la evaluación del proyecto final se realiza basándose en unos requerimientos básicos que el alumno debe cumplir. Cualquier elemento que se añada debe contribuir en la calificación del alumno.

Referencias para el alumno

Destacar que tanto la bibliografía básica como la secundaria que seguidamente se detalla y que se proporciona a los alumnos como referencia de los contenidos teóricos y prácticos de la asignatura, debe estar disponible en la biblioteca del campus para que los alumnos tengan un acceso real a las fuentes de información.

Básicas

AA. VV., Materiales fotocopiados y disponibles en reprografía (algunos materiales están disponibles además en formato digital, accesibles desde la página web de la asignatura).

Bosque Sendra, Joaquín. *Sistemas de Información Geográfica*, Ed. Rialp: Madrid (2ª Edición, 1997).

Gutiérrez Puebla, Javier y Gould, Michael. *SIG: Sistemas de Información Geográfica*. Ed. Síntesis: Madrid. 1994.

Worboys, Michael. *GIS: A computing perspective*. Londres: Taylor & Francis. 1995.

Secundarias

Burrough, Peter y MacDonnell, Rachel. *Principles of Geographical Information Systems*. Oxford University Press. 1998.

Chuvieco Salinero, Emilio. *Teledetección*. Ed. Rialp: Madrid. (3ª edición, 1996).

Laurini, Robert y Thompson, Derek. *Spatial Information Systems*. Londres: Academic Press, 1992.

Macmillan, B. y otros (Eds.). *Geocomputation*. Londres: John Wiley. 1998.

Maguire, David y otros (Eds.). *Geographical Information Systems*. 2 volúmenes. Londres: Longman Scientific, 1991.

Referencias

[1] [NCG90] National Center for Geographic Information and Analysis (NCGIA), "NCGIA Core Curriculum in GIS", University of California: Santa Barbara, CA., 3 volúmenes.

[2] [NCG99] National Center for Geographic Information and Analysis (NCGIA), "NCGIA Core Curriculum in GIScience", University of California: Santa Barbara, CA. Revisión de NCGIA90, disponible on-line en el siguiente URL: http://www.ncgia.ucsb.edu/giscc/

[3] [Fran95] A. Frank (Ed.), Materials for a Postgraduate Course on GIS, Publicación del proyecto subvencionado por programa Comett. 3 Volúmenes. Depto. De GeoInformación, Universidad Técnica de Viena.

[4] Programa de la asignatura "Sistemas de Información Geográfica. Ingeniería en Geodesia y Cartografía. Universidad Politécnica de Madrid, 1997.

[5] Joan Nunes i Alonso, Programa de Sistemas de Información Geográfica I. Departamento de Geografía, Universidad Autónoma de Barcelona. 1998.

Anexo I: Programa de teoría

Unidad 1. Presentación de la asignatura. Introducción a los SIG. (2 horas).

Unidad 2. Naturaleza de los datos geográficos. (3 horas).

Unidad 3. SIG raster. Modelos de datos raster. (2 horas).

Unidad 4. Teledetección espacial. Clasificación de imágenes satélite. (1 hora).

Unidad 5. SIG vectorial. Modelos de datos vectoriales. (2 horas).

Unidad 6. Modelos digitales de terreno: MDT. (1 hora).

Unidad 7. Algoritmos: Geometría computacional básica. (2 horas).

Unidad 8. Aplicaciones de SIG. (2 horas).

Unidad 9. Sistemas de posicionamiento global: GPS. (1 hora).

Unidad 10. SIG distribuido en Internet (2 horas).

Unidad 11. Interoperabilidad a través de estándares internacionales (ISO/TC211, OGC) (1 hora).

Unidad 12. Proyectos SIG: estudio de necesidades y flujo de trabajo. (2 horas).

Unidad 13. Tendencias en los SIG O-O (ejemplo de Smallworld) (1 hora).

Unidad 14. Bases de datos espaciales. (2 horas).

Unidad 15. Integración de sistemas. SIG basados en componentes. (3 horas).

Anexo II: Programa de prácticas

Todas las prácticas que se detallan seguidamente se desarrollan en sesiones de dos horas.

Práctica 1. SIG raster I: Funcionamiento de un SIG raster y conversión de datos.

Práctica 2. Teledetección : Composición de las imágenes de satélite y análisis de mapas raster.

Práctica 3. SIG vectorial I: Formato y manejo de ficheros vectoriales, capas y tablas.

Práctica 4. SIG vectorial II: Análisis espacial: buffer, distancia, consultas, búsqueda, etc.

Práctica 5. Modelos Digitales de Terreno (MDT): Visualización en 2-D y 3-D (2,5-D).

Práctica 6. Sistema de Posicionamiento Global (GPS): Salida de la UJI con receptor portátil de GPS.

Práctica 7. SIG por Internet: Servicios de Mapas en Web e Interoperabilidad. SVG.

Práctica 8. Programación con componentes (MapObjects)

Práctica 9. Visita a usuarios de un SIG O-O : Visita en grupos a las oficinas de una empresa.

Práctica 10. Programación SIG por componentes en red: Proyecto final.

Arquitectura de ordenadores

Prácticas Experimentales de Memorias Cache

Julio Sahuquillo, José Flich y Jorge Real
Departamento de Informática de Sistemas y Computadores
Universidad Politécnica de Valencia
Camino de Vera, 14. 46021 Valencia
e-mail: {jsahuqui, jflich, jorge}@disca.upv.es

Resumen

El conocimiento de las memorias cache se considera básico e imprescindible en la formación de cualquier titulado universitario (ya sea técnico o superior) en Informática.

Su estudio se puede abordar desde distintos puntos de vista. Por una parte, desde un punto de vista teórico describiendo su funcionamiento básico: cómo se localiza un bloque almacenado, qué bloque debe reemplazarse, etc. Por otra parte, desde un punto de vista práctico, comprobando el funcionamiento estudiado, normalmente mediante un simulador.

Este artículo se enmarca dentro del punto de vista práctico y propone el uso de prácticas experimentales para el estudio de las memorias cache como complemento a los simuladores. Se trata de utilizar un sencillo programa de medida de tiempos de acceso al sistema de memoria, para comprobar experimentalmente el efecto del sistema de cache sobre las prestaciones de un computador personal. Mediante la realización de las prácticas el alumno deduce cuestiones como: ¿cuántos niveles de cache tiene el computador?, ¿cuántas vías tiene cada uno de ellos?, ¿cuál es su tamaño de bloque?, etc. Estas prácticas se han realizado por primera vez durante el presente curso académico en la Escuela Técnica Superior de Informática Aplicada y en la Facultad de Informática de la Universidad Politécnica de Valencia con un buen grado de aceptación tanto por parte del profesorado como del alumnado.

1. Introducción

Las memorias cache ejercen un papel de crucial importancia en las prestaciones globales del procesador. De hecho, hoy en día es insostenible el diseño de un procesador de propósito general o de altas prestaciones sin incorporar al menos dos niveles de memorias cache.

Aunque en los últimos lustros una gran cantidad de trabajos se orientaron a la propuesta de nuevas organizaciones de cache [1] con miras a aumentar su eficiencia, en esencia, su organización interna apenas ha variado desde sus orígenes.

Este motivo supone que apenas existan diferencias entre los contenidos teóricos de organización y funcionamiento que incorporan los planes de estudio de las distintas universidades españolas, europeas o de cualquier otro país. Estos contenidos en general se cubren en asignaturas de los primeros cursos de carrera. Así, en la Escuela Técnica Superior de Informática (ETSIA) de la Universidad Politécnica de Valencia (UPV) se cubren en la asignatura Estructura y Tecnología de Computadores II (ETC2), anual de segundo curso, y en la Facultad de Informática (FI) de la misma universidad se cubren en la asignatura Estructura de Computadores (EC), también anual de segundo curso. Para el estudio de estos contenidos se pueden utilizar libros de texto como [2], [3].

Aspectos más avanzados sobre las memorias cache, como pueden ser las caches libres de bloqueo, o protocolos de coherencia de cache, se tratan en asignaturas de cursos más avanzados, como es el caso de la asignatura Arquitectura de Computadores, de cuarto curso de la FI de la UPV.

El hecho de que los contenidos teóricos apenas difieran en distintos centros, no significa que los contenidos prácticos sean también similares. Hoy en día, en general, se suelen seguir dos alternativas: no realizar prácticas de memoria cache, o utilizar un simulador de su funcionamiento para realizarlas.

En líneas generales estos simuladores utilizan como parámetros de entrada las características de la cache (tamaño de la cache, tamaño de bloque, correspondencia y algoritmo de reemplazo) y una pequeña traza con instrucciones de referencia a memoria (lectura o escritura) junto con la dirección de memoria correspondiente. El objetivo del simulador es observar cómo se van reemplazando unos bloques a otros y calcular la tasa de aciertos. Estos simuladores pueden ser de gran ayuda de cara a que el alumno pueda por sí mismo realizar problemas y comparar los resultados.

Las prácticas que se proponen en este artículo se orientan a estudiar las prestaciones de las cache de un computador de manera experimental y comprobar el determinante papel que estas ejercen sobre las prestaciones globales del computador. A partir de los tiempos de acceso observados en el direccionamiento repetitivo de los elementos de un vector, se pueden deducir características como cuántos niveles de cache tiene el computador, el tamaño de las caches, el número de vías y el tamaño del bloque. Estas prácticas añaden un pequeño grado de complejidad respecto a las que se realizan con simuladores típicos, por lo que es necesario preparar una práctica muy guiada en cuanto a la secuencia de acciones a realizar y que ayude a la interpretación de unos resultados gráficos que, a primera vista, no son evidentes.

Cabe resaltar que estas prácticas son complementarias de las que puedan realizarse con simuladores del comportamiento de la cache.

El resto de este artículo se organiza como sigue. En la sección 2 se describe la motivación que animó a los autores a prepara un práctica de estas características. En la sección 3 se describe el concepto de *stride*, necesario para poder realizar la práctica. En la sección 4 se presenta el programa de test utilizado. La sección 5 describe la parte experimental a desarrollar por los alumnos, mientras que en la 6 se indica cómo interpretar los resultados obtenidos. Por último, en la sección 7 se presentan las conclusiones de esta nueva experiencia en el presente curso académico.

2. Motivación

Las memorias cache se incorporaron en los computadores con el objetivo de reducir la latencia de acceso a la información (datos e instrucciones). Cuando un procesador intenta acceder a un dato (o instrucción) sólo realiza el acceso físico a memoria en el caso de que el bloque que lo contiene no se encuentre almacenado en la cache. Si el procesador dispone de dos niveles de cache (que es lo usual) se busca primero en la de nivel uno, que es la más cercana al procesador y trabaja, en general, con un tiempo de acceso de uno o dos ciclos del procesador. En el caso de que la información buscada no se encuentre en la cache de nivel uno, se accede a la cache de nivel dos bastante más lenta (este acceso se realiza en paralelo con el acceso al nivel 1 en muchos procesadores). Sólo en el caso de que el acceso a ambas caches resulte en fallo, se accede a la memoria principal.

La penalización que ejercen los fallos de cache sobre el tiempo medio de acceso a los datos, y en consecuencia en las prestaciones globales del sistema, es realmente acuciante y según las predicciones tecnológicas, la gran diferencia de tiempos entre la velocidad del procesador y la velocidad de memoria continuará creciendo. Esto es debido en parte a que la tecnología empleada en la construcción de la memoria principal es de por sí mucho más lenta que la empleada en la construcción del procesador y las caches.

Los usuarios con frecuencia conocen el modelo de su procesador y la frecuencia con la que trabaja, por ejemplo, Pentium 4 a 2.4 GHz; la capacidad de memoria principal instalada y su tipo, por ejemplo 256 MB de DDR; la capacidad de su disco duro, etc. Sin embargo, la mayoría desconocen las características de la memoria cache, a pesar de su gran importancia en las prestaciones.

Los argumentos descritos, nos llevaron a diseñar una propuesta de prácticas orientadas a comprobar experimentalmente las características del sistema de cache, ya que consideramos de vital importancia en la formación de un Ingeniero en Informática ejercitar sus habilidades también sobre el *hardware* y no sólo con simuladores.

Este tipo de prácticas ya se están realizando en algunas universidades americanas de primer nivel como la de Berkeley bajo la dirección de David Patterson [5]. De hecho, el código fuente del programa que se utiliza para la realización de la práctica se encuentra disponible en el texto de Hennessy y Patterson [4]. Nuestra aportación ha sido la preparación íntegra paso a paso con el fin

de que el alumno no sólo pudiese ejecutar el programa de medida de tiempos de acceso, sino también entender sus resultados y extraer sus propias conclusiones. Con este fin, se dedica un tiempo de las clases de problemas de aula a realizar ejercicios de preparación para la práctica. Asimismo la práctica integra un conjunto de ejercicios destinados a consolidar los fundamentos teóricos y a facilitar la interpretación de los resultados experimentales.

3. Concepto de *stride*: mínimo y máximo

El programa de test que se utiliza para comprobar las características de la cache, accede de manera iterativa a los elementos de un vector de tamaño variable.

Si se accede a todos los elementos del vector de uno en uno (0, 1, 2, 3, 4, 5, 6, 7, etc) se dice que el vector se recorre con *stride* o distancia 1. Si se accede a los elementos del vector uno sí uno no (0, 2, 4, 6, 8, 10, etc) se dice que se recorre con *stride* 2. Si se accede uno sí tres no (0, 4, 8, etc) se recorre con *stride* 4 y así sucesivamente. En la Figura 1 se muestra un vector de 32 elementos donde aparecen sombreados los que serán accedidos para distintos valores del *stride*. El *stride* máximo es la mitad de los elementos del vector, y el *stride* mínimo es uno. En el ejemplo, el *stride* máximo es 16 (32/2) donde se accede sólo a dos elementos (0 y 16).

En la práctica se proponen una serie de ejercicios para que el alumno ejercite el concepto de *stride*, como base para ser capaces de analizar e interpretar los resultados. Por tanto, la secuencia obligatoria debe ser realizar los ejercicios antes de la práctica experimental propiamente dicha. Aunque ya se han realizado ejercicios en clases de problemas, dada la relativa complejidad de la práctica, es aconsejable realizar más ejercicios de cara a fomentar una sólida base común en los alumnos de los distintos grupos.

3.1. Ejercicios sobre *stride* de apoyo a la interpretación de resultados

En esta sección se presenta un subconjunto de los ejercicios que debe realizar el alumno antes de pasar a la parte experimental de la práctica.

Stride	Elementos del vector																														
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
1	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**	**31**
2	**0**	1	**2**	3	**4**	5	**6**	7	**8**	9	**10**	11	**12**	13	**14**	15	**16**	17	**18**	19	**20**	21	**22**	23	**24**	25	**26**	27	**28**	29	**30**	31
4	**0**	1	2	3	**4**	5	6	7	**8**	9	10	11	**12**	13	14	15	**16**	17	18	19	**20**	21	22	23	**24**	25	26	27	**28**	29	30	31
8	**0**	1	2	3	4	5	6	7	**8**	9	10	11	12	13	14	15	**16**	17	18	19	20	21	22	23	**24**	25	26	27	28	29	30	31
16	**0**	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	**16**	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

Figura 1. Ejemplo de un vector de 32 elementos accedido con distintos valores de stride.

Ejercicios básicos de *stride*:

Suponga un vector de enteros de 4096 bytes (1024 elementos de 4 bytes).

- ¿Qué elementos (palabras) se referenciarán si se accede con *stride* 256?
- ¿Cuál es el *stride* máximo con que se puede acceder al vector?
- Si se sabe que el máximo *stride* con que se puede acceder a un vector es 2048, ¿cuál es el número de elementos del vector? ¿Y el tamaño del vector en bytes?
- Si se accede a un vector de 64 KB con el máximo *stride* posible, ¿a cuántos elementos distintos (palabras) se referencia? ¿Y a cuántos bloques?
- Y si se accede a dicho vector con el *stride* anterior al máximo (notación máximo-1), ¿a cuántos elementos distintos (palabras) se referencia? ¿Y a cuántos bloques?

Ejercicios de *stride* máximo (determinación del número de vías):

Suponga una cache de 16KB, 2 vías, bloques de 16 bytes, y algoritmo de reemplazo LRU. Suponga asimismo que se accede a un vector de 32 KB (8K elementos de 4 bytes). Si se accede con el máximo *stride* (2 elementos) se accedería a los elementos 0 y 4 K (4096), pero las direcciones de palabra correspondiente (las direcciones de palabra son múltiplos de 4) serían la 0 y la $4096 \times 4 = 2^{14}$. Indique en la siguiente tabla la dirección de los mismos desglosada en etiqueta, conjunto y desplazamiento, asumiendo que el vector empieza a almacenarse a partir de la dirección cero.

Elemento referenciado (dirección de palabra)	Dirección (en bytes)		
	Etiqueta	Conjunto	despl.
0 (dir. palabra 0)			
4096 (dir. palabra 4096×4)			
	31 ... 13	12 ... 4	3 ... 0

- ¿Van al mismo conjunto?
- Si se accede en un bucle relativamente grande (5000 iteraciones) al vector con *stride máximo*, ¿cuántos fallos y cuántos aciertos se producirán en la segunda y sucesivas iteraciones?
- Indique la dirección de los elementos accedidos si se accede con el *stride* anterior al máximo (se utilizará la notación *máximo-1*).

Elemento referenciado (dirección de palabra)	Dirección (en bytes)		
	Etiqueta	Conjunto	despl.
	31 …13	12 … 4	3 …0

- ¿Van al mismo conjunto?
- Si se accede en un bucle relativamente grande (5000 iteraciones) al vector con *stride máximo-1*, ¿cuántos fallos y cuántos aciertos se producirán en la segunda y sucesivas iteraciones?

El siguiente cuadro resume al alumno las principales conclusiones de este ejercicio.

Se puede observar que, si el tamaño del vector es múltiplo en una potencia exacta de dos del tamaño de la cache (32KB, 64KB, 128KB, 256KB, etc.), sucederá lo mismo si se accede con *strides* máximo y máximo-1.

- Si la cache tuviese 4 vías manteniendo el resto de parámetros iguales que en el ejercicio anterior, ¿cuántos fallos y cuántos aciertos se habrían producido si se hubiese accedido con *stride* máximo en la segunda y resto de iteraciones del bucle?
- ¿Y con *stride máximo-1*?
- ¿Y si se hubiese accedido con *stride máximo-2*?

Accediendo al vector con valores decrecientes de *stride*, el valor de *stride* que precede al primero que alcanza una tasa de fallos de aproximada-

mente el 100%, permite determinar el número de vías.

Ejercicios de *stride* mínimo (determinación del número de palabras del bloque):

Suponga la misma cache y el mismo vector que en el ejercicio anterior, y que se accede al vector con *stride* 1. Indique la dirección de los 4 primeros elementos a los que se accedería. Para ello se le ofrece al alumno una tabla auxiliar de manera análoga a los ejercicios anteriores.

- ¿Pertenecen todos al mismo bloque?
- En el peor de los casos, es decir, suponiendo que siempre que se accede al primer elemento del bloque se produjese un fallo, ¿cuál sería la tasa de aciertos si se accediera con *stride* 1? ¿Y si se accediese con *stride* 2? ¿Y si se accediese con *stride* 4?

Luego, el primer *stride* que produce una tasa de aciertos del 0% determina el tamaño del bloque en palabras.

4. Programa de test

Para realizar la práctica se ofrecen dos ficheros: *benchmark.c* y *grafico.config*:

Benchmark.c es un pequeño programa de test cuyo bucle principal accede a los elementos de un vector. La Figura 2 muestra el código de este programa, resaltando el bucle principal. Los accesos a los elementos del vector se realizan con la instrucción: `x[index]=x[index]+1;` donde se realiza una lectura (*load*) para leer el operando `x[index]` y poder realizar la suma, y otro acceso para escribir el resultado (*store*). La suma de ambos tiempos (lectura+escritura) es lo que se mide en el programa. Para eliminar la sobrecarga impuesta por las instrucciones de control del bucle principal, se utiliza la técnica del doble bucle para restarle su efecto. El bucle que cuantifica la sobrecarga aparece a continuación del bucle principal.

Grafico.config es un pequeño *script* que se ofrece para formatear el gráfico de salida.

5. Parte experimental

Esta parte consiste en la compilación y ejecución del programa *benchmark.c* sobre un determinado ordenador y la visualización de los resultados utilizando para ello la popular herramienta *gnuplot* sobre el sistema operativo Linux. El laboratorio donde se realizan las prácticas está dotado de computadores de tres clases distintas, por lo que los resultados obtenidos dependen del puesto de trabajo donde se realice la práctica.

Tras la obtención experimental de los resultados, el alumno deberá realizar su interpretación, lo que confirmará si ha cubierto o no el objetivo de la práctica.

6. Análisis de Resultados Experimentales

El laboratorio en el que se realiza la práctica se encuentra equipado con tres tipos de ordenadores personales:

- Pentium II 350MHz
- AMD K6-2 450MHz
- AMD Athlon 700 MHz

La Figura 3 muestra como ejemplo el tiempo medio de acceso para distintos tamaños de vectores en el Pentium II. El alumno debe deducir a partir de los resultados cuestiones relativas a la cache de nivel 1, a la cache de nivel 2 y a la memoria principal.

6.1. Cuestiones relativas a la cache de nivel 1

Si se sabe que la cache de datos del Pentium II de nivel 1 es de 16KB, la del K6 de 32KB y la del Athlon de 64KB, responda a las siguientes preguntas:

- ¿Qué tipo de procesador se encuentra instalado en su computador?
- ¿Cuántas vías tiene la cache?
- ¿Cuál es el tamaño del bloque en palabras?
- ¿Cuál es el tiempo aproximado de cada acceso (lectura+escritura) a la cache?

```
#include <stdio.h>
#include <sys/times.h>
#include <sys/types.h>
#include <time.h>
#define CACHE_MIN (1024)        /* cache más pequeña */
#define CACHE_MAX (1024*1024)   /* cache más grande  */
#define SAMPLE 10
#define CLOCK_TCK 60.0     /* número de ticks de reloj por ciclo */
int x[CACHE_MAX];

double get_seconds() {    /* rutina para leer el tiempo */
        struct tms rusage;
        times(&rusage);    /* utilidad UNIX para medir el tiempo */
        return (double) (rusage.tms_utime)/CLOCK_TCK;
}

int main() {
int register i, index, stride, limit, temp;
int steps, tsteps, vsize;
float sec0, sec, den;

for (vsize=CACHE_MIN; vsize <= CACHE_MAX; vsize=vsize*2)
  {
    for (stride=1; stride <= vsize/2; stride=stride*2)
      {
        sec = 0;
        limit = vsize - stride + 1;  /* tamaño de vector para este bucle */
        steps = 0;

        do {                    /* repetir durante un segundo */
          sec0 = get_seconds(); /* instante de tiempo antes del bucle */
          for (i=SAMPLE*stride; i !=0; i=i-1)
            for (index=0; index<limit; index=index+stride)
              x[index] = x[index]+1;   /* bucle de accesos a cache = lectura + escritura */

          steps = steps+1;                      /* contador de iteraciones del bucle while */
          sec = sec+ (get_seconds() - sec0);   /* acumula tiempo transcurrido en el bucle */
        } while (sec < 1.0);

        /* repetir un bucle vacío para restar la sobrecarga */
        tsteps = 0;   /* variable auxiliar para realizar el mismo numero de iteraciones */
        do {
          sec0 = get_seconds();
          for (i=SAMPLE*stride; i !=0; i=i-1)
            for (index=0; index<limit; index=index+stride)
              { temp = temp+index;  }  /* dummy code */

          tsteps = tsteps+1;
          sec = sec - (get_seconds() - sec0);
        } while (tsteps < steps);

        den = steps*SAMPLE*stride;
        den = den*((limit-1)/stride+1);
        printf("Tamaño: %4d KB Stride: %7d T(lec+esc): %5.0f ns\n",
                        vsize*sizeof(int)/1024, stride, sec*1e9/den);

        /* printf(" %4d %7d  %5.0f \n", vsize*sizeof(int)/1024 , stride, sec*1e9/den); */
      }
    printf("    0     0      0 \n");  /*linea auxiliar para la presentación de resultados */
  }
}
```

Figura 2. Programa benchmark.c.

6.2. Cuestiones relativas a la cache de nivel 2

Los procesadores en los que se encuentra trabajando incorporan una cache de nivel 2 unificada para instrucciones y datos, por lo que ambos compiten por un espacio en dicha cache. Esto significa, que no se producirá un cambio brusco en el tiempo de acceso sino que habrá un cambio paulatino. El tamaño de la cache de nivel 2, lo define el tamaño del vector previo a la estabilización del tiempo de acceso para la memoria principal.

- ¿Cuál es el tamaño de la cache en KB?
- ¿Cuál es el tiempo aproximado de cada acceso (lectura+escritura)?
- ¿Cuántas veces más grande es el retardo de la cache de nivel 2 que el de la de nivel 1?

6.3. Cuestiones sobre la memoria principal

- ¿Cuál es el tiempo aproximado de cada acceso (lectura+escritura) en nanosegundos?
- ¿Cuántas veces más grande es el retardo de memoria respecto a la de nivel 1?
- ¿Qué indican los picos en los tiempos de acceso?

7. Conclusiones

Es aconsejable realizar prácticas sobre memorias cache para afianzar los conocimientos teóricos. En la mayoría de las universidades estas prácticas se realizan mediante un simulador, de gran utilidad para afianzar los conceptos básicos como determinar el conjunto donde se almacenará el bloque, aplicación de los algoritmos de reemplazo, etc.

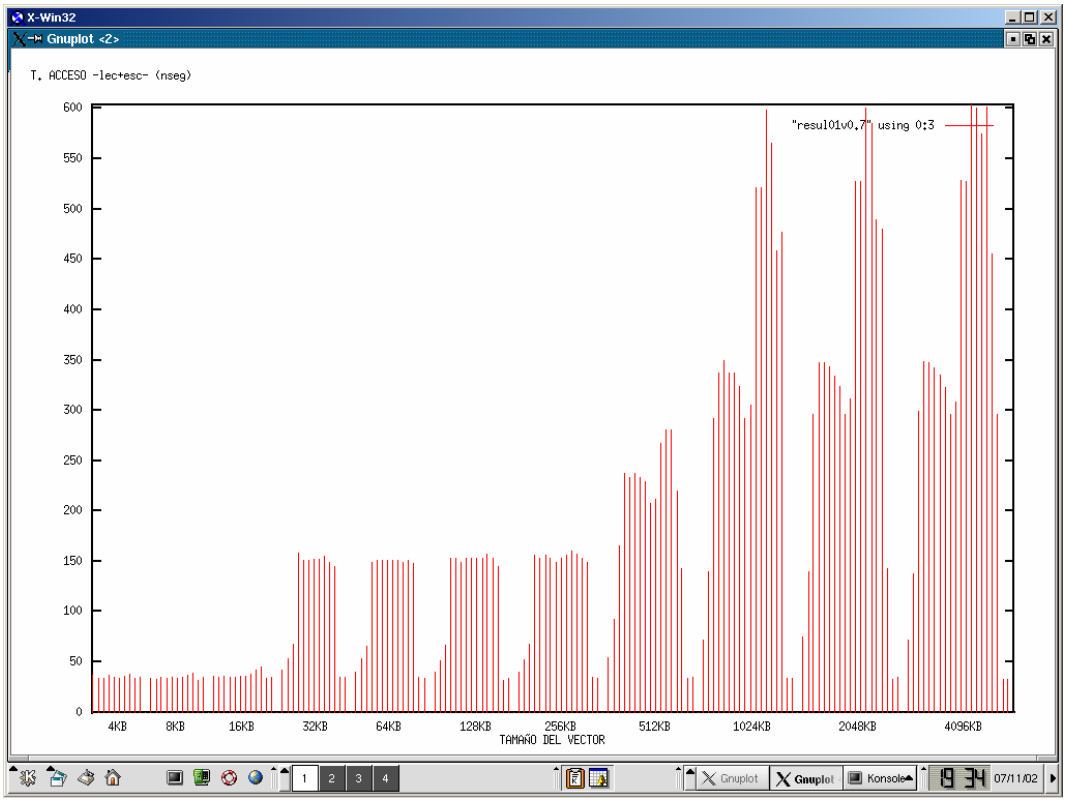


Figura 3. Resultados obtenidos sobre el Pentium II.

En este trabajo se ha presentado una práctica complementaria a las anteriores, con la peculiaridad de ser totalmente experimental ya que se realiza sobre la propia memoria cache que integra el procesador.

Como principales ventajas ofrecidas por la práctica, cabría resaltar:

- Se realiza directamente sobre el *hardware* del PC sin la utilización de simuladores.
- Permite comprobar la necesidad de la memoria cache y su importante papel en las prestaciones, "escondiendo" la latencia a memoria.
- Cuantifica (aproximadamente) los retardos de la cache, tanto de nivel 1 como de nivel 2.
- Permite comprender por qué con los retardos actuales (procesadores del laboratorio de prácticas) no es necesaria una cache de nivel 3.
- Se aprende a deducir experimentalmente las características (tamaño, nº de vías, y tamaño del bloque) de la cache de nivel 1.

Como principal inconveniente habría que mencionar que la práctica incorpora cierto grado de dificultad por lo que es necesario, dedicar una clase de teoría a explicar conceptos que ayuden al alumno a la interpretación de los resultados de la práctica.

Por último, cabe mencionar que la práctica se ha realizado por primera vez durante el presente curso, siguiendo las directrices descritas en este trabajo, y se ha podido comprobar que el alumno muestra cierto grado de entusiasmo cuando logra resolver y comprender los resultados experimentales, lo cual lo han podido conseguir un porcentaje relativamente elevado de ellos. Además, muchos han podido realizar el mismo experimento sobre sus propios ordenadores personales, lo cual es un factor adicional de motivación.

Referencias

[1] J. Sahuquillo and A. Pont, "Splitting the Data Cache: A Survey," July-September 2000 special issue of the IEEE Concurrency. ISSN1092-3063/00.

[2] D. A. Patterson, J. L. Hennessy, Estructura y Diseño de Computadores: interficie circuitería programación, Reverté, S.A. 2000 (2ª edición).

[3] V. C. Hamacher, Z. G. Vranesic, S. G. Zaky, Computer Organization McGraw-Hill, 2002 (5ª edición)

[4] J. L. Hennessy, D. A. Patterson, Computer Architecture. A quantitative approach, Morgan Kauffmann, 2002 (3ª edición.

[5] http://inst.eecs.berkeley.edu/~cs61c/fa01/calendar/week13/lab10.

Prácticas de Diseño de Sistemas de Memoria

José Flich, Jorge Real, Julio Sahuquillo
Departamento de Informática de Sistemas y Computadores (DISCA)
Escuela Técnica Superior de Informática Aplicada
Camino de Vera, s/n
46020 Valencia
e-mail: {jflich, jorge, jsahuqui}@disca.upv.es

Resumen

Los conocimientos a impartir en los estudios universitarios de Informática sobre el sistema de memoria se prestan a un aprendizaje por niveles, desde el conocimiento básico de una celda de bit hasta el diseño de un mapa de memoria. Como consecuencia, las prácticas que se diseñen para reforzar estos conocimientos también pueden seguir una organización y secuenciación por niveles.

Por esta razón se ha diseñado un conjunto de prácticas sobre el sistema de memoria, dentro del contexto de los estudios universitarios de Informática de la Universidad Politécnica de Valencia, para ser realizadas mediante una aproximación por niveles, empezando por el diseño de un chip de memoria, siguiendo por la construcción de un módulo de memoria a partir de chips y finalizando con el diseño de un mapa de memoria utilizando varios módulos de memoria de diferentes características.

En dichas prácticas se ha escogido la herramienta de diseño y simulación digital Xilinx. Esta herramienta posee una gran versatilidad y flexibilidad en todas las fases de la realización de un sistema electrónico digital, desde su diseño hasta su validación por simulación y posterior implementación. Además, el alumno también utiliza dicha herramienta en otras asignaturas de la titulación por lo que ya se encuentra familiarizado con ella. Por otra parte, Xilinx es una herramienta que podrá ser utilizada por el alumno en su cercano futuro profesional debido a su potencia y su gran aceptación.

En este artículo se describen brevemente las prácticas de diseño de sistemas de memoria que se han diseñado con Xilinx en el mencionado contexto docente.

1. Introducción

Aprovechando la puesta en marcha del plan de estudios de 2001, tanto en la Escuela Técnica Superior de Informática Aplicada (ETSIA) como en la Facultad de Informática (FI) de la Universidad Politécnica de Valencia, se realizó un esfuerzo para el planteamiento de nuevas prácticas así como el uso de nuevas herramientas para llevarlas a cabo.

En la unidad docente de Estructura del Computador (en concreto, en el seno de la asignatura Estructura y Tecnología de Computadores I) se optó por utilizar la nueva herramienta de simulación Xilinx [2] en sustitución de Cascad [1]. Esta herramienta es muy potente, pero también relativamente compleja, por lo que es aconsejable utilizarla en distintas asignaturas para amortizar el tiempo que conlleva su aprendizaje.

En la unidad docente de Estructura del Computador, la que nos ocupa, se cubre el estudio de las distintas unidades funcionales del computador, empezando en el primer curso de carrera con la asignatura Estructura de Computadores I de la ETSIA y Fundamentos de Computadores de la FI. En esta asignatura ya se han estudiado nociones básicas sobre Xilinx, por lo que en segundo curso ya se asumen conocidas.

Tanto en la asignatura de Estructura y Tecnología de Computadores II (ETC2) de la ETSIA como en la asignatura de Estructura del Computador (EC) de la FI se han desarrollado durante el presente curso prácticas de diseño de memorias. La asignatura ETC2 es una asignatura troncal de la titulación de Ingeniero Técnico en Informática de Sistemas (ITIS) y la asignatura EC es una asignatura obligatoria de la titulación de Ingeniero Superior en Informática (II).

Ambas asignaturas son anuales y en las dos se imparte la misma docencia. En particular, se imparten los conocimientos relacionados con los sistemas de memoria y de entrada/salida en el primer cuatrimestre, mientras que en el segundo cuatrimestre se imparten los conocimientos de unidades aritmetico-lógicas y segmentación básica del procesador. Las prácticas realizadas en las sesiones de laboratorio han sido preparadas para ejercitar los contenidos teóricos estudiados en las sesiones de teoría.

En lo referente a los conceptos teóricos de memoria del computador, se plantea un aprendizaje por niveles. Es decir, el alumno debe, en primer lugar conocer las estructuras internas de funcionamiento de un chip básico de memoria, conociendo el diseño de la celda de memoria y de los circuitos de direccionamiento de celdas, así como el diseño de chips de memoria con organización interna 3D y su patillaje externo. Posteriormente se aborda el diseño de módulos de memoria a partir de la construcción por bloques, utilizando para ello los chips de memoria ya conocidos. Por último, el alumno compone un mapa de memoria a partir de sus especificaciones, utilizando para ello diferentes bloques de memoria y calculando e implementando las funciones de selección de cada módulo.

Durante todo este recorrido por niveles, el alumno también debe conocer la interconexión del procesador con la memoria, haciéndose hincapié en aspectos como la selección de octeto y de palabra por parte de la UCP y las señales de bus asociadas.

Por todo esto, las prácticas relacionadas con el diseño de sistemas de memoria deben, por una parte, potenciar los conocimientos adquiridos en las clases teóricas, y por otra, deben orientarse al diseño de sistemas de memoria por niveles.

Además, se debe tender en las sesiones de laboratorio a la utilización de herramientas de trabajo de calidad profesional, en la medida de lo posible. Por un lado, la probabilidad de que el alumno trabaje en su futuro profesional con una herramienta de estas características va a ser alta, por lo que se le estará facilitando al alumno al mismo tiempo el aprendizaje de la herramienta. Por otro lado, una herramienta de calidad permitirá una adecuada implementación y evaluación de las tareas a realizar en la práctica, mejorando con ello su calidad.

Para la realización de las prácticas de diseño de sistemas de memoria, se había utilizado hasta el curso actual (en las asignaturas correspondientes del Plan de Estudios anterior) la herramienta de modelado y simulación de sistemas lógicos Cascad [1]. Aunque Cascad permitía realizar un diseño y su posterior simulación, también introducía una serie de inconvenientes que limitaban la realización de las prácticas por parte del alumno y su diseño mismo. Principalmente, Cascad limitaba el número de componentes que se podían utilizar en un esquemático (por ejemplo, solamente se podía modelar una memoria 2D de 16 bits con capacidad 16×1). Algunas prácticas, como por ejemplo la de diseño de módulos de memoria, no eran factibles. Además, las prácticas de diseño de mapas de memoria se debían simplificar para no utilizar módulos de memoria, debido a las limitaciones de la herramienta.

Los citados inconvenientes motivaron a que se utilizase una nueva herramienta de diseño que permitiese una mayor flexibilidad y capacidad para el diseño de tales prácticas. Dicha herramienta, como se ha mencionado, es la aplicación Xilinx.

Xilinx permite la realización de diseños lógicos sofisticados, tanto en la variedad de componentes que soporta por medio de la utilización de librerías, así como en el número de componentes que puede formar el sistema

definitivo. Por lo tanto, existe una mayor flexibilidad en el diseño de prácticas que motiven y permitan al alumno ejercitar verdaderamente los conocimientos adquiridos en las sesiones de teoría.

Xilinx permite además una simulación detallada de todo el circuito. Resulta muy interesante la posibilidad de incluir sondas de simulación en los propios esquemas, lo que permite una mejor evaluación del comportamiento y de las posibles causas de fallos en el diseño. Otro aspecto de interés es el hecho de que las simulaciones de Xilinx son en base a cronogramas, lo que permite ejercitar la lectura e interpretación de este tipo de diagramas. En Cascad resultaba necesario incluir en el circuito elementos ajenos a los objetivos de la práctica con el fin de facilitar la simulación de los circuitos (elementos de entrada de valores, relojes de simulación, etc.). Finalmente, otra característica muy interesante y útil de Xilinx es la posibilidad de utilizar dispositivos triestado, cosa imposible en Cascad y que nos obligaba también a utilizar circuitos adicionales (como puertas OR para unir todas las salidas de datos de las celdas de memoria).

A continuación se describen brevemente los contenidos de las asignaturas ETC2 y EC así como la planificación de las sesiones de laboratorio. Seguidamente se profundiza en las prácticas de diseño de sistemas de memorias. Por último, se finaliza con las conclusiones del artículo.

2. Las asignaturas

Como se ha comentado anteriormente, las asignaturas ETC2 y EC se imparten en titulaciones diferentes. Sin embargo, ambas asignaturas cubren los mismos contenidos teóricos y prácticos y tienen el mismo número de créditos.

Los objetivos de ambas asignaturas son proporcionar una visión completa del computador, estudiando todas sus unidades funcionales. Esto es, la unidad de memoria, la unidad de entrada/salida, los periféricos y buses y los circuitos aritméticos y sus prestaciones. Asimismo, también es su objetivo dar a conocer los aspectos de procesadores y computadores actuales para la mejora de las prestaciones.

Ambas asignaturas constan de 6 créditos teóricos y 6 créditos de prácticas, estos últimos desglosados en 3 de problemas (se imparten junto a los créditos teóricos) y 3 de laboratorio. En la Tabla 1 se puede apreciar la organización de los contenidos teóricos que se imparten.

Los contenidos teóricos se distribuyen en cuatro bloques temáticos. En los dos primeros bloques se tratan los sistemas de memoria y de E/S. El tercer bloque temático se centra en la unidad aritmética de enteros y de coma flotante, y el último bloque en aspectos de segmentación de procesadores y técnicas actuales de mejora de las prestaciones.

Bloque	Tema	Créd
Memoria	T1. El sistema de memoria	0,3
	T2. Diseño del sistema de memoria	1
	T3. Jerarquía de memoria	1
E/S	T4. Gestión de la entrada y salida	0,9
	T5. Buses del computador	1,2
	T6. Dispositivos de entrada y salida	0,4
ALU	T7. Suma y resta de enteros	0,6
	T8. Multiplicación y división de enteros	0,8
	T9. Unidad aritmética de coma flotante	0,4
Procesadores actuales	T10. Introducción. a la segmentación	0,6
	T11. Procesador segmentado	1,2
	T12. Ej. procesadores actuales	0,6

Tabla 1. Contenidos teóricos

3. Las prácticas de diseño de sistemas de memoria

Las prácticas se distribuyen en los mismos bloques temáticos, aunque en este artículo nos centraremos en las relacionadas con la memoria principal del primer bloque temático. Se han diseñado tres prácticas relacionadas con el diseño de sistemas de memoria.

3.1. Práctica 1. Circuitos de Memoria.

En la primera práctica, el alumno realiza su primer diseño de una memoria. Es por ello, que se diseña una chip de memoria sencillo con organización 3D y con capacidad de 16×4. Para ello, el alumno debe realizar un diseño por niveles. En el primer nivel, el alumno implementa una celda de bit utilizando un biestable y las puertas necesarias. La Figura 1 muestra el diseño. En el segundo nivel, el alumno implementa un plano de la memoria. Para ello, se reutilizan 16 celdas de memoria conectadas de la forma apropiada. La Figura 2 muestra el diseño de un plano de memoria. Por último, el alumno implementa el último nivel de la memoria. En este caso, el alumno conecta de la forma apropiada cuatro planos de memoria. La Figura 3 muestra el diseño del último nivel.

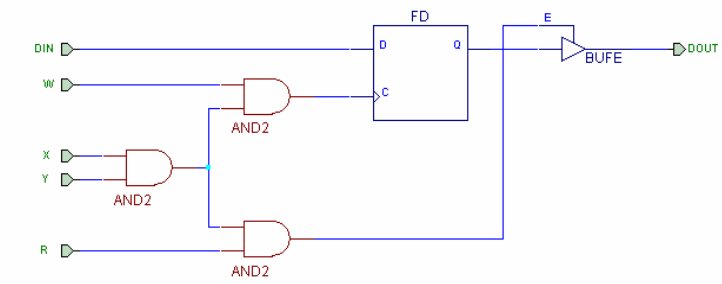


Figura 1. Diseño de una celda de memoria con Xilinx. Leyenda: R es la señal de lectura, W es la señal de escritura, X e Y son la señales de selección de celda, DIN es el bit a escribir, DOUT es el bit leído. El circuito FD es un biestable de tipo D y BUFE es un buffer triestado.

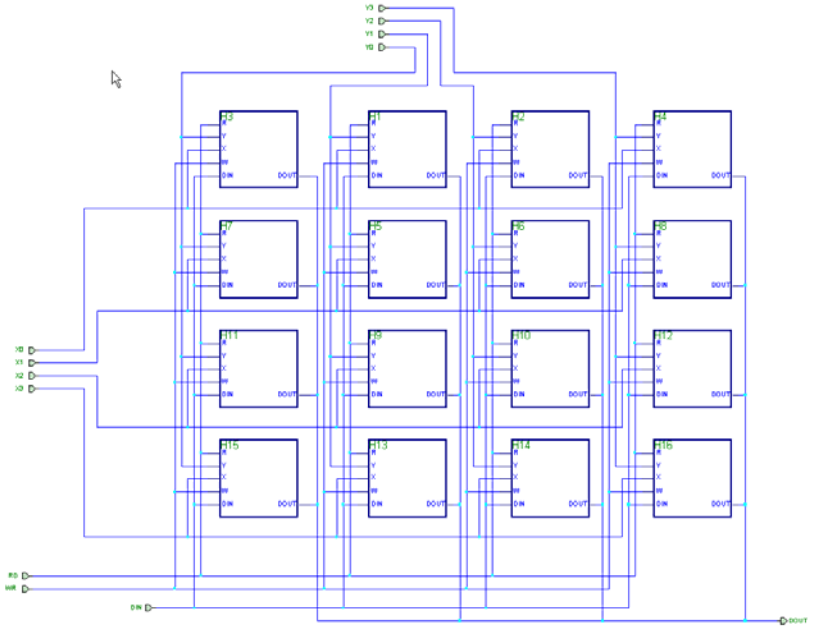


Figura 2. Diseño de un plano de memoria con Xilinx. Leyenda: RD es la señal de lectura, WR es la señal de escritura, Xi e Yj son las señales de selección de la celda, DIN es el bit a escribir, DOUT es el bit leído.

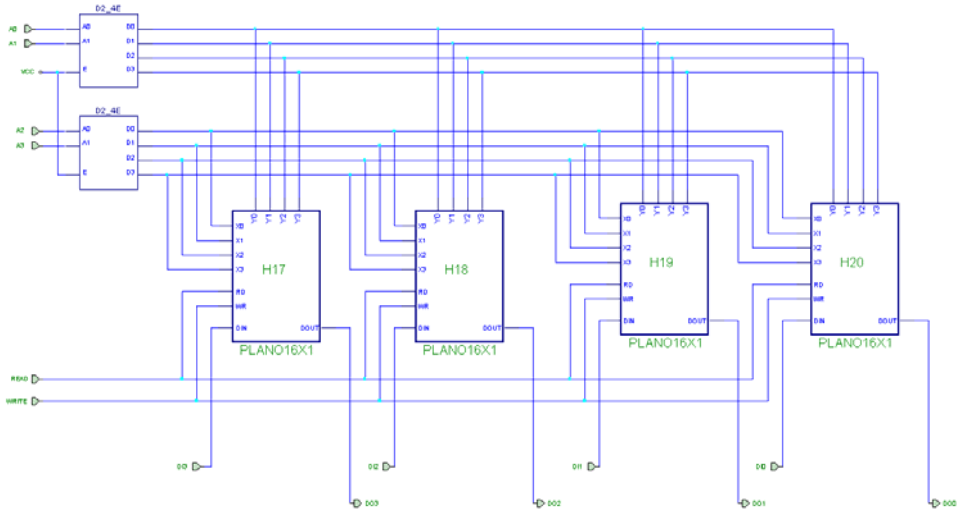


Figura 3. Diseño de una memoria 16 x 4. . Leyenda: READ es la señal de lectura, WRITE es la señal de escritura, A0A1 son los bits de selección de columna del plano, A2A3 son los bits de selección de fila del plano, DI*i* es el dato *i* a escribir, DO*i* es el bit *i* leído.

En esta primera práctica se evalúa el diseño del chip memoria implementado por medio del simulador de Xilinx. Para ello, el alumno debe realizar diferentes accesos tanto de lectura como de escritura a diferentes direcciones del chip de memoria y comprobar el correcto funcionamiento de éste, poniendo en juego las señales de dirección, datos, lectura y escritura del circuito diseñado y observando los cronogramas resultantes.

3.2. Practica 2. Módulos de Memoria.

En la segunda práctica de diseño de sistemas de memoria, el alumno implementa un módulo de memoria a partir de chips de memoria. En concreto, el alumno debe implementar una memoria de 128×16 (128 palabras de 16 bits) a partir de chips de memoria de 32×8 previamente construidos y similares al construido por el alumno en la práctica anterior. Al mismo tiempo, se debe conectar dicha memoria a una UCP con bus de datos de 16 bits. Para realizar esta práctica, el alumno debe, en primer lugar, calcular la organización interna del módulo (número de filas y columnas) y el conexionado con la UCP. Para esto, debe tener en cuenta el direccionamiento de octetos del procesador, diseñando además un circuito combinacional que genere las líneas de selección de octecto a partir de la dirección (par o impar) y del tipo de acceso (octeto o palabra) que realice la UCP.

Para comprobar el correcto funcionamiento del módulo de memoria, se deben simular diferentes accesos al módulo, tanto de lectura como de escritura, comprobando que estas operaciones se realicen correctamente. En la Figura 4 se puede ver la salida del simulador Xilinx donde se realizan dos accesos de escritura de palabra en el módulo (direcciones 64h y 66h), seguido de dos accesos de lectura de palabra en las mismas direcciones de memoria.

En esta práctica también se ejercita el concepto de direccionamiento de octetos y palabras. En concreto, se deben realizar diferentes tipos de acceso (lectura y escritura de octetos o palabras en direcciones pares o impares) y analizar los resultados obtenidos por inspección del cronograma.

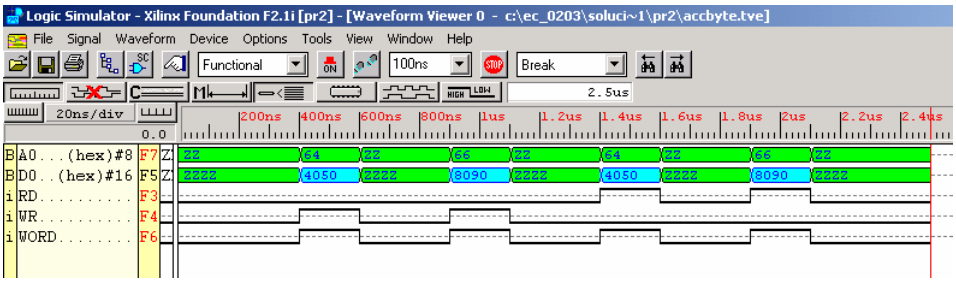


Figura 4. Simulación por Xilinx de un módulo de memoria: dos escrituras de palabras de 16 bits seguidas de dos lecturas en las mismas direcciones.

Asimismo, el alumno también utiliza el simulador (en concreto incluyendo sondas de simulación en los esquemas) para identificar los chips de memoria que son accedidos, dentro del módulo, para unas determinadas direcciones de memoria.

3.3. Practica 3. Mapas de Memoria.

En la tercera práctica sobre diseño de sistemas de memoria, se utilizan módulos de memoria ya preparados de diferente capacidad para implementar un mapa de memoria. En primer lugar se debe calcular *a lápiz* sobre el boletín de prácticas las funciones de selección para un mapa de memoria dado. Una vez calculadas deben implementarse en un módulo de selección. Se realizan dos implementaciones del mapa: una usando puertas lógicas a partir de las funciones de selección calculadas, y otra haciendo uso de un decodificador del tamaño adecuado. En la Figura 5 se puede apreciar el diseño inicial del mapa de memoria.

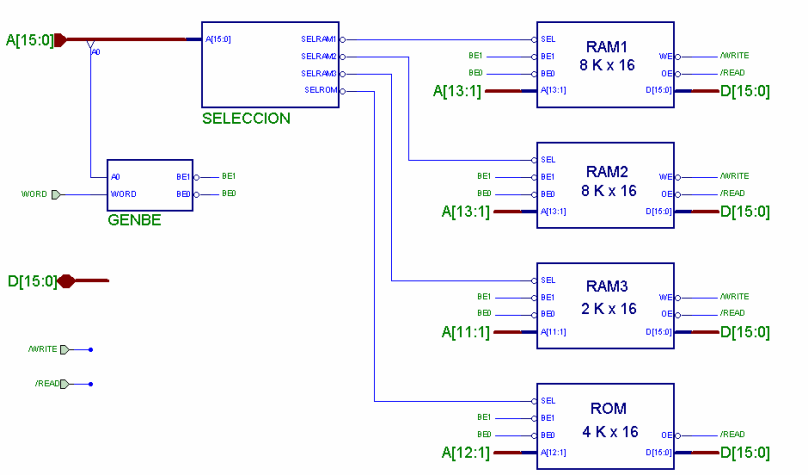


Figura 5. Práctica sobre el diseño de un mapa de memoria.

Para comprobar el circuito, el alumno debe emitir diversos accesos con direcciones diferentes comprobando con el mapa de memoria la correcta selección del módulo adecuado. Los módulos de memoria se entregan ya prediseñados y con un contenido por defecto (utilizando la herramienta LogiBlox de Xilinx). De esta forma es muy sencillo comprobar el funcionamiento del circuito de selección: la lectura en una dirección del módulo *n* siempre debe entregar un valor *n*. En caso contrario, el circuito de selección no es correcto.

4. Conclusiones

En el presente artículo se han descrito las prácticas que sobre el sistema de memoria se han elaborado en las asignaturas de Estructura y Tecnología de Computadores II de la titulación de Ingeniero Técnico en Informática de Sistemas y en Estructura del Computador de la titulación de Ingeniero en Informática, ambas titulaciones de la Universidad Politécnica de Valencia.

Dichas prácticas han sido elaboradas tomando como herramienta de diseño y evaluación la herramienta Xilinx. Los principales motivos para dicha elección han sido:

1. La herramienta permite una gran flexibilidad en el diseño de sistemas con un elevado número de componentes. Así como en el agrupamiento, movimiento de objetos etc.
2. La herramienta es utilizada en asignaturas de la misma titulación, por lo que el alumno amortiza el tiempo invertido para estudiarla.
3. Permite definir jerárquicamente componentes propios.
4. La herramienta permite una gran flexibilidad en la evaluación por medio de simulación, lo que permite una rápida comprobación del correcto funcionamiento de los diseños realizados.
5. Incorpora librerías de chips reales por lo que el alumno trabaja directamente con ellos. Esta faceta, incentiva al alumno a la preparación y realización de las prácticas.
6. Podrá ser utilizada en un futuro cercano en la vida profesional del alumno.
7. Existe una versión para estudiantes disponible libremente.

La realización de las prácticas por niveles en sentido de complejidad creciente reutilizando los objetos que el alumno ha diseñado en prácticas anteriores, por ejemplo, la celda básica de un bit, es un aliciente en la preparación y realización de las prácticas por parte del alumno.

Por último cabe mencionar que las prácticas descritas se han realizado por primera vez durante el primer cuatrimestre del presente curso académico. Los resultados han sido satisfactorios ya que han sido aceptadas de buen grado por parte del alumnado. Muchos de ellos han solicitado una copia de la herramienta para realizar una preparación previa o realizar por iniciativa propia cuestiones optativas y de ampliación de la práctica.

Tal vez el inconveniente más señalado es su complejidad, pero esta desventaja a priori queda paliada por la reutilización de Xilinx en otras asignaturas de la carrera.

Referencias

[1] Computer Assisted Simulation for Circuit Analysis & Design (CASCAD). Reference Manual. Edusoft.

[2] Página Web de Xilinx, http://www.xilinx.com

Nueva aplicación didáctica para electrónica digital

Javier García Zubía
Dpto. de Arquitectura de Computadores
Facultad de Ingeniería
Universidad de Deusto
48007 Bilbao
e-mail: zubia@eside.deusto.es

Resumen

El objetivo del programa BOOLE-DEUSTO es ayudar al profesor y alumno de electrónica digital a analizar y diseñar sistemas digitales básicos. En el presente trabajo se describe la nueva versión de BOOLE-DEUSTO, cuya funcionalidad y utilidad han crecido considerablemente respecto de la primera versión. El interés por BOOLE-DEUSTO se centra en su didáctica, potencia, gratuidad y amplia distribución universitaria.

1. Introducción y originalidad de BOOLE-DEUSTO

El programa BOOLE-DEUSTO existe desde 1.995, desde entonces más de diez personas y miles de líneas de código lo conforman. En el año 2.000 se presentó una primera versión profesional en el IV Congreso de Tecnologías Aplicadas a la Enseñanza celebrado en Barcelona [1], siendo premiado como Mejor Equipo Software TAEE 2.000. BOOLE-DEUSTO destacó por ser didáctico, sencillo, completo, gratuito, fácilmente instalable y adaptado a las necesidades del alumno. También fue presentado en el JENUI 2001, y en [2] y [3], lo que ayudó a divulgarlo. Ahora está disponible la nueva versión 2.0 de BOOLE. El número de mejoras de BOOLE 2.0 y su amplia distribución universitaria, tanto nacional como internacional, nos anima a presentar este trabajo.

Recordemos (ver [2]) que BOOLE-DEUSTO ayuda al alumno y al profesor en la primera fase de aprendizaje de la electrónica digital. Esta fase es descuidada por los entornos profesionales, como OrCAD, Electronic WorkBench, Xilinx, etc., que buscan más el resultado que el proceso, y que son caros y difíciles de manejar e instalar. Estos entornos están orientados al profesional, y no al aula, y parece poco adecuado utilizarlos en un primer momento en la clase. BOOLE-DEUSTO viene a ocupar este hueco, ayudando por igual al profesor y al alumno.

2. Campo de aplicación

BOOLE-DEUSTO es de gran utilidad en un curso básico de electrónica digital y está orientado al análisis y diseño de sistemas combinacionales y secuenciales a nivel de bit.

Centrándonos en los sistemas combinacionales, la figura 1 muestra las opciones disponibles en el programa.

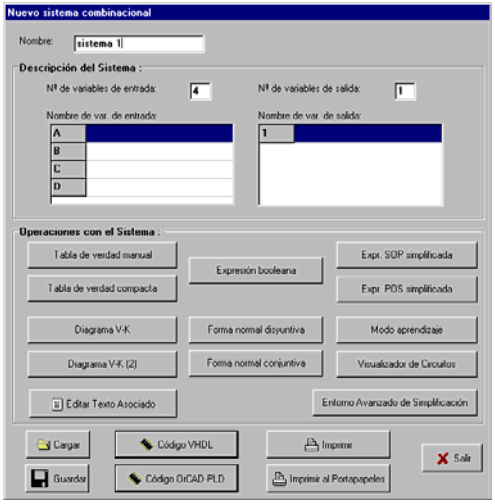


Figura 1. Menú principal de sistemas combinacionales

BOOLE-DEUSTO ofrece utilizar para sistemas combinacionales:

- Expresiones booleanas.
- Tablas de verdad.
- Diagramas de Veitch-Karnaugh.
- Formas normales.
- Expresiones simplificadas o minimizadas.
- Expresiones NAND/NOR.
- Circuitos lógicos: redes AND-OR y OR-AND.
- Simplificador booleano especializado.
- Simplificador booleano para el aprendizaje del alumno.
- Generador de código VHDL, ABEL y OrCAD-PLD.

En cuanto a los sistemas secuenciales o autómatas, BOOLE-DEUSTO ofrece:

- Captura gráfica del diagrama de transición de estados de Mealy o Moore.
- Verificación del autómata.
- Reducción o minimización de estados del autómata.
- Diseño del autómata: generación de las tablas.
- Conversión Moore-Mealy, y viceversa.
- Obtención del circuito lógico.
- Generador de código VHDL, ABEL y OrCAD-PLD.
- Simulación interactiva y batch.

En cuanto al entorno en su conjunto, BOOLE permite:

- Salvar y cargar sistemas digitales.
- Imprimir o guardar resultados.
- Copiar al portapapeles las figuras y tablas.
- Editar un texto para cada sistema.
- Versiones en español e inglés.

3. Estrategia de BOOLE-DEUSTO

Como ya se ha dicho, los entornos profesionales están más interesados en los resultados, que en los métodos y sus pasos, resultando inadecuados para el aula. Muchos profesores han dado soluciones parciales a esta incompatibilidad, pero no conocemos ningún entorno que lo haya hecho globalmente y de forma profesional. BOOLE-DEUSTO quiere ser una herramienta fundamental en el desarrollo de un curso básico de electrónica digital, tecnología de computadores, etc.

Otra característica del BOOLE, y que pocos entornos tienen o buscan, es que en ningún momento marca el método a seguir para diseñar un sistema. BOOLE simplemente ofrece los métodos (ver punto 2), y es el alumno/usuario el encargado de elegirlos, aplicarlos y observar los resultados obtenidos. BOOLE es un entorno no dirigido, que bien podría tomar el nombre de calculadora booleana. Quizá éste sea uno de los aspectos más destacados por cuantos profesores lo usan y recomiendan.

Por último, BOOLE presenta dos aspectos estratégicos que son reclamados por el mundo de la enseñanza. Uno es que permite al profesor/usuario generar, resolver y documentar ejercicios de forma rápida. Y dos, BOOLE se distribuye como un único fichero .exe, sin instalación, sin licencias, sin Internet, sin requisitos hardware, etc.; cuando el profesor lo entrega o distribuye sabe que nunca tendrá que resolver problemas de instalación.

4. Nueva versión de BOOLE-DEUSTO

Un buen número de profesores de España, Europa y América, más de cien, ya utilizan en clase y entregan a sus alumnos una copia del BOOLE. En este apartado se describirán las novedades que presenta BOOLE en la versión 2.0. Esta descripción se centrará en los sistemas combinacionales y secuenciales por separado.

4.1. Sistemas combinacionales

Como primera novedad para sistemas combinacionales, BOOLE permite su captura mediante una expresión booleana cualquiera. El procesador de expresiones booleanas desarrollado permite introducir cualquier expresión sin restricciones de ningún tipo, admitiendo los operadores NOT, AND, OR y XOR en cualquier orden. El procesador está validado y ha sido diseñado aplicando técnicas de compiladores. En la figura 2 vemos un ejemplo.

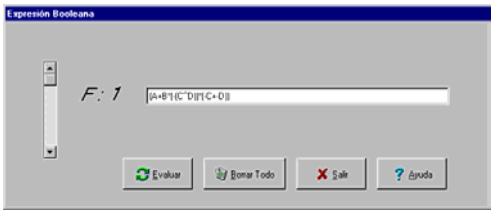


Figura 2. Expresión booleana.

Una vez evaluada la expresión, el BOOLE crea el sistema correspondiente. El usuario puede verlo, transformarlo, operarlo, etc, todo ello como si manejara una calculadora. La figura 3 muestra el diagrama V-K de la expresión introducida.

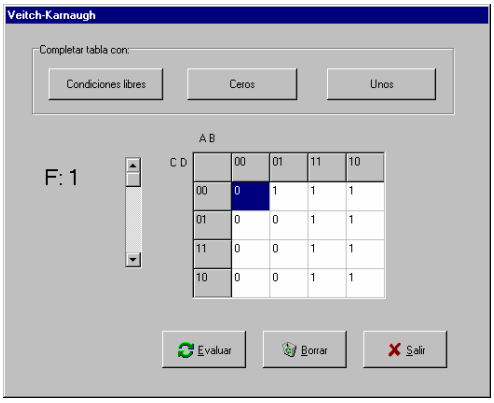


Figura 3. Diagrama V-K de la expresión capturada

Podemos ver otra novedad en la parte inferior derecha de la figura 1. En ella BOOLE ofrece el entorno avanzado de simplificación. El REDUCTIO, desarrollado en su totalidad por Borja Sotomayor, simplifica una función booleana mediante seis métodos distintos, heurísticos y no heurísticos, lo que permite al usuario comparar la bondad de cada uno de ellos.

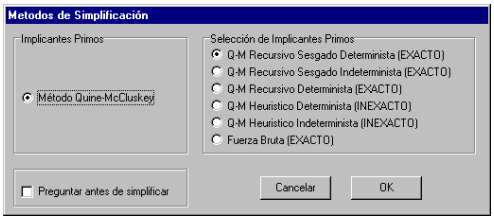


Figura 4. Métodos de simplificación de REDUCTIO

Por último, y dentro de los sistemas combinacionales, aparece como novedad la posibilidad de generar automáticamente el código VHDL del sistema creado. De esta manera podemos conectar BOOLE con programas profesionales, y alejarle de ser considerado un programa sin aplicación práctica. A continuación se muestra el código VHDL de un sistema combinacional.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity sistema 1 is
   Port (
        P5: in std_logic;
        P10: in std_logic;
        P25: in std_logic;
        P50: in std_logic;
        E1: out std_logic;
        E2: out std_logic;
        E5: out std_logic;
        E10: out std_logic;
        E20: out std_logic
        );
end sistema 1;

architecture behavioral of sistema is
begin
   E1<=((P10) or (P5))
   E2<=((P5))
   E5<=((P25) or (P10))
   E10<=((P50) or (P25))
   E20<=((P50))
end behavioral;
```

Figura 5. Código VHDL de un sistema combinacional

Esta planificado añadir el lenguaje ABEL a los VHDL y OrCAD-PLD ya existentes.

4.2. Diagramas de V-K

La primera versión de BOOLE ya hacía hincapié en el uso de los diagramas de Veitch-Karnaugh, de manera que el alumno podía cargar, y luego simplificar, cualquier función booleana mediante diagramas V-K, acercándose al modo de trabajar en clase.

En el nuevo BOOLE los V-K siguen siendo protagonistas. Veamos, el alumno introduce una función booleana mediante el V-K y luego la simplifica para ver el resultado, la novedad se centra en que BOOLE dibuja cada uno de los lazos de la función simplifcada.

En la figura 6 se puede ver un V-K y los lazos correspondientes a su simplificación.

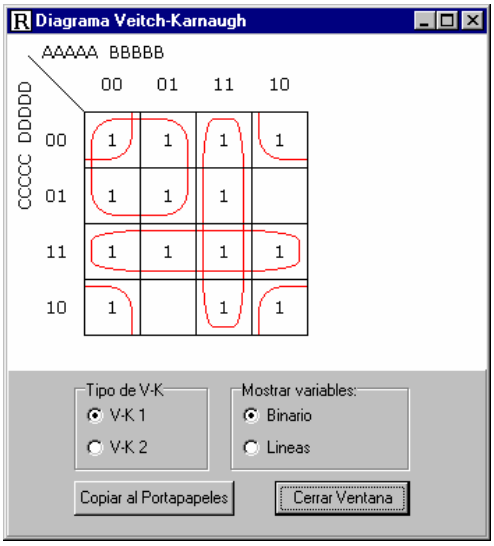


Figura 6. Visualización de lazos en el V-K

Esta habilidad del BOOLE vuelve a mostrarle como un entorno preocupado por las necesidades del alumno y del profesor, sobre todo en lo que concierne a los métodos y sus pasos. Aspectos como éste, y otros del BOOLE, quedan fuera del enfoque de entornos profesionales.

Actualmente se trabaja para que BOOLE también permita que el alumno introduzca mediante lazos la expresión simplificada, "pintando los lazos", sin escribir obligatoriamente la función algebraica correspondiente.

4.3. Sistemas secuenciales: Autómatas

El grueso de las novedades se concentra en los autómatas. En primer lugar BOOLE transforma un autómata de Moore en el correspondiente de Mealy, y viceversa. BOOLE redibuja el autómata y asigna los nuevos nombres a los estados. Las figuras 7 y 8 muestran el autómata de Mealy original y el equivalente de Moore.

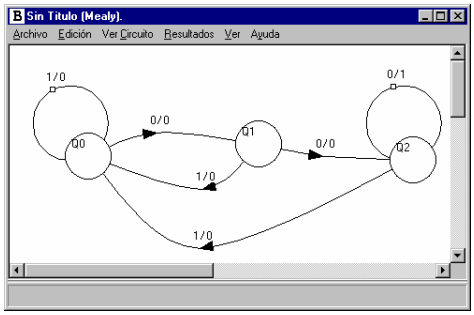


Figura 7. Autómata de Mealy original

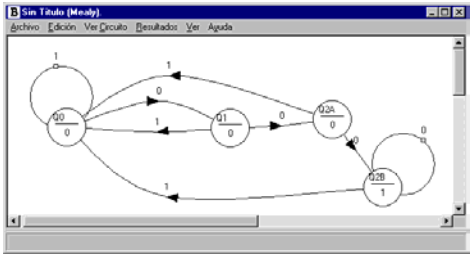


Figura 8. Autómata de Mealy equivalente

En el nuevo BOOLE se ofrece la posibilidad de reducir un autómata, ya sea de Moore o de Mealy. El alumno introduce un autómata y el BOOLE obtiene y redibuja aquel que con menos estados es equivalente al original. Las figuras 9 y 10 muestran un autómata original con cinco estados y el correspondiente reducido con cuatro estados, respectivamente.

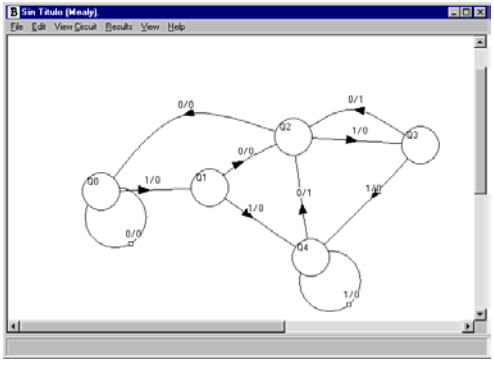


Figura 9. Autómata original de Mealy

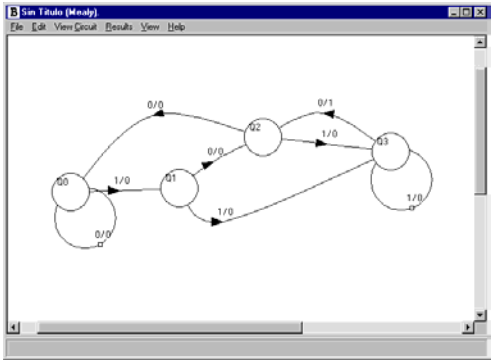


Figura 10. Autómata de Mealy reducido

Todo el proceso de reducción de estados es autómático, incluido el redibujar el nuevo autómata. Además BOOLE ofrece al alumno ver la tabla de minimización paso a paso, y así comprobar el método aplicado. En la figura 11 se puede ver la tabla y otras informaciones adicionales.

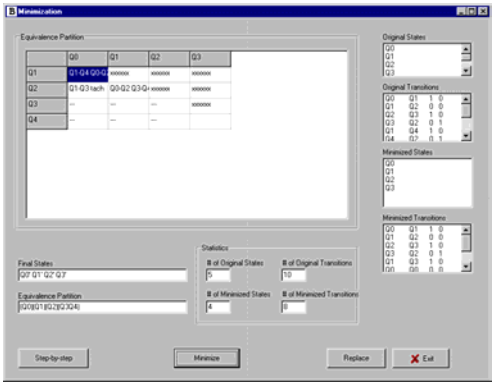


Figura 11. Autómata de Mealy reducido

Por último, en la minimización de estados BOOLE permite procesar autómatas incompletamente especificados, tanto en sus salidas como en las transiciones. El algoritmo implementado es de cierta complejidad y dota a BOOLE de una potencia de la que carecen incluso algunos entornos profesionales (lo mismo que para la simplificación booleana). Las figuras 12 y 13 muestran un autómata de Moore incompleto y el correspondiente minimizado.

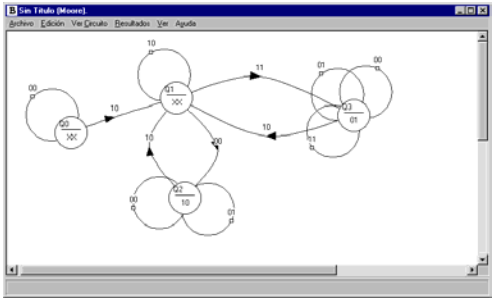


Figura 12. Autómata original de Moore

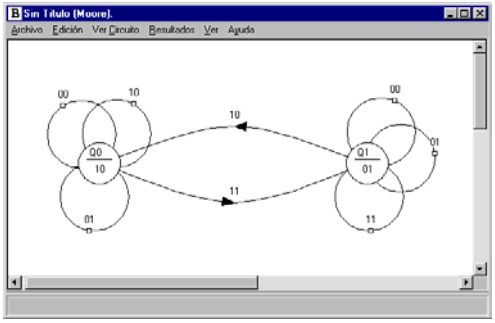


Figura 13. Autómata de Moore reducido

Una vez capturado y reducido el autómata podemos visualizar sus tablas de diseño y su circuito lógico. Las figuras 14 y 15 muestran las tablas de resultados y el circuito lógico del autómata de la figura 13, respectivamente.

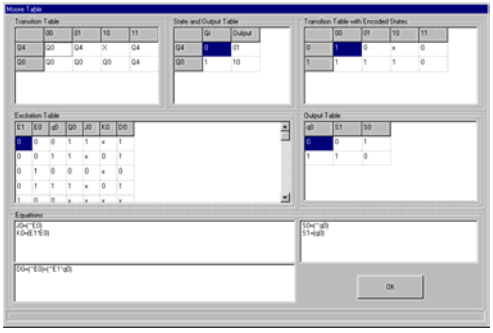


Figura 14. Diseño del autómata de Moore

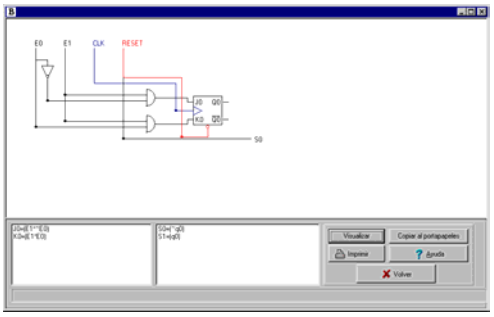


Figura 15. Circuito del autómata de Moore

También se puede obtener el código VHDL del autómata cargado, permitiendo la conexión de BOOLE con cualquier entorno profesional del tipo Xilinx ISE o Foundation, MaxPlus II, OrCAD, ispDESIGN de Lattice, etc. A continuación se lista el código VHDL.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity Sistema is
       Port (
              inicio: in std_logic;
              ck: in std_logic;
              E0: in std_logic;
              E1: in std_logic;
              S0: out std_logic;
              S1: out std_logic
              );
end Sin_Titulo;
architecture behavioral of Sistemais
type nombres_estados is (Q1, Q0);
signal estado: nombres_estados;
signal entrada_aux: std_logic_vector
(1 downto 0);
begin
entrada_aux<=E0&E1
process(inicio, ck)
begin
if inicio='1' then
estado<=Q1;
elsif ck='1' and ck'event then
case estado is
when Q1 =>
case entrada_aux is
when "00" => estado<=Q1;
when "01" => estado<=Q1;
when "10" => estado<=Q0;
when "11" => estado<=Q1;
when others => estado<=Q1;
end case;
when Q0 =>
case entrada_aux is
when "00" => estado<=Q0;
when "01" => estado<=Q0;
when "10" => estado<=Q0;
when "11" => estado<=Q1;
when others => estado<=Q1;
end case;
when others => estado<=Q1;
end case;
end if;
end process;
process(estado)
begin
case estado is
when Q1 =>
S0<='0';
S1<='1';
when Q0 =>
S0<='1';
S1<='0';
end case;
end process;
end behavioral;
```

4.4. Sistemas secuenciales: Simulación

Otro aspecto en que BOOLE ha mejorado considerablemente es en la simulación del autómata capturado. Al simular, el profesor muestra en clase la evolución del sistema diseñado, aclarando conceptos y situaciones. Por otra parte, cuando el alumno simula está comprobando que lo diseñado cumple las especificaciones, es más, está justificando frente al profesor la idoneidad de lo creado. La importancia de ambas situaciones nos ha llevado a diseñar cuidadosamente, aunque no muy espectacularmente, la simulación de un autómata. Hemos dado prioridad a dos aspectos: agilidad y diversidad. En el menú, BOOLE ofrece simular de forma interactiva, batch, rápida o detallada. Veamos algunas características de cada una de ellas.

En la simulación interactiva, el alumno va introduciendo entradas *sobre la marcha*, y simultáneamente ve los resultados: on-line. Sin embargo en batch, el alumno/profesor primero prepara la secuencia de entrada con la que va excitar al autómata y luego la introduce al autómata. Elegir una u otra opción depende de cada aplicación y del carácter del usuario. En la figura 16 vemos la pantalla de simulación interactiva, en ella podemos ver que un estado está marcado con un círculo: es el estado actual de la simulación.

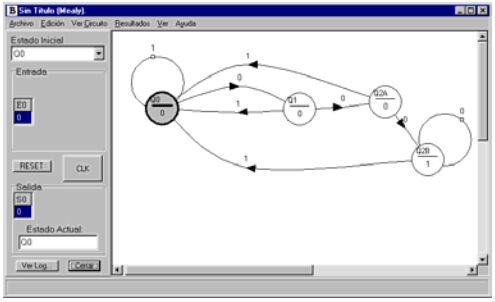
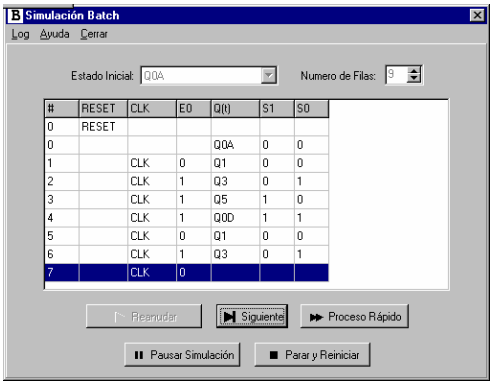


Figura 16. Simulación interactiva

La figura 17 muestra cómo se introduce y simula una secuencia de entrada en modo batch.



#	RESET	CLK	E0	Q(t)	S1	S0
0	RESET					
0				Q0A	0	0
1		CLK	0	Q1	0	0
2		CLK	1	Q3	0	1
3		CLK	1	Q5	1	0
4		CLK	1	Q0D	1	1
5		CLK	0	Q1	0	0
6		CLK	1	Q3	0	1
7		CLK	0			

Figura 17. Simulación batch

La diferencia entre simulación rápida o detallada estriba en que en la primera el usuario sólo puede cambiar las entradas en los flancos, es decir, un flanco una entrada. Mientras que en la detallada el usuario puede cambiar la entrada entre flancos, y, por ejemplo, de esta manera podrá apreciar la diferencia entre Moore y Mealy. La más común es la rápida, pero la detallada puede ser la idónea para que el profesor explique ciertos casos particulares. La figura 18 muestra el aspecto de una simulación interactiva detallada.

En cualquier caso, tanto si la simulación es batch, interactiva, rápida o detallada, el BOOLE ofrece una tabla con los resultados para que el alumno pueda observarlos con detalle, o imprimirlos, o guardarlos o cargarlos, aportando comodidad y potencia al trabajo de clase. La figura 19 muestra la tabla-resultado de una simulación.

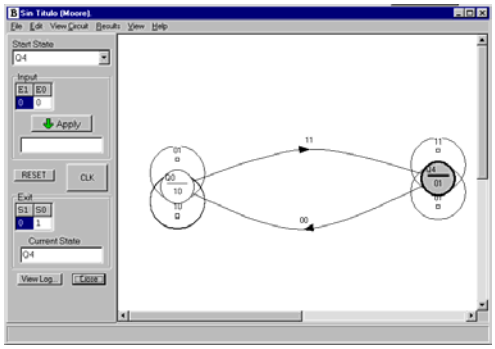
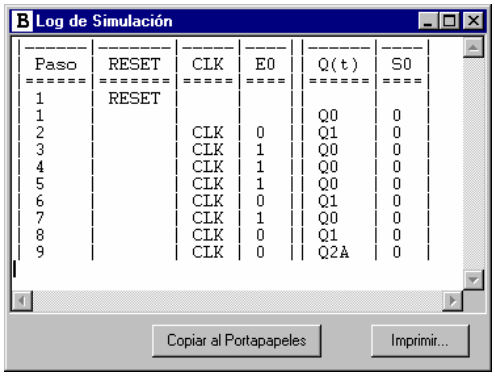


Figura 18. Simulación batch



Paso	RESET	CLK	E0	Q(t)	S0
1	RESET				
1				Q0	0
2		CLK	0	Q1	0
3		CLK	1	Q0	0
4		CLK	1	Q0	0
5		CLK	1	Q0	0
6		CLK	0	Q1	0
7		CLK	1	Q0	0
8		CLK	0	Q1	0
9		CLK	0	Q2A	0

Figura 19. Log de una simulación

5. Experiencia en el aula

Las primeras versiones de BOOLE son del año 1.995, y ya desde 1.996 empezamos a usarlo en el aula, y sobre todo se lo dimos a los alumnos para que lo usaran en casa como una calculadora booleana que les ayudaba a corregir los ejercicios, y a documentar los hechos por ellos.

En cuanto al uso en el aula: el profesor enseña un método, hace algún ejemplo y manda unos ejercicios. El alumno los resuelve en casa ayudándose del BOOLE para corregirlos y documentarlos y/o para repasar los métodos aplicados. El peligro de usar BOOLE es claro: puede que el alumno no haga los ejercicios, sino que directamente use el programa; igual que cuando aprendíamos a dividir y teníamos una calculadora. Es más una cuestión del alumno que del profesor el buen uso que se haga del BOOLE.

En cuanto al campo de aplicación del BOOLE, este es amplio. BOOLE implementa con todo detalle, paso a paso, todos los métodos de manipulación booleana, las técnicas de diseño combinacional con tabla de verdad y el diseño de autómatas, quedando fuera del BOOLE, por falta de una metodología sistemática, el diseño de sistemas combinacionales y secuenciales con funciones estándar. En términos generales el BOOLE ayuda en al menos el 50% de un curso básico de electrónica digital.

Otra cuestión es, ¿cuánto se tarda en dominar el BOOLE? Nuestra experiencia nos dice que basta con una hora escasa para enseñar su mecanismo. Resumiendo, con poco esfuerzo se cubre buena parte de la asignatura.

Desde 1996 BOOLE-DEUSTO ha sido utilizado por unos 700 alumnos al año, tanto en Ingeniería en Informática como en Ingeniería Industrial; los primeros en la asignatura de Tecnología de Computadores, y los segundos en Electrónica Digital. Además, y como ya se ha dicho, BOOLE es utilizado en varias universidades y centros españoles, europeos y americanos, y acompaña, como software de apoyo, al libro [4].

Nuestra experiencia con BOOLE es muy positiva, y así nos lo dicen los alumnos. Ellos destacan lo idóneo del programa y lo fácil de usar. Ambos comentarios, y otros, quedan reforzados con lo que nos hacen saber profesores y alumnos de otros centros. Esto último, a nuestro parecer, es la mejor prueba de que BOOLE no es algo específico de esta universidad, sino que es claramente exportable. Esta *exportabilidad* debería ser el primer objetivo de cualquier sw con ánimo de difusión.

Queda pendiente, una vez que BOOLE esté en su versión definitiva (curso 2002-2003), una evaluación sistemática de su calidad con encuestas, grupos, etc.

6. Conclusiones

Con las novedades comentadas, BOOLE-DEUSTO refuerza sus características iniciales: análisis y diseño de sistemas digitales, orientado al aula, didáctico, ágil, potente, profesional, completo, transportable, gratuito y atractivo.

Referencias

[1] García Zubía, J. y Sanz Martínez, J. "Adecuación de los entornos computacionales a la clase de electrónica digital: BOOLE-DEUSTO". *Actas del IV Congreso de Tecnologías Aplicadas a la Enseñanza de la Electrónica*,Barcelona, 2.000.

[2] García Zubía, J. "Metodología y necesidades computacionales en sistemas digitales". *Actas de las VII Jornadas de Enseñanza Universitaria de la Informática*, Palma de Mallorca, 2001.

[3] García Zubía, J., Sanz Martínez, J. y Sotomayor, B. "BOOLE-DEUSTO, la aplicación para sistemas digitales". *Actas de las VII Jornadas de Enseñanza Universitaria de la Informática*, Palma de Mallorca, 2001.

[4] Angulo Usategui, J.M. y García Zubía, J. *Sistemas digitales y Tecnología de computadores*, Ed. Paraninfo, 2.001.

Optimización de una implementación JPEG teniendo en cuenta la arquitectura actual de los procesadores

J. A. Padilla
Logic Factory
http://www.logic-factory.com

M. Anguita, F. J. Fernández, A. F. Díaz, A. Cañas, A. Prieto
Dpto. de Arquitectura y Tecnología de Computadores
Universidad de Granada
18071 Granada
e-mail: manguita@atc.ugr.es

Resumen

Creemos conveniente proponer trabajos fin de carrera con el objetivo de que los estudiantes comprueben que pueden *mejorar* ostensiblemente las prestaciones de sus programas en *poco tiempo* aplicando los conocimientos sobre *arquitectura* de computadores que han ido adquiriendo a lo largo de la titulación. Mostrando los resultados obtenidos en estos trabajos a estudiantes de diferentes cursos de la titulación, pretendemos incrementar su *motivación* en las materias de arquitectura. Aquí se muestra el incremento en prestaciones obtenido, aprovechando la arquitectura actual de los computadores, en una implementación de descodificador *JPEG*. Para la mejora de prestaciones, se propone seguir un *proceso iterativo* de varios pasos.

1. Introducción

La *motivación* es uno de los aspectos que más influye en el proceso de aprendizaje, de hecho un estudiante motivado se aproxima al máximo de sus posibilidades. En [12] se define la motivación como "aquellas condiciones o estados del individuo que le activan o dan energía para llevar a cabo una conducta dirigida hacia determinados objetivos, impulsándole a superar los obstáculos que se le presentan". Para favorecer la motivación es esencial proporcionar un *sentido de utilidad* a los contenidos que se imparten. Los estudiantes que prevén dedicar su vida laboral a la programación de aplicaciones, a menudo, no se enfrentan al estudio de la arquitectura y estructura de computadores suficientemente motivados, lo que les puede llevar a fracasar en estas materias. Para incrementar la motivación de los estudiantes, se les podría demostrar, que pueden mejorar las prestaciones de sus programas aplicando los conocimientos de arquitectura de computadores que se adquieren en estas materias. Con el fin de ser más convincentes, deberían obtener las mejoras ellos mismos, o se les debería mostrar resultados que estimen que pueden obtener ellos mismos, como por ejemplo resultados logrados por otros estudiantes de la propia titulación.

Por ello creemos conveniente proponer trabajos fin de carrera, en los que los estudiantes comprueben que pueden *mejorar* en gran medida las prestaciones de sus programas en *poco tiempo* aplicando los conocimientos sobre arquitectura y estructura de computadores que han adquirido. Así, los estudiantes afianzan estos conocimientos, se favorece su utilización posterior, y además obtenemos resultados que se pueden mostrar a otros estudiantes, como se comentó más arriba.

Por otra parte, incrementar las prestaciones de un programa atendiendo a la arquitectura, no sólo resulta actualmente de interés para aplicaciones embebidas, también para aplicaciones ejecutadas en el entorno multitarea de un sistema de propósito general. Las *aplicaciones embebidas* (en impresoras, fotocopiadoras, DVD, cámaras de fotografía o vídeo digital, asistentes personales, teléfonos móviles, etc) requieren altas prestaciones, en muchos casos tiempo real, y están sometidas a otras restricciones como: consumo de potencia, tamaño o precio. Estas restricciones hacen que se opte para estas aplicaciones por procesadores de bajo precio, baja frecuencia de reloj, y también por ejemplo por ajustar el tamaño

y precio de la memoria. Por tanto, para conseguir las prestaciones necesarias para las aplicaciones embebidas, dadas estas restricciones, el programador tiene que *extraer el máximo provecho de la arquitectura*. Los fabricantes de procesadores de propósito general, suelen ofrecer versiones de sus procesadores con consumos de potencia reducidos, para su utilización en aplicaciones incrustadas (Ultrasparc de Sun, PowerPC de IBM-Motorola-Apple, Pentium de Intel, etc).

Resulta también de interés mejorar prestaciones para aplicaciones ejecutadas en el entorno multitarea de un PC u otro *sistema de propósito general*, donde el usuario desea ejecutar varios programas simultáneamente y con buenas prestaciones. Además, para la ejecución de aplicaciones multimedia en estos entornos, evitando en lo posible el uso de hardware específico, se ha de explotar al máximo la arquitectura ([3], [6], [13], [14]).

Adicionalmente, hay que tener en cuenta, que las *prestaciones* y el *tiempo de desarrollo del software* son esenciales, entre otras cosas, debido a la competencia entre empresas, y que aprovechando los conocimientos de arquitectura que ya se tienen, se puede conseguir una buena relación prestaciones-tiempo de desarrollo. Para mejorar adicionalmente esta relación, el programador puede utilizar el software de ayuda a la programación y optimización que proporcionan los fabricantes para sus arquitecturas. Software tradicionalmente ofrecido por los fabricantes de DSP (*Digital Signal Processor*), y que ahora se ha extendido a procesadores de propósito general, especialmente a raíz de la incorporación de unidades SIMD (MMX, SSE y SSE2 de Intel, VIS de Sun, 3Dnow! de AMD o Altivec para PowerPC), por ejemplo el paquete Vtune y los compiladores de Intel [17], CodeAnalyst de AMD [1], o VSDK de Sun [16].

Para mejorar prestaciones en una aplicación, se pueden utilizar los conocimientos adquiridos sobre la *arquitectura abstracta* del procesador o ISA (*Instruction Set Architecture*): repertorio de instrucciones, conjunto de registros y su gestión (pila o acceso directo), tipos de datos y modos de direccionamiento; y sobre la *arquitectura concreta*: arquitectura segmentada, superescalar, VLIW, SIMD, jerarquía de memoria. También se puede aplicar los conocimientos sobre las optimizaciones que actualmente pueden realizar (y las que no son capaces de realizar) los compiladores ([9], [18], [2], [10]).

En estos trabajos fin de carrera, se propone al estudiante la implementación completa de alguna aplicación, se eligen aplicaciones que puedan atraer al estudiante por su amplia utilización (se están desarrollando trabajos para jpeg, mpeg-2 y mp3), o por su interés en investigación (actualmente se está desarrollado un trabajo sobre procesamiento de flujo óptico para evaluación de movimiento).

En la Figura 1 se muestra el proceso que se puede seguir para obtener la versión optimizada de la implementación, objetivo del trabajo:

1. *Programar aplicación.* Partiendo de una descripción detallada de la aplicación, se pretende que el estudiante desarrolle un programa tal y como lo haría habitualmente. Simultáneamente buscaría entradas al programa representativas que le permitan comprobar que éste funciona correctamente, y que serán utilizadas posteriormente para analizar las prestaciones del código.
2. Incluir en puntos del código *medidas de prestaciones,* con el fin de analizar el tiempo de ejecución que suponen los diferentes bloques funcionales de la aplicación.
3. *Ejecutar el programa* con el conjunto de entradas seleccionado y *recolectar*, el tiempo de ejecución total y los tiempos obtenidos para cada bloque funcional.
4. Si se ha llegado a las *prestaciones* que se tenían como *objetivo* o se ha agotado el *tiempo de desarrollo propuesto* termina el proceso.
5. *Localizar el bloque o bloques funcionales* que destaquen por su *mayor tiempo* de ejecución.
6. *Analizar* la implementación de los bloques seleccionados en el paso 5, y *modificar* esta implementación, teniendo en cuanta la arquitectura, con el objetivo de *mejorar* prestaciones. La posibilidad que ofrecen los compiladores de generar el código ensamblador de un programa, se utiliza en esta etapa para ver aquellos aspectos de la arquitectura concreta o aquellas instrucciones del repertorio máquina que el compilador no ha utilizado en estos bloques, con el fin de que sean aprovechados explícitamente por el programador. A continuación, se procedería nuevamente a recolectar tiempos, es decir se vuelve al punto 3.

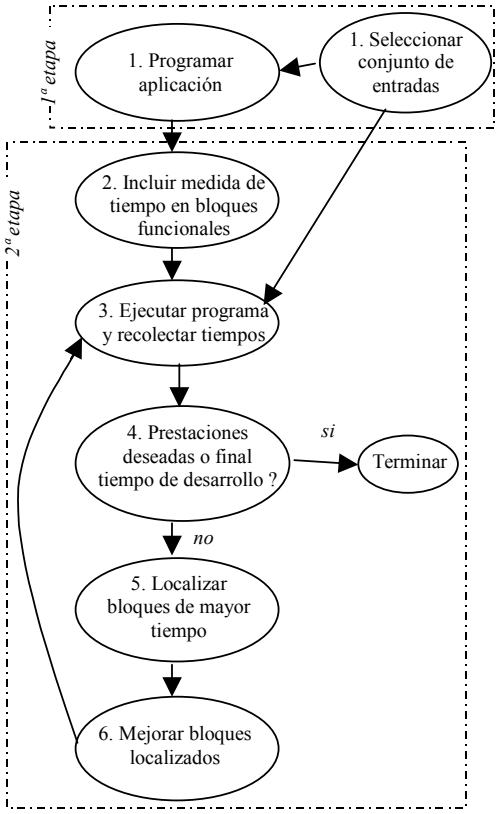


Figura 1. Proceso que se puede seguir para optimizar la implementación.

En el paso 6, para mejorar prestaciones en cada bloque funcional atendiendo a la arquitectura, se pueden abordar los siguientes puntos:

- Mejorar la construcción de los bucles. El conocimiento de la aplicación nos puede llevar a realizar construcciones de bucles más eficientes aplicando por ejemplo un "desenrollado de bucles" ([9], [18]) más apropiado o realizando una reordenación de instrucciones dentro del bucle para optimizar la utilización de las unidades funcionales.
- Utilización de instrucciones del procesador apropiadas a las operaciones realizadas en el programa y que el compilador no haya generado. Hay varios motivos por los que los compiladores no generan instrucciones que tiene el procesador. Puede por ejemplo, que el compilador no detecte todas las situaciones en las que resulta útil la instrucción, o puede que la desconozca por tratarse de una instrucción incluida en un procesador de reciente aparición. Por ejemplo, resulta complicado para los compiladores generar instrucciones SIMD, incorporadas en los noventa en DSPs (TMS320C80 de Texas Instruments en 1992) y procesadores de propósito general (MAX en PA de HP en 1993; VIS en Sparc de Sun en 1995; MMX en x86 de Intel en 1996). En [3], [6], [13] y [14] se pueden ver ejemplos del incremento en prestaciones obtenido en aplicaciones multimedia aprovechando principalmente la arquitectura SIMD.
- Analizar la penalización por predicción errónea en saltos. Se podría optar por cambiar el sentido de algún salto, o por sustituir construcciones if-then-else por instrucciones de movimiento condicional (*cmov*), si el compilador no las ha generado.
- Analizar la penalización por falta de alineación en los datos. Habrá que alinear los datos explícitamente en caso de ser necesario.
- Optimizar el acceso a memoria. El conocimiento de la aplicación nos permite detectar fácilmente si es eficiente almacenar los datos en memoria principal saltando las caches, y si resulta rentable precaptar (con la suficiente antelación) los datos con los que se va a operar.

Un análisis detallado sobre las optimizaciones que se pueden realizar para procesadores x86, se encuentra en [2] y [10].

Aquí se muestra como ejemplo, la mejora de una implementación de descodificador JPEG con pérdida "*baseline"* teniendo en cuenta la arquitectura. En la Sección 2 se describe brevemente la aplicación JPEG. La Sección 3 muestra las mejoras que se han ido obteniendo aplicando el proceso de la Figura 1. Por último, la Sección 5 muestra algunas conclusiones.

2. Descripción del descodificador JPEG

JPEG es un estándar internacional (1992) ISO (*International Organization for Standardization*) y ITU (*International Telecommunication Union*) para la compresión de imágenes [11]. Se utiliza con el fin de disminuir espacio de almacenamiento o tiempo de transmisión, en aplicaciones que

necesitan almacenar o transferir imágenes. Un descodificador JPEG, a partir de los datos de una imagen comprimida y de unas especificaciones de tablas (de cuantificación, Huffman), genera como salida la imagen reconstruida correspondiente. En la Figura 2 se pueden ver los bloques funcionales del descodificador JPEG: preprocesamiento, descodificador de entropía, descuantificación, IDCT (*Inverse Discrete Cosine Transform*) y postprocesamiento.

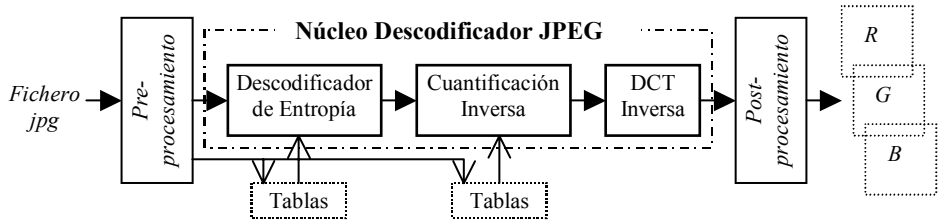


Figura 2. Bloques funcionales del descodificador JPEG

El núcleo del descodificador (entropía, descuantificación, IDCT) se aplica a nivel de bloques de 8x8 píxeles de la imagen. La imagen se irá formando a partir de los bloques 8x8 reconstruidos por este núcleo. En la práctica, una imagen a color se representa por varias componentes. Por ejemplo, en los computadores se utilizan imágenes representadas típicamente en formato RGB, que tiene tres componentes: rojo, verde y azul. JPEG, en lugar de RGB, prefiere trabajar con imágenes en color en formato YCbCr, porque mejora la razón de compresión (ver [11] y Capítulo 5 en [5]). Cada componente se codifica por separado, aplicando el proceso a bloques de 8x8. Para una imagen a color, el descodificador irá reconstruyendo cada componente de color a partir de los bloques 8x8 que va descodificando. A continuación se comenta brevemente cada etapa de procesamiento de la figura, una descripción detallada se encuentra en [5] y [11]:

1. *Preprocesamiento.* Esta etapa ha de conocer el formato de los ficheros JPEG. Debe extraer del fichero, además de las secuencias de bits que representa los bloques de imagen 8x8 comprimidos, información necesaria para el proceso de descodificación, como el tamaño de la imagen (o de las diferentes componentes de que consta la imagen) y las especificaciones de tablas utilizadas en el descodificador de entropía y descuantificador.
2. *Descodificador de entropía*. Deshace la codificación por longitud de ráfagas y codificación Huffman realizada por el codificador. Genera un bloque de 8x8 datos.
3. *Descuantificación*. Multiplica cada dato del bloque 8x8 generado en la etapa anterior, por un valor extraído de la tablas de descuantificación.
4. *IDCT*. Genera un bloque 8x8 (s_{yx}) aplicando al bloque 8x8 de entrada (S_{vu}) la transformada del coseno inversa:

$$s_{yx} = \frac{1}{4}\sum_{u=0}^{7}\sum_{v=0}^{7} C_u C_v S_{vu} \cos\frac{(2x+1)u\pi}{16}\cos\frac{(2y+1)v\pi}{16}$$

$$C_u, C_v = \begin{cases} 1/\sqrt{2} & u,v = 0 \\ 1 & u,v \neq 0 \end{cases} \qquad y,x = 0,1,...,7 \qquad (1)$$

5. *Postprocesamiento*. Ya que JPEG prefiere la codificación YCbCr y el computador el formato RGB, típicamente se debería realizar en esta etapa una conversión de formato YCbCr a RGB.

3. Mejora JPEG

En el proceso de mejora se han seguido los pasos de la Figura 1. La implementación de la aplicación se ha realizado en gcc, utilizándose en la compilación la última versión de djgpp v2.03 [8], con gcc 3.1 que incluye optimizaciones para Pentium. Estas optimizaciones se aplican si se compila con la opción *–mcpu=pentium*. Se han utilizado también las opciones de optimización *–O6* y *–fast-math*, recomendadas en la FAQ [8], y las opciones *–finline* y *–finline-function*. Para probar el funcionamiento del programa y para las posteriores medidas de prestaciones, se han buscado imágenes JPEG de distintos tamaños. Las medidas de prestaciones presentadas en esta

sección, se han generado con las tres imágenes que se pueden ver en las Figuras 3, 4 y 5, en un procesador Pentium de la familia P6 a 750 MHz con cache de nivel 2 de 256KB.

Figura 3. Sensation, de 651x507=330.057 píxeles

Figura 4. Gladiator, de 2044x1603=3.276.532 píxeles

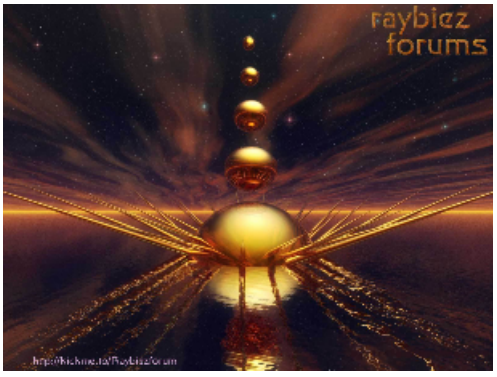


Figura 5. Rabiez, de 4800x3600=17.280.000 píxeles

El *paso 1* (Figura 1) acaba una vez que el programa generado, siguiendo el método habitual del estudiante, funciona correctamente. En el programa concreto implementado, la entrada es un fichero en formato jpg y la salida es un fichero (.raw) con la imagen RGB correspondiente a la imagen de entrada comprimida ([15]). En la transformada del coseno inversa y en la conversión a formato RGB se opera con datos punto flotante de simple precisión (32 bits), en la cuantificación se opera con datos enteros de 16 bits.

A continuación, en el *paso 2*, se incorporan en el código medidas que permiten determinar el tiempo que suponen los diferentes bloques funcionales de la aplicación (Figura 2): pre-procesamiento, descodificador de entropía, descuantificación, IDCT y postprocesamiento. Concretamente, se mide para cada bloque funcional y para el total del programa, el número de *ciclos de reloj del procesador* que consume su ejecución. Este número de ciclos se obtiene utilizando la instrucción ensamblador *rdtsc*, que devuelve el número de ciclos transcurridos desde que se reinició el sistema. Así se obtiene una precisión a nivel de ciclo de reloj, y las medidas son independientes de la frecuencia de reloj del procesador.

En las Tablas 1, 2 y 3, se muestran los ciclos de reloj generados a partir distintas versiones del programa ejecutable JPEG. Las versiones etiquetadas *0* y *1*, se han obtenido con el programa fuente inicial generado en la 1ª etapa (Figura 1). En la *versión 0* no se ha usado ninguna optimización del compilador, para *1* se han utilizado las opciones del compilador indicadas más arriba. Estas opciones se han empleado también para el resto de versiones, que han sido generadas a partir de los fuentes derivados en las sucesivas iteraciones del ciclo de la Figura 1.

La *versión* 1 del ejecutable presenta una ganancia en prestaciones de aproximadamente 1.36 frente a la versión 0. Para facilitar la comparación entre las diferentes versiones de código y entre los diferentes bloques funcionales en cada versión, se muestra en la Figura 6 la ganancia en prestaciones obtenida con las versiones 2, 3 y 4 sobre la versión inicial no mejorada 1, y en la Figura 7 un gráfico de barras apiladas para una de las imágenes. Destacar que en la etapa de postprocesamiento se incluye no sólo la conversión de formato YCbCr a formato RGB, sino el tiempo que supone almacenar la imagen RGB en un fichero. En las tablas se indica entre paréntesis para la versión 4, el tiempo que supone exclusivamente la conversión a RGB.

En la *primera iteración* del ciclo de la Figura 1, se genera la *versión 2*. En el *paso 3*, se obtienen

los ciclos de reloj para la *versión 1*. Como se puede observar en las Tablas 1, 2 y 3, y en la Figura 7, para la versión 1, destaca especialmente el tiempo que supone la IDCT. El código de la IDCT implementa la ecuación (1) mediante dos bucles. Analizando el código de la IDCT en el *punto 6* encontramos, que este bucle realiza además de una conversión de entero a punto flotante, cuatro multiplicaciones y una suma punto flotante en secuencia. Dado la gestión como pila de los registros punto flotante, habrá dependencia de datos entre una instrucción punto flotante y la siguiente, lo que impide que se puedan ejecutar solapadas. Para mejorar el código, se ha eliminado de los bucles la multiplicación de Cu y Cv. Esta reducción de las operaciones punto flotante con dependencias, ha mejorado sustancialmente las prestaciones, como ilustra la entrada para la *versión 2* (IDCT,Total) de las tablas y la Figura 7.

Versión	PreProc.	Entropía	Cuantifica.	IDCT	PostProc.	Total	
0	22.231.304	31.949.307	24.986.927	1.103.095.183	238.439.287	1.420.702.008	1,8943 seg.
1	11.819.328	22.927.176	5.251.895	804.966.456	188.421.065	1.033.385.920	1,3778 seg.
2	11.998.307	23.056.586	5.264.861	476.696.138	185.279.810	702.295.702	0,9364 seg.
3	12.581.971	22.713.347	5.299.239	22.417.991	80.313.005	143.325.553	0,1911 seg.
4	12.196.677	23.201.289	5.515.107	24.209.187	58.861.170 (45.594.759)	123.983.430	0,1653 seg.

Tabla 1. Ciclos consumidos en la ejecución de distintas versiones del programa ejecutable jpeg para la imagen *sensation.jpg* de tamaño 651x507 píxeles. El total se muestra también en segundos.

Versión	PreProc.	Entropía	Cuantifica.	IDCT	PostProc.	Total	
0	199.447.421	210.624.617	248.061.269	10.887.416.445	2.268.030.990	13.813.580.742	18,418 seg.
1	91.872.159	153.737.675	52.452.921	7.941.436.233	1.880.995.067	10.120.494.055	13,494 seg.
2	91.907.311	151.360.028	52.527.640	4.687.296.770	1.864.541.342	6.847.633.091	9,130 seg.
3	102.988.656	158.353.327	56.500.800	223.694.530	840.953.735	1.382.491.048	1,843 seg.
4	93.466.460	156.428.114	53.190.553	224.931.647	675.947.272 (431.555.659)	1.203.964.046	1,605 seg.

Tabla 2. Ciclos consumidos en la ejecución de distintas versiones del programa ejecutable jpeg para la imagen *gladiator.jpg* de tamaño 2044x1603 píxeles. El total se muestra también en segundos.

V	PreProc.	Entropía	Cuantifica.	IDCT	PostProc.	Total	
0	1.058.033.739	1.245.022.425	1.300.937.763	56.790.821.647	12.149.408.467	72.544.224.041	96,73 seg.
1	494.082.363	911.295.916	273.857.193	41.375.100.757	10.245.955.535	53.300.291.764	71,07 seg.
2	491.510.464	914.454.782	275.542.000	24.308.083.918	10.070.236.280	36.059.827.444	48,08 seg.
3	526.428.886	932.158.849	280.805.647	1.159.061.793	4.575.981.152	7.474.436.327	9,97 seg.
4	510.515.417	919.902.202	280.914.782	1.169.168.836	4.454.250.089 (2.456.779.992)	7.269.502.442	9,69 seg.

Tabla 3. Ciclos consumidos en la ejecución de distintas versiones del programa ejecutable jpeg para la imagen *rabiez.jpg* de tamaño 4800x3600 píxeles. El total se muestra también en segundos.

En la *segunda iteración* del ciclo de optimización, se parte de la *versión 2* y se genera la *3* mejorando los bloques funcionales más lentos: IDCT y Postprocesamiento. En esta iteración, para mejorar prestaciones se ha optado por operar con aritmética entera en lugar de punto flotante de simple precisión y por la utilizado instrucciones SIMD del repertorio MMX de la familia x86. En

las operaciones se ha utilizado precisión de 16 bits, con esta precisión se obtiene una imagen reconstruida de calidad similar a la generada con aritmética punto flotante de simple precisión (ver Capítulo 5 en [5]). Utilizando enteros se evita el coste de las operaciones de conversión entre tipos de datos, y utilizando SIMD se pueden llegar a realizar cuatro operaciones en paralelo. Tanto en la IDCT, como en la conversión YCbCr a RGB, se emplean sumas de multiplicaciones. Para acelerar estos bloques se ha utilizado la instrucción MMX *pmaddwd*, que realiza cuatro multiplicaciones en paralelo seguidas de dos sumas en paralelo. Estas mejoras han supuesto una ganancia en prestaciones superior a 7 respecto a la *versión 1* no mejorada (Figura 6). La menor reducción de tiempo observada en el postprocesamiento respecto a la IDCT, es debido a que éste no sólo incluye conversión a RGB también el almacenamiento en un fichero de esta imagen RGB resultante.

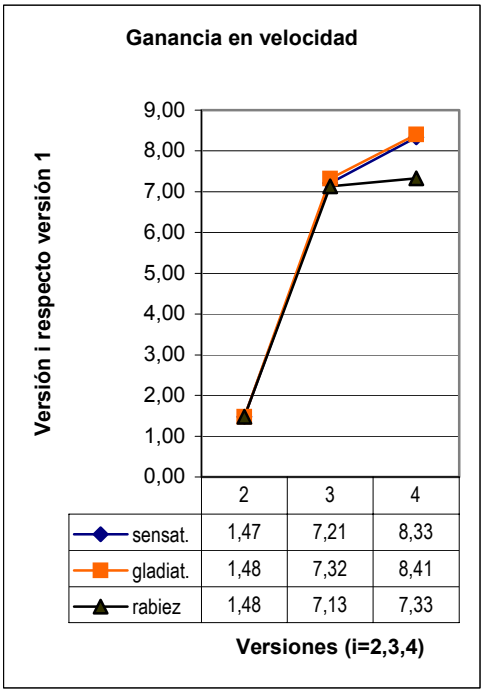


Figura 6. Ganancia en velocidad de las versiones 2, 3 y 4 del programa respecto a la *versión 1* para las tres imágenes

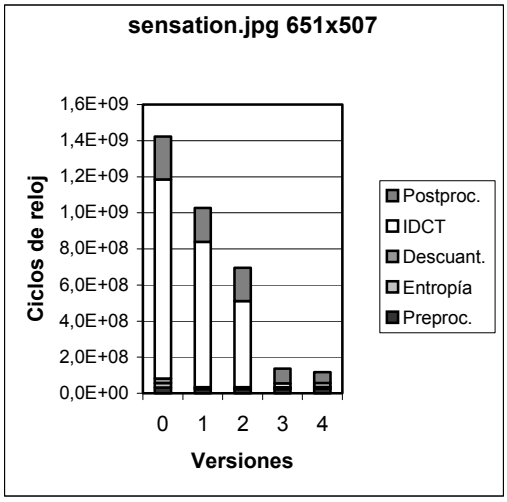


Figura 7. Ciclos de reloj para *sensation.jpg*

En la *tercera iteración* se parte de la *versión 3* generándose la *versión 4*. En la versión 3, la etapa de postprocesamiento domina en tiempo con respecto a las demás. Para mejorar esta etapa se ha separado la conversión a RGB del almacenamiento en un fichero, que se realizaban de forma solapada. Esto ha supuesto una reducción en tiempo, llegándose a ganancias superiores a 8. Con imágenes de gran tamaño, se incrementa la influencia del almacenamiento en disco en el postprocesamiento. Lo que se puede comprobar, comparando en la entrada 4 de las tablas, el tiempo de la conversión RGB (entre paréntesis) con el tiempo de postprocesamiento. Por este motivo, para rabiez.jpg, la ganancia es sólo de 7,33.

4. Conclusiones

Algunos de los trabajos fin de carrera que proponemos dentro del área de Arquitectura y Tecnología de Computadores, persiguen que los estudiantes comprueben que pueden emplear los conocimientos de arquitectura que han adquirido, para mejorar las prestaciones de sus programas. Las mejoras en prestaciones conseguidas en estos trabajos fin de carrera, se muestran a estudiantes que actualmente cursan asignaturas de arquitectura y estructura de computadores, con el objetivo de motivarles en el estudio de estas materias.

En estos trabajos se propone la programación de una aplicación, que despierte interés para el estudiante, en dos etapas (Figura 1). En la primera etapa, el estudiante genera un programa que resuelve la aplicación siguiendo su método habitual. En la segunda etapa, el código generado por el estudiante se mejora teniendo en cuenta la arquitectura actual de los computadores. En esta etapa el estudiante aplica conocimientos que ha adquirido sobre arquitectura y estructura de computadores durante la titulación. Para este proceso de mejora proponemos utilizar los pasos especificados en la Figura 1.

Aquí se muestran como ejemplo, la mejora en prestaciones obtenida a partir de un programa generado por un estudiante para descodificar imágenes en formato jpeg "*baseline*" ([15]). Se ha obtenido en un único procesador, una ganancia en prestaciones entre 7 y 8 con respecto al programa original, pasando el tiempo de procesamiento para una imagen con más de tres millones de píxeles de 13,49 segundos a 1,6 segundos en un Pentium de la familia P6 a 750MHz. Estas mejoras se han conseguido en menos de una semana, mientras que el estudio de la aplicación y su programación han llevado cerda de dos meses. Los estudiantes se sorprenden al ver que en poco tiempo, se pueden conseguir ganancias de 8 con un único procesador.

Actualmente se están desarrollando otros trabajos fin de carrera de optimización y se está aplicando esta idea en una práctica de 2 horas en la asignatura Estructura de los Computadores. En esta práctica, se optimiza el cálculo del mínimo de una lista de números usando una instrucción de movimiento condicional. La ganancia que se obtiene es superior a 2.

Referencias

[1] CodeAnalyst, http://www.amd.com/us-en/Processors/DevelopWithAMD/0,,30_2252_3604,00.html

[2] "AMD Athlon™ Processor x86 Code Optimization Guide", http://www.amd.com/us-en/Processors/ProductInformation/0,,30_118_1274_3734^3748,00.html

[3] Akramullah, S. M. ; Ahmad, I. ; Liou, M. L. "Optimization of H.263 Video Encoding Using a Single Processor Computer: Performance Tradeoffs and Benchmarking", IEEE Transactions on Circuits and Systems for Video Technology, vol. 11, n. 8, pp. 901-915, August 2001.

[4] Bhaskaran, V. ; Konstantinides K., "Image and Video Compression Standars", Kluwer.1999.

[5] Casalino, F. et al. "MPEG-4 Video Decoder Optimization", in Proc. IEEE Int. Conf. Multimedia Computing and Systems, vol.1, pp. 363-368, 1999.

[6] CodeAnalyst de AMD, ttp://www.amd.com/us-en/Processors/DevelopWithAMD/0,,30_2252_3604,00.html

[7] Djgpp home, http://www.delorie.com/djgpp/

[8] Dowd, K.; Severance C.R., Hihg Performance Computing", O'Really, 1998.

[9] "Intel® Pentium® 4 Processor Optimization Reference Manual", http://www.intel.com/design/pentium4/manuals/248966.htm

[10] ITU-T Recommendation T.81 y T.84, "Digital Compression and Coding of Continous-Tone Still Images", 1992.

[11] de Juan Herrero, J. "Introducción a la enseñanza universitaria. Didáctica para la formación del profesorado", Dykinson S.L., Madrid, 1995.

[12] Lappalainen, V. ; "Peformance of an Advanced Video Codec on a General-Purpose Processor with Media ISA Extensions", in IEEE Transactions on Consumer Electronics, vol. 46, no. 3, pp. 706-716, August 2000.

[13] Lappalainen, V. ; Hämäläinen, T. D. ; P. Liuha. "Overview of Research Efforts on Media ISA Extensions and Their Usage in Video Coding", in IEEE Transactions on Circuits and Systems for Video Technology, vol. 12, n. 8, pp. 660-670, August 2002.

[14] Padilla Pérez, J.A. "JPEG con MMX", Proyecto Fin de Carrera, ETSI Informática de Granada. Curso 2001-2002.

[15] VSDK de sun, http://www.sun.com/processors/vis/vsdk.html

[16] Vtune de Intel, http://developer.intel.com/software/products/vtune/

[17] Wadleigh, K.R., Crawford, I. L.; "Software optimization for High Performance Computing", Hewlett-Packard Professional Books, 2000.

De la pizarra al circuito: una metodología para mejorar el aprendizaje en las prácticas de diseño lógico

José Luis Poza, Juan Carlos Cano, Juan Luis Posadas
Dpto. de Informática de Sistemas y Computadores
Escuela Técnica Superior de Informática Aplicada
Universidad Politécnica de Valencia. 46022 Valencia
e-mail: {jopolu, jucano, jposadas}@disca.upv.es

Resumen

En este artículo se presenta una metodología que pretende servir de apoyo para una mejor comprensión, por parte del alumno, de los conceptos de diseño de circuitos digitales. Dicha metodología se aplica a las prácticas de laboratorio y utiliza dos herramientas metodológicas complementarias.

Tradicionalmente las prácticas de apoyo relacionadas con lógica digital se vienen realizando, en la Universidad Politécnica de Valencia, en entrenadores lógicos, los cuales presentan el principal inconveniente de su limitación de uso y complejidad creciente del montaje en relación a la magnitud del circuito.

El objetivo principal de la metodología propuesta es ofrecer una herramienta de aprendizaje basada en simulaciones, que permita al alumno poner en práctica los conceptos que posteriormente serán utilizados en la implementación de circuitos en el entrenador.

Así, el alumno tiene la posibilidad de adquirir los conceptos tratados en clases teóricas y ponerlos en práctica mediante una transición más suave: pizarra, simulación, circuito real.

1. Introducción

La enseñanza de materias como Estructura y Tecnología de Computadores implica la transmisión de una gran cantidad de conceptos novedosos para el alumno de primer curso.

Para facilitar la comprensión de estos conocimientos se pueden utilizar diferentes herramientas didácticas de apoyo a las clases magistrales de teoría, inevitables en un primer curso universitario.

La herramienta más utilizada son las clases prácticas de laboratorio. En la asignatura Estructura y Tecnología de Computadores I, impartida en la Escuela Técnica Superior de Informática Aplicada, dichas prácticas se realizan utilizando un entrenador de circuitos lógicos que permite implementar un amplio rango de circuitos digitales. Dicho entrenador resulta motivador para el alumno, debido a que construye circuitos reales a partir de una especificación teórica.

Sin embargo, la utilización del entrenador lógico, tiene sus principales inconvenientes en su reducida disponibilidad para el alumno y el gran número de cables necesarios para implementar un circuito de complejidad mediana. En ocasiones, aspectos tales como rotura de cables, mal conexionado, etc. provoca que el alumno se desvíe del objetivo que persigue la práctica. Para subsanar dichos inconvenientes se plantea la posibilidad de emplear una herramienta metodológica complementaria, que no suplementaria, que permita al alumno realizar una preparación previa a la realización de la práctica en el entrenador.

Los restantes apartados se organizan como sigue. En el apartado 2 se describe la asignatura de Estructura y Tecnología de Computadores en cuyas prácticas se ha ensayado el método. El apartado 3 se dedica a explicar los fundamentos pedagógicos y la problemática del entorno actual de las prácticas, así como las dos herramientas empleadas en la metodología presentada. Un ejemplo de aplicación se presenta en el apartado 4. Finalmente las conclusiones se exponen en el apartado 5.

2. Entorno

La asignatura *Estructura y Tecnología de Computadores 1* (ETC1) [1] aparece en los actuales planes de estudio de la Escuela Técnica Superior de Informática Aplicada en la Universidad Politécnica de Valencia.

Esta asignatura es de carácter troncal y anual con un total de 12 créditos (9 teóricos + 3 laboratorio) y se imparte en el primer curso de las titulaciones de Ingeniería Técnica en Informática de Sistemas (ITIS) e Ingeniería Técnica en Informática de Gestión (ITIG).

Los alumnos que cursan esta asignatura presentan una escasa integración en el sistema universitario, hay que tener en cuenta que es en los primeros cursos y, especialmente en las asignaturas troncales y obligatorias, donde se pone de manifiesto un mayor grado de masificación de los estudios de Informática en la Universidad Politécnica de Valencia. Este factor influye negativamente tanto en los alumnos como en el profesorado.

El número de alumnos de ETC1 en el actual curso académico es del orden de 600. Estos se dividen en 7 grupos de teoría y en 21 grupos de laboratorio. Así, resulta una media de 85 alumnos por grupo de teoría y un máximo de 30 por grupo de laboratorio. El laboratorio de prácticas dispone de 15 puestos de trabajo, donde trabajan 2 alumnos por puesto.

Los objetivos de la asignatura son:

- Conocer y comprender las técnicas de representación de datos tanto numéricos como alfanuméricos en el computador.
- Iniciar al alumno en la comprensión del funcionamiento e implementación de circuitos combinacionales y secuenciales a partir de puertas lógicas. Estos circuitos serán la base utilizada posteriormente para la creación de los circuitos correspondientes a las unidades funcionales del computador.
- Comprender los principios de la programación en ensamblador, tanto de programas elementales como de subrutinas.
- Conocer una ruta de datos de una máquina elemental capaz de ejecutar un conjunto de instrucciones, y comprender cómo se ejecutan.
- Como último objetivo, que fundamenta la existencia de las prácticas, es adquirir una formación práctica, con ejemplos reales en el laboratorio, en la que se apliquen y consoliden los conocimientos aprendidos en clases de teoría.

2.1. Teoría

Los contenidos teóricos [2], se han estructurado en los cuatro bloques que se muestran en la tabla 1.

BLOQUE 1: Representación
Introducción al Computador, Representación de la información, Aritmética Básica
BLOQUE 2: Circuitos Lógicos
Puertas, Circuitos Combinacionales (circuitos de aritmética básica, ALU), Circuitos Secuenciales (Biestables, Banco de Registros, Contadores)
BLOQUE 3: La Interfaz HW/SW
Juegos de Instrucciones y Modos de Direccionamiento, Lenguaje Ensamblador, Ensamblado, Enlace y Carga
BLOQUE 4: La Estructura del Computador
Procesador: Buses, Ruta de Datos, Unidad de Control, microprogramación

Tabla 1. Bloques temáticos para ETC1

2.2. Prácticas

Las prácticas en el laboratorio son el complemento esencial a las clases de teoría y problemas, sobre todo cuando se están enseñando materias tecnológicas y de carácter aplicado. Hay que pensar que en ellas se acerca al alumno a la realidad del mundo laboral exterior además de servirle para aplicar los conceptos asimilados en las clases de teoría.

Los objetivos que se persiguen con las prácticas de laboratorio son:

- Permitir que el alumno compruebe experimentalmente una gran parte de la materia explicada en teoría.
- Permitir la aplicación práctica de los conocimientos adquiridos en las clases teóricas.

- Despertar el interés por los temas tratados, procurando que el alumno razone y profundice en estos temas de forma práctica.
- Aprender a trabajar en equipo, facilitando el intercambio de ideas, como después es usual que ocurra en su actividad profesional.

Con estos objetivos, se plantea la realización obligatoria de un conjunto de 11 prácticas (tabla 2) directamente relacionadas con los bloques temáticos impartidos en las clases teóricas.

BLOQUE 1: Representación	
P1	Problemas de representación de la información.
BLOQUE 2: Circuitos Lógicos	
P2	Manejo del entrenador lógico. Puertas y circuitos básicos. Álgebra de Boole.
P3	Formas canónicas, simplificación, diseño e implementación de circuitos.
P4	Decodificadores y Multiplexores.
P5	La Unidad Aritmético Lógica (UAL).
P6	Circuitos secuenciales. Registros y Contadores.
BLOQUE 3: La Interfaz HW/SW	
P7	Introducción al simulador de procesadores MIPS. Codificación de instrucciones y datos.
P8	Programación en ensamblador.
BLOQUE 4: La Estructura del Computador	
P9	Introducción al Xilinx. El procesador I: Banco de Registros y UAL
P10	El procesador II: Ruta de Datos
P11	El procesador III: Unidad de Control

Tabla 2. Prácticas en ETC1

El control de las prácticas se realiza en el propio laboratorio, al resolver las dudas planteadas y al revisar el correcto funcionamiento del circuito o del programa. Además, el alumno entrega al finalizar la sesión de prácticas la memoria correspondiente donde debe resolver una serie de cuestiones que se le plantean.

En los grupos de prácticas se han detectado algunos problemas que han motivado la búsqueda de herramientas complementarias de apoyo al trabajo habitual del laboratorio. Algunos de estos problemas, y sus consecuencias son los siguientes:

- Masificación de alumnado. Consecuentemente es habitual que dos alumnos compartan el entrenador, con lo que sólo uno de ellos lo manipula directamente. Además, el elevado número de grupos de prácticas no permite el acceso libre al laboratorio.
- Distancia conceptual y metodológica con el contenido: Los alumnos pasan de ver o dibujar circuitos estáticos, a manipular cables reales, circuitos integrados alimentados.
- Falta de disponibilidad: El entrenador no se puede llevar al aula para mostrarlo.

3. Metodología propuesta

En la enseñanza universitaria [3], los alumnos aprenden los conceptos de la teoría por medio de los contenidos, estos pueden proporcionarse al alumno por medio de varias vías (figura 1). Las clases magistrales o las sesiones de problemas pertenecen a la vía teórica. El profesor explica los conceptos y el alumno estudia los contenidos por medio de los apuntes.

Las prácticas aparecen en el proceso de enseñanza-aprendizaje para facilitar la comprensión de los contenidos dando diferentes visiones que le facilitarán el estudio. A partir de los conocimientos adquiridos en la teoría y, por medio de las habilidades desarrolladas en las prácticas, el alumno tiene diferentes puntos de vista del mismo contenido.

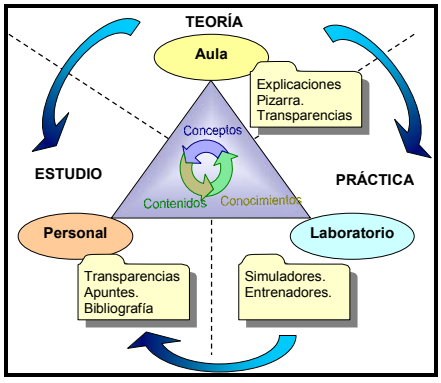


Figura 1. Vías de aprendizaje.

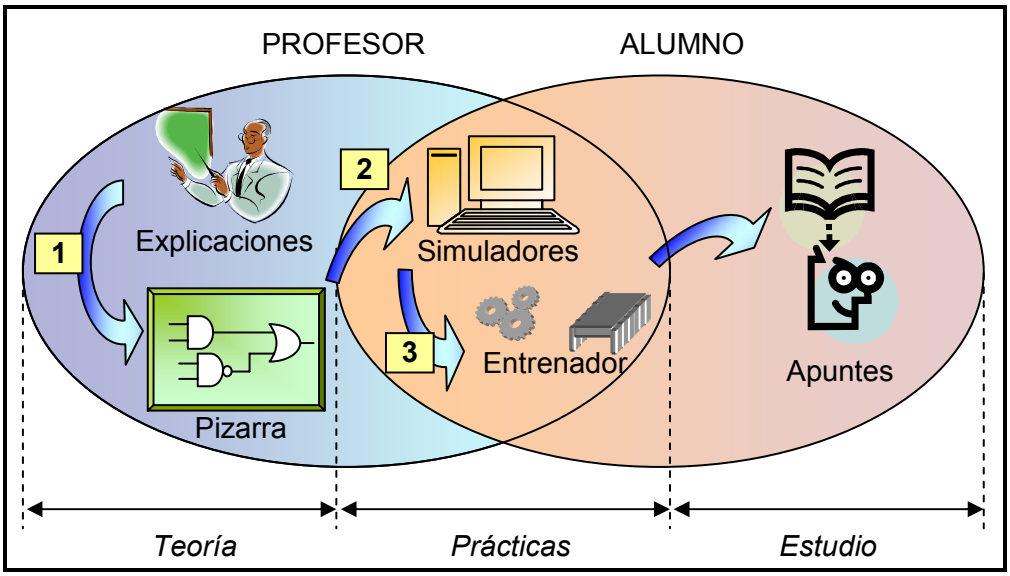


Figura 2. Esquema de la metodología propuesta y ensayada.

De las vías de aprendizaje, algunas dependen de un profesor que asesore, ayude o acompañe en el proceso y de las explicaciones, tanto verbales como en pizarra. En estos casos el protagonista es el profesor. Las prácticas también requieren de un profesor, sin embargo, en las prácticas el alumno es el protagonista, todo lo contrario que en las sesiones de teoría.

Tanto las metodologías que se emplean en teoría como las que se emplean en prácticas sirven para proporcionar al alumno unos conocimientos y unas destrezas que le permitan, por medio del estudio, adquirir los conocimientos necesarios (figura 2).

La metodología propuesta se basa en apoyar los contenidos que el profesor explica, por medio de ejemplos de la pizarra (figura 2, vía 1). El circuito explicado de forma estática en la pizarra se visualiza dinámicamente por medio del simulador, esta simulación se puede realizar directamente en el aula. A continuación, el alumno construye el circuito con el simulador XILINX (figura 2, vía 2), de esta manera se reafirman los conceptos de construcción, composición y conexión de componentes. A continuación, el alumno realiza la práctica más fácilmente debido a que ya tiene claro el funcionamiento (figura 2, vía 3) en el entrenador.

3.1. Xilinx

La herramienta XILINX [4] se emplea para diseño de circuitos digitales. Las características más notables de esta herramienta son:

- Realización de esquemas de circuitos digitales por medio de los símbolos correspondientes.
- Inserción de componentes de librerías o creación de componentes personales.
- Simulación de circuitos, tanto de forma visual como por medio de cronogramas.

La versión que se ha empleado en las prácticas es la de aprendizaje (figura 3), esta versión no es operativa para un entorno de desarrollo empresarial, pero sí en un entorno académico.

XILINX está preparado para realizar complejos circuitos sobre tarjetas reales, por ello todo el proceso que se realiza aparece en el marco de diseño. De este marco se emplea la herramienta de diseño (Schematic Editor) y la de simulación (Simulation) que aparecen destacadas en la figura.

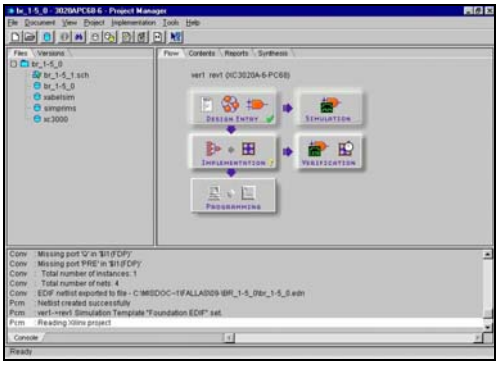


Figura 3. Entorno de desarrollo XILINX

Todo circuito que se cree y simule con XILINX estará compuesto de una serie de componentes como puertas lógicas, codificadores, multiplexores, etc. En XILINX estos componentes se encuentran agrupados en librerías. Las librerías se pueden crear y personalizar, de manera que en las prácticas se le puede proporcionar al alumno una serie de componentes ya seleccionados previamente.

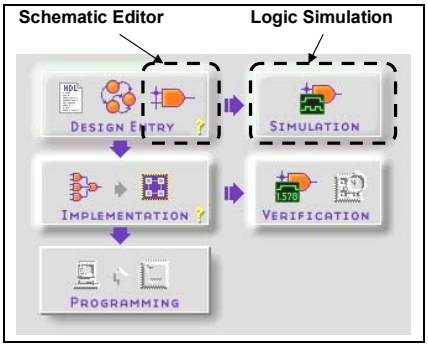


Figura 4. Herramientas de XILINX

3.2. Entrenador

El entrenador digital utilizado en prácticas es un laboratorio portátil diseñado especialmente para comprobar todo tipo de circuitos digitales. Esta herramienta es un instrumento de trabajo que permite al alumno poner en práctica los diferentes conceptos tratados en clases teóricas, verificando el funcionamiento de los diferentes circuitos que se proponen de una forma motivadora e instructiva.

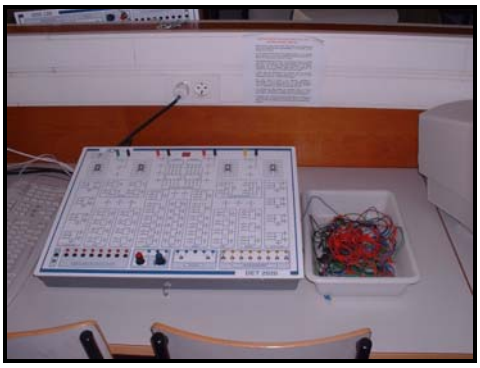


Figura 5. Entrenador lógico utilizado.

La figura 5 muestra un imagen del entrenador de circuitos digitales utilizado en prácticas. El entrenador incorpora los siguientes elementos:

- Interruptores y monitores lógicos: permiten obtener las entradas de los diferentes circuitos digitales realizados así como monitorizar las salidas de los mismos.
- Puertas lógicas de diferente tipo y con diferente número de entradas.
- Biestables J-K así como Registros y Contadores, que permiten realizar una amplia gama de circuitos secuenciales.
- Zócalos: los zócalos del entrenador se utilizan para conectar circuitos integrados comerciales. A través de los bornes asociados se tiene acceso a las patillas del circuito integrado.

4. Práctica ejemplo: multiplexores

La metodología propuesta en el apartado 3 se ha puesto en práctica con los alumnos voluntarios en un grupo "piloto" de la asignatura. Debido a las características de la asignatura, se decidió realizar esta experiencia con alumnos que se ofrecieran voluntarios.

La práctica escogida para probar esta metodología fue la práctica de multiplexores y decodificadores. Esta práctica es la tercera en la que emplea el entrenador, y la primera en la que los contenidos están relacionados con lógica digital, pudiendo ser fácilmente implementada con la herramienta de simulación XILINX.

4.1. Descripción de la práctica

En prácticas anteriores, los alumnos han puesto en práctica los conceptos de la lógica combinacional, trabajando con puertas y funciones lógicas elementales. Igualmente han utilizado la lógica digital para implementar funciones simplificadas.

El objetivo de esta tercera práctica es que el alumno conozca el funcionamiento de circuitos de mayor complejidad que serán utilizados en la construcción de unidades funcionales tales como el Banco de Registros y la Unidad Aritmético Lógica. En concreto, en esta práctica los alumnos trabajan con circuitos decodificadores y multiplexores.

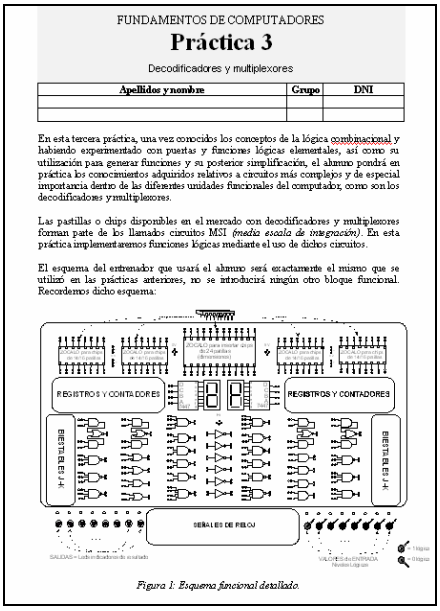
El alumno debe realizar las siguientes tareas:

- Implementación y composición de decodificadores binarios, utilizando el integrado 74139 (incorpora dos decodificadores de dos entradas)
- Implementación y composición de multiplexores, utilizando el integrado 74153 (incorpora dos multiplexores de 4 entradas)
- Implementación de funciones lógicas utilizando los circuitos anteriormente construidos.
- Construcción de circuitos con entrada de selección. A partir de los circuitos anteriores el alumno debe modificarlos para incorporar una entrada de selección.

Los alumnos realizan la práctica utilizando un boletín de prácticas (ver figura 6) en el que se les va dirigiendo a través de las diversas actividades que deben realizar. Para aprovechar el tiempo de utilización del laboratorio los alumnos deben tener preparado dicho boletín antes de iniciar su correspondiente sesión práctica. Este boletín puede consultarse en la página Web de la asignatura [5]. En el mismo boletín tienen el espacio necesario para responder a las preguntas.

Para realizar un seguimiento de las actividades del alumno, éste entrega el boletín al profesor para que lo corrija. Finalmente, el profesor, devuelve el boletín al alumno, para que pueda verificar los diferentes errores que haya podido cometer.

A continuación se describen algunos montajes de la práctica utilizando ambas metodologías.



FUNDAMENTOS DE COMPUTADORES

Práctica 3

Decodificadores y multiplexores

Apellidos y nombre	Grupo	DNI

En esta tercera práctica, una vez conocidos los conceptos de la lógica combinacional y habiendo experimentado con puertas y funciones lógicas elementales, así como su utilización para generar funciones y su posterior simplificación, el alumno pondrá en práctica los conocimientos adquiridos relativos a circuitos más complejos y de especial importancia dentro de las diferentes unidades funcionales del computador, como son los decodificadores y multiplexores.

Las pastillas o chips disponibles en el mercado con decodificadores y multiplexores forman parte de los llamados circuitos MSI *(media escala de integración)*. En esta práctica implementaremos funciones lógicas mediante el uso de dichos circuitos.

El esquema del entrenador que usará el alumno será exactamente el mismo que se utilizó en las prácticas anteriores, no se introducirá ningún otro bloque funcional. Recordemos dicho esquema:

Figura 1: Esquema funcional detallado.

Figura 6. Boletín de prácticas.

4.2. Montaje con XILINX

En el montaje de la práctica con XILINX, el alumno desarrolla primero el circuito lógico que va a construir. Éste circuito implica la conexión de las entradas del multiplexor a nivel alto o nivel bajo, según corresponda, para a continuación realizar las conexiones de la selección del multiplexor. El circuito desarrollado se puede ver en la figura 7.

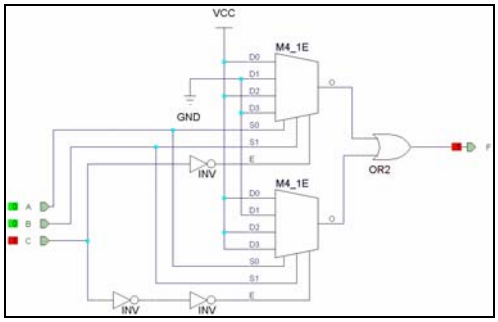


Figura 7. Simulación del circuito de la práctica explicado en el aula.

Este circuito se desarrolla de manera similar al desarrollado en la pizarra, de tal manera que sirve de conexión de la clase teórica con la herramienta XILINX. La gran ventaja de desarrollar el mismo circuito que en el explicado en el aula, es poder realizar una simulación de manera dinámica, con la observación directa de los resultados y la comparación con la tabla de verdad del comportamiento del circuito presentada por el profesor en la teoría.

A continuación el alumno realiza el mismo circuito, pero utilizando el circuito integrado 74153 que es proporcionado por las librerías de circuitos del XILINX. El resultado se puede ver en la figura 8.

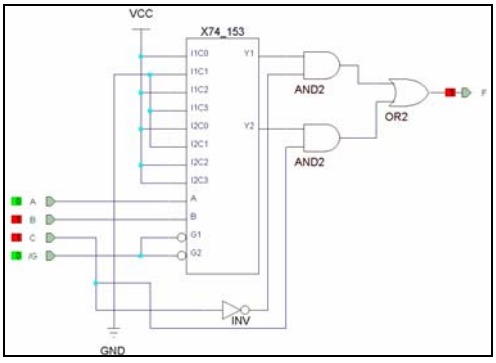


Figura 8. Simulación del circuito a desarrollar en la práctica.

Este último paso de simulación se corresponde con el circuito que deben construir en el entrenador, de esta forma se logra la conexión de lo explicado en la pizarra con lo practicado en el entrenador.

4.3. Montaje en el entrenador

La Figura 9 muestra el esquema del patillaje del integrado 74153. A partir de esta descripción, el alumno debe ser capaz de montar dicho circuito en uno de los zócalos del entrenador comprobando su correcto funcionamiento.

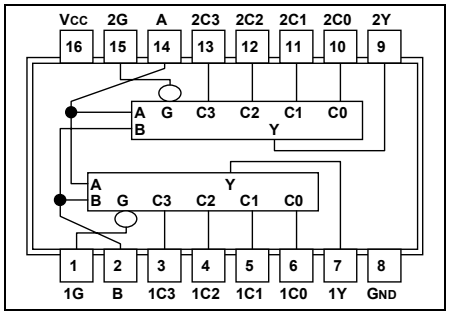


Figura 9. Patillaje del multiplexor 74153

Una vez implementado el circuito, el alumno debe comprobar su correcto funcionamiento completando una tabla de verdad que describa el comportamiento de dicho circuito.

A partir de esta comprobación inicial se plantean circuitos de mayor complejidad que permiten analizar la asimilación por parte del alumno, tanto de los conceptos teóricos como de sus habilidades de construcción de circuitos digitales.

Por medio del circuito integrado 74153 y de las puertas lógicas disponibles en el entrenador, se le plantea al alumno la composición de un multiplexor de 3 entradas de selección. Basándose en el esquema realizado como trabajo previo a la sesión de laboratorio, el alumno debe construir el circuito en el entrenador utilizando los latiguillos de conexiones que sean necesarios (ver figura 10).

Por cada uno de los circuitos construidos, el alumno debe verificar el funcionamiento del circuito implementado.

Por cada una de las posibles combinaciones de las entradas, el circuito construido debe implementar la función indicada en el boletín de prácticas.

Para ello, el alumno debe comparar la salida del circuito con los resultados previamente evaluados y anotados en una tabla de verdad de funcionamiento del circuito.

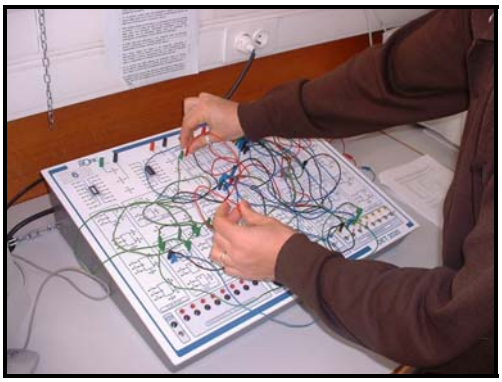


Figura 10. Construcción de un circuito en el entrenador.

Finalmente cabe indicar que la utilización de un entrenador lógico resulta una práctica altamente motivadora para el alumno, el cual construye pequeños circuitos utilizando integrados reales de una forma tangible. Sin embargo, este tipo de herramientas solamente son viables para pequeños circuitos, donde el número de cables no sea elevado. A medida que los circuitos necesitan de la utilización de un mayor número de cables el entrenador adolece de no ser modular y la construcción de los mismos se hace más compleja, aumentando la probabilidad de fallo debido a un mal montaje o simplemente a fallos de los elementos de conexión. Como ejemplo, la figura 11 muestra detalles del montaje anterior.

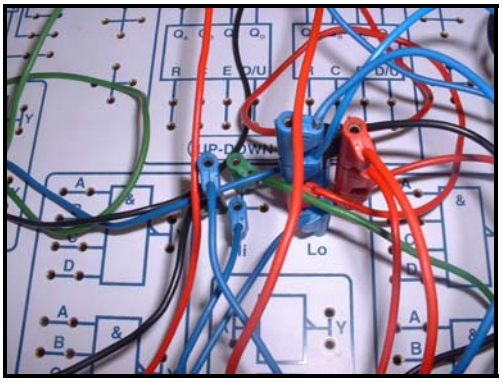


Figura 11. Detalles de implementación del circuito en el entrenador.

5. Conclusiones

En esta experiencia se ha evaluado un método consistente en la preparación previa de las prácticas por medio de un software de simulación de circuitos.

Aunque la experiencia presentada se ha llevado a cabo en un solo grupo de prácticas, los resultados obtenidos muestran que dichos alumnos incrementaron sus habilidades, tanto en el montaje como en la comprensión, de cada uno de los circuitos, así como las cuestiones de ampliación propuestas en las prácticas.

Finalmente, faltaría comprobar la incidencia en la evaluación de dicha metodología, comparando los resultados que estos alumnos obtengan en los ejercicios de examen relacionados con los contenidos de las prácticas, con las notas de sus compañeros. Aspectos que quedan como tareas pendientes.

Como propuestas futuras, aunque requerirá de un esfuerzo adicional, tanto por parte del profesor como del alumno, está el extender esta metodología a todas las prácticas en las que se emplee el entrenador y a todos los grupos. En una primera fase se plantea que sea de forma voluntaria, en tal caso se realizará un seguimiento de los resultados de estos alumnos tanto en las prácticas como en las preguntas del examen que estén relacionadas con los contenidos que se han aprendido en prácticas.

Referencias

[1] Estructura de computadores. Julio Sahuquillo y otros. SPUPV. 1998.
[2] Patterson D.A. y Hennesy J.L. Organización y diseño de computadores. Mc Graw-Hill, 1994
[3] Benedito, I., Antoli, V. y otros. Innovaciones en el aprendizaje universitario. PPU. Barcelona, 1988.
[4] Página Web de XILINX. www.xilinx.com/
[5] Página Web de la asignatura ETC1. poseidon.disca.upv.es/etc1

Motivando a los estudiantes en prácticas relacionadas con Estructura de los Computadores

F. J. Fernández, A. Cañas, A. F. Díaz, M. Anguita, H. Pomares, A. Prieto
Dpto. de Arquitectura y Tecnología de Computadores
Universidad de Granada
Escuela Técnica Superior de Ingeniería Informática
e-mail: {javier, acanas, afdiaz, manguita, hector}@atc.ugr.es, aprieto@ugr.es

Resumen

En este trabajo se presentan diferentes actividades docentes desarrolladas durante los últimos cursos en las prácticas de varias asignaturas del área de Arquitectura de Computadores en la Universidad de Granada, destinadas a motivar al estudiante en cada asignatura concreta y en el área en general. Se destaca, por sus excelentes resultados de cara a la motivación, la actividad consistente en el diseño, montaje y evaluación de un ordenador financiado por los propios estudiantes y sorteado entre los mismos al finalizar el curso.

1. Introducción

La asignatura de Estructura de los Computadores tiene una importante función motivadora hacia otras asignaturas del área de Arquitectura de Computadores en los planes de estudio de Informática, dado su carácter troncal y su frecuente ubicación en 1er ó 2º curso. Según la adscripción de materias a áreas de conocimiento en cada universidad, puede suceder que la única asignatura del área de Arquitectura que cursen los estudiantes después de Estructura en 2º sea Arquitectura en 5º, tres años después. Durante esos tres años el estudiante puede desligarse completamente de los conocimientos del área si no escoge ninguna optativa de la misma, para encontrarse en 5º con una o dos troncales que debe cursar obligatoriamente y con una cuestionable motivación. La forma correcta de resolver el problema es conseguir una alta motivación en las asignaturas de Estructura procurando que los estudiantes no salgan de ellas sin haber aprendido, como sugiere [15].

Las actividades aquí descritas han sido diseñadas para conseguir una mayor motivación del estudiante hacia las siguientes asignaturas (y hacia el área en general) en la Universidad de Granada:

- *Estructura de los Computadores* (EC), en 2º de Ing. Técn. en Inform. de Gestión (ITIG) y Sistemas (ITIS).
- *Estructura de los Computadores 1* (EC1) y *2* (EC2), en 2º de Ing. Informático (II).
- *Introducción al Diseño de Computadores* (IDC), en 3º de ITIS.
- *Arquitecturas de Sistemas Basados en Microprocesadores* (ASBM), optativa en 3º de ITIG e ITIS.
- *Procesadores Integrados* (PI), en 4º curso de Ing. Electrónico (IE).

Las propuestas de innovación consisten principalmente en un proyecto sobre el montaje de un PC, y en una serie de mejoras en el resto de las prácticas, sin olvidar la aplicación de las TIC: materiales en formato hipermedia, tutorías electrónicas, páginas web de las asignaturas y uso de una plataforma virtual de desarrollo propio [4].

En la mayor parte de las actividades prácticas descritas se ha seguido un modelo de "aprendizaje por proyectos" para fomentar la autoformación de los estudiantes, al estilo de [13].

2. Proyecto sobre el montaje de un PC

Dentro del marco de la *Primera Propuesta de Actuaciones 2002* de la UGR [16], se solicitó un Proyecto de Innovación Docente en el que han participado siete profesores del departamento, consistente en el diseño, montaje y evaluación de un PC por parte de los estudiantes.

Esta actividad se ha desarrollado en algunas de las asignaturas mencionadas (EC en ITIG, ASBM en ITIS, y EC2 en II). Se trata de conseguir que los estudiantes puedan responder a las típicas preguntas *¿este ordenador que he escogido es bueno?*, *¿la relación precio-prestaciones es correcta?*, *¿se puede comprar más barato en otro lugar?*, *¿me lo puedes escoger (o comprar) tú?*, y contribuir a su formación como futuros profesionales informáticos.

2.1. Objetivos

- Dominar, al menos durante un cuatrimestre, los precios de los diversos componentes.
- Tener una idea de primera mano acerca de la velocidad a la que evolucionan los mismos, y de los márgenes con que trabajan los comercios del ramo.
- Dominar la información disponible: manuales *online* de fabricantes, foros, sitios web, etc.
- Aplicar los conocimientos teóricos adquiridos en la asignatura, y mejorar la percepción de su utilidad por parte de los estudiantes.
- Permitir a los estudiantes conocer las distintas destrezas de sus compañeros en un entorno competitivo pero controlado.
- Mejorar la relación estudiante-profesor, y la de los estudiantes entre sí.

2.2. Contenido

Los alumnos siguen todos los pasos que implican la adquisición y construcción de un ordenador por piezas. El proyecto se ha subdividido en cuatro apartados, cada uno de los cuales permite subir la nota del examen de teoría hasta 1 punto:

- *Póster PC*: esta innovación se introdujo hace dos cursos y es en cierto modo independiente de los siguientes tres apartados, por lo que se describe más adelante (Sección 3.1, Figura 4). En este curso se ha utilizado como ejercicio preparatorio para el resto del Proyecto PC.
- *Recolección de publicidad*: cada estudiante realiza un par de diseños de computador a partir de las piezas sueltas (salvo el monitor), aportando una demostración del precio de los componentes y su compatibilidad (Figura 1).

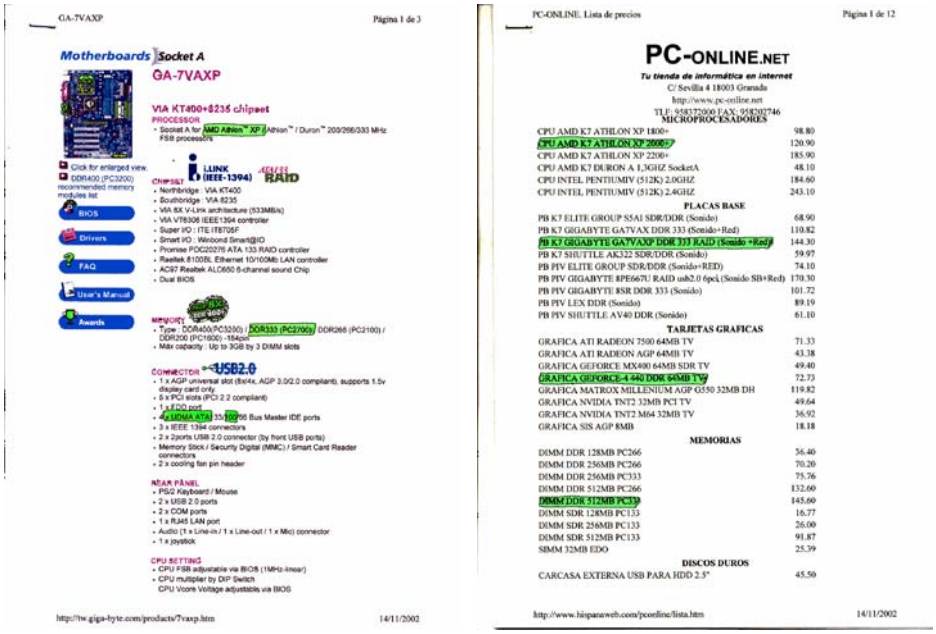


Figura 1. Ejemplo de publicidad: (izda.) web de *Gigabyte*, en donde se resalta la compatibilidad de los componentes escogidos; (dcha.) web de *PC-Online Granada*, donde el estudiante ha destacado los precios oficiales de los mismos. Siendo documentos "oficiales" públicamente accesibles, las URL proporcionadas permiten la comprobación.

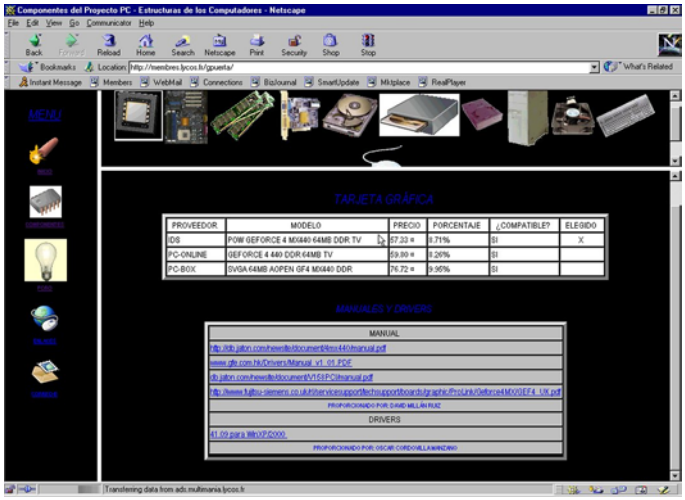


Figura 2. Ejemplo de actualización del equipo. Cada cambio aceptado por el profesor implica que los responsables correspondientes busquen el nuevo manual PDF y soliciten la actualización de las respectivas URL al *webmaster*.

- *Diseño y compra del PC*: dado un diseño inicial de un PC con un coste precalculado, si el número de estudiantes que se comprometen a realizar este apartado (y el siguiente) es suficientemente elevado como para que la cuota sea reducida (máximo 12 €), se procede a solicitar sucesivas modificaciones en el diseño, cambiando componentes siempre dentro del presupuesto total.

  Las intervenciones se formalizan en un foro de discusión donde quede constancia del autor, fecha y hora de cada propuesta, con el objeto de asignar una puntuación según la calidad de cada modificación.

  Se puede obtener parte de la puntuación por tareas administrativas asociadas: gestión de las cuotas, del foro, de una página web actualizada que refleje en cada instante el diseño actual (Figura 2), aportación de manuales impresos, transporte para realizar la compra, etc.
- *Montaje y sorteo del PC*: los componentes se traen al centro, los estudiantes se reparten en grupos de tamaño adecuado (≈10), y se procede en sucesivas sesiones a montar el ordenador (Figura 3), instalarle un sistema operativo y ejecutar algunos programas *benchmark* habituales para estimar sus prestaciones, así como alguna aplicación con la cual verificar su correcto funcionamiento.

  Para calmar la inquietud que este apartado puede despertar en parte del estudiantado, el profesor puede recomendar la lectura previa de algún libro con ilustraciones como [11], [14] o cualquier otro documento de la extensa literatura y sitios web existentes [3].

  El sorteo puede realizarse en cualquier fecha posterior al montaje por parte del último grupo. Es conveniente que sea un acto público, al objeto de evitar discusiones por el resultado. Participan en el mismo todos los estudiantes que hayan pagado

Figura 3. Ejemplo de montaje. Se organizan grupos de un tamaño apropiado (5 grupos de 11 en nuestro caso) y se procede a montar el PC, instalar un SO, ejecutar algunas aplicaciones y algunos programas *benchmark*.

la cuota, independientemente de su calificación.

El ordenador se compra sin monitor, dada la enorme diferencia de precios según el fabricante y las prestaciones del mismo. Durante las sesiones de montaje se puede usar el monitor del profesor o algún otro disponible en el departamento. En este curso el Vicerrectorado ha aportado un monitor que también ha sido añadido al PC para su sorteo.

2.3. Evaluación de la experiencia

Se ha observado que los estudiantes pierden el miedo a las entrañas del ordenador, pueden hablar con propiedad y conocimiento de causa sobre precios y prestaciones, y aprenden a discutir públicamente utilizando fuentes documentales. Por su parte, el profesor mejora su imagen ante el estudiantado y reutiliza en la asignatura el esfuerzo de actualizarse sobre precios y prestaciones.

Se ha desarrollado un cuestionario que se pasó tras el sorteo del PC en la asignatura EC de ITIG, y los resultados son muy positivos: los 56 estudiantes que pagaron la cuota quedaron muy satisfechos con la experiencia y fuertemente interesados no sólo por la asignatura EC sino por otras optativas que pudieran ampliar sus conocimientos del área.

En futuras ediciones se debe reforzar la puntuación del apartado de "diseño" frente al de "montaje". Los estudiantes no dudan en señalar este último como el más importante; claramente es el que produce una mayor motivación, pero dominar las modestas habilidades manuales que requiere dicho montaje no es el núcleo cognoscitivo que trata de transmitir la asignatura de Estructura.

3. Mejoras en el resto de las prácticas

Además del proyecto de montaje de un PC, en los últimos años se ha intentado mejorar el resto de las prácticas de las asignaturas enumeradas en la Sección 1. El objetivo principal ha sido elaborar prácticas atractivas que motiven al estudiante sin descuidar los aspectos pedagógicos. Muchas de ellas utilizan hardware real, necesidad ya destacada en [8].

Cada estudiante o pareja de estudiantes realiza una o varias de las siguientes prácticas, dependiendo de la asignatura:

- Póster sobre la arquitectura de un PC.
- Práctica de ensamblador de 80x86.
- Práctica de microprogramación.
- Práctica de entrada/salida (E/S) con PCLab y SoundBlaster.
- Práctica de E/S con el microcontrolador 8051.
- Práctica con procesadores de señales digitales (DSP).

A continuación se describen tales prácticas y se indican las asignaturas en las que se realizan.

3.1. Póster sobre la arquitectura de un PC

Se realiza en las mismas asignaturas que el Proyecto PC (EC en ITIG, ASBM en ITIS, y EC2 en II). Cada estudiante saca una foto del interior de su PC y añade la información técnica que pueda encontrar sobre los distintos componentes y sus prestaciones, al estilo de los trabajos de infografía encontrados en revistas del ramo (Figura 4). Se puede recomendar el uso de programas de información del sistema como el AIDA 32 [10] para ayudar a los estudiantes menos avezados a completar con éxito esta actividad.

3.2. Práctica de ensamblador de 80x86

Constituye el núcleo principal de prácticas en las asignaturas EC en ITIG e ITIS, EC1 en II, y PI en IE. Utilizamos la arquitectura 80x86 por su disponibilidad tanto en los centros como por parte de los estudiantes —con otros microprocesadores sería necesario el uso de simuladores—.

Hace varios años, cada estudiante o pareja escribía 5 ó 6 programas de menos de 100 líneas. Los estudiantes consideraban estos pequeños programas retos asequibles debido a su corto enunciado y relativamente rápida implementación. Sin embargo, detectamos que la motivación era escasa debido a no poder realizar programas vistosos, que requieren más líneas, y a que realizar un cuarto o quinto programa se convertía en una tarea repetitiva.

Actualmente se realiza una única práctica obligatoria por pareja (los estudiantes muy motivados pueden realizar más) de unas 1000 ó 1500 líneas de código. Las 6 ó 7 prácticas disponibles son pequeños proyectos relacionados con sonido, gráficos, juegos o comunicaciones. Cada proyecto cuenta con un detallado guion en el

Figura 4. Ejemplo de póster sobre la arquitectura del PC. Para obtener la máxima calificación se debiera haber añadido información sobre la estructura de buses (FSB, DDR, PCI, *North-SouthBridge*) y sus respectivas velocidades.

que se marcan unos requisitos mínimos y se proponen diversas mejoras optativas para aumentar la calificación. El profesor asigna a cada pareja uno de los proyectos con el objeto de que la distribución sea más o menos uniforme. Una pareja puede intercambiar el proyecto asignado con otra, o bien plantearse un proyecto alternativo de similar complejidad.

Debido a la magnitud de los objetivos marcados, se dedica más de medio cuatrimestre a su realización. En las dos primeras semanas se realiza una introducción progresiva en la que el profesor supervisa la confección de subrutinas sencillas que puedan añadirse posteriormente al proyecto. Transcurridas varias semanas los estudiantes suelen haber cubierto los requisitos mínimos, y puesto que comienzan a dominar el lenguaje ensamblador y su programa empieza a hacer algo llamativo, suelen dedicar voluntariamente un esfuerzo considerable a realizar algunas mejoras de las propuestas en el guion o sugeridas por ellos mismos.

Algunos de los proyectos propuestos son:

- *Reproducción de sonido usando PWM*. El objetivo es reproducir por el sencillo altavoz del PC (¡sin usar la tarjeta de sonido!) un archivo de sonido digitalizado, usando modulación por anchura de pulso (PWM) [9]. El programa controla el temporizador del PC y otros circuitos para ejecutar miles de veces por segundo una rutina de servicio de interrupción encargada de "reproducir" cada muestra de sonido. La dificultad de esta práctica queda recompensada cuando el estudiante logra oír su música favorita con calidad de radio AM en un dispositivo pensado para emitir pitidos. Entre las mejoras posibles cabe destacar por lo vistosas el dibujo de un medidor gráfico de la señal de audio o la adición de eco al sonido.
- *Trazado de líneas y aplicación a juegos*. Este proyecto consta de dos partes: en la primera se elaboran tres rutinas que tracen en la pantalla un número elevado de líneas rectas aleatorias utilizando tres algoritmos diferentes (el objetivo es comprobar cuál es más rápido); la segunda parte consiste en realizar un juego de frontón (Figura 5) o tenis usando una de las rutinas de la primera parte. Esta técnica de motivación basada en juegos también ha sido utilizada en [2] con resultados positivos.
- *Protector de pantalla*. Se trata de realizar un programa residente que apague la pantalla tras cierto período de inactividad, y vuelva a encenderla cuando se use el teclado o el ratón. Una de las mejoras consiste en incorporar alguna animación del tipo "juego de la vida" en lugar de dejar la pantalla negra.
- *Comunicación serie entre dos PC*. Consiste en realizar un programa para enviar y recibir

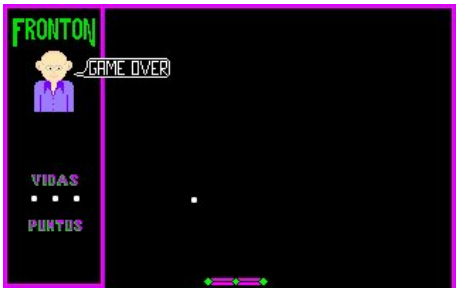


Figura 5. Juego de frontón realizado en ensamblador por dos estudiantes de EC1 en el curso 2001-2002.

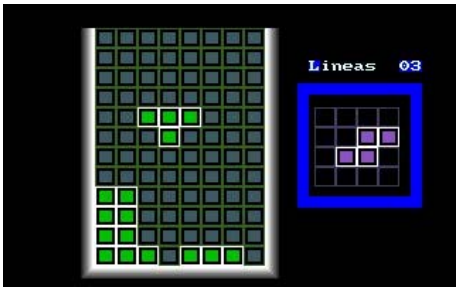


Figura 6. Juego del Tetris realizado en ensamblador por dos estudiantes de EC1 en el curso 2002-2003.

archivos por el puerto serie del PC utilizando el estándar XMODEM y un cable módem-nulo construido por los propios estudiantes.

Como se ha indicado antes, algunos estudiantes prefieren sugerir otro proyecto en el que tengan especial interés en lugar del que se les haya asignado. Algunos ejemplos de trabajos realizados son una calculadora residente que trabaja en diferentes bases, un juego del Tetris (Figura 6), o incluso un magnífico juego de naves espaciales.

3.3. Práctica de microprogramación

Se realiza en las asignaturas EC1 de II e IDC de ITIS. Se utiliza la tarjeta de evaluación Am29203 de AMD (Figura 7) y cuatro herramientas desarrolladas en nuestro departamento [12][7]:

1. *Editor de microinstrucciones*, que ayuda a crear un microprograma binario detallando los campos de cada microinstrucción.
2. *Microensamblador* (Figura 8), que permite escribir un microprograma en ensamblador.
3. *Monitor*, para interactuar con la tarjeta.

Figura 7. Tarjeta de evaluación Am29203 utilizada en prácticas de microprogramación.

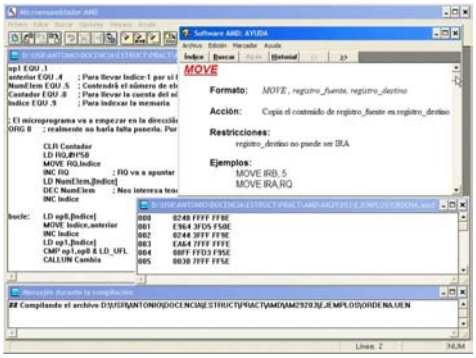


Figura 8. Microensamblador.

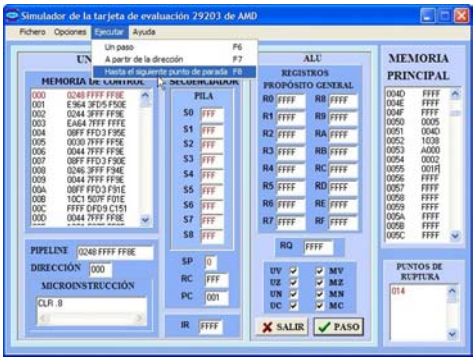


Figura 9. Simulador de la tarjeta Am29203.

4. *Simulador* de la tarjeta (Figura 9).

El estudiante realiza el siguiente trabajo:

1. Diseñar microinstrucciones binarias utilizando el editor de microinstrucciones.

2. Diseñar una arquitectura y su repertorio completo de instrucciones máquina.
3. Implementar ese repertorio de instrucciones usando el microensamblador.
4. Crear un programa para comprobar el correcto funcionamiento del repertorio diseñado primero en el simulador y después en la tarjeta real mediante el monitor.

En esta práctica lo estimulante es diseñar una arquitectura desde cero y comprobar que realmente funciona.

3.4. Práctica de E/S con PCLab y SoundBlaster

Un concepto esencial en las asignaturas relacionadas con Estructura de los Computadores es la comprensión de las tres técnicas fundamentales de E/S: programada, por interrupciones y mediante acceso directo a memoria (DMA). En la asignatura EC2 de II se realiza una práctica consistente en programar la digitalización de sonido, a bajo nivel y utilizando las tres técnicas mencionadas, en una tarjeta de adquisición de datos PCLab —estándar en plantas industriales—, o en una tarjeta de sonido SoundBlaster (en este caso no es posible usar la segunda técnica). Se utiliza el lenguaje C++, aprendido por los estudiantes en 1er curso. Esta práctica resulta tan atractiva o más que la de reproducción de sonido en ensamblador.

3.5. Práctica de E/S con 8051

Se realiza en las asignaturas EC2 de II, ASBM en las ITIG e ITIS, y PI en IE. Se utilizan dos

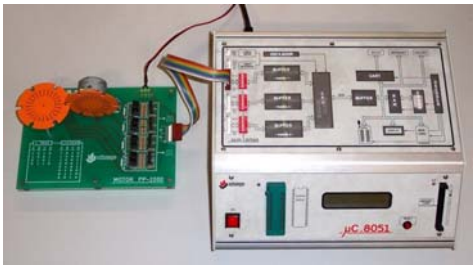


Figura 10. Equipo MCA-51, de Alecop [1], para prácticas de E/S con 8051, conectado a la tarjeta MOTOR PP-2000, con dos motores paso-a-paso.

equipos de desarrollo diferentes, dependiendo de la práctica concreta: el MCA-51 de Alecop [1] (Figura 10) y el SBC-51 de Dynalog Micro-Systems [6] (Figura 11). Se proponen las siguientes prácticas, de las cuales cada pareja realiza una o varias:

- Control de motores paso-a-paso (Figura 10)
- Control de semáforos (Figura 11 izda.)
- Control de impresora (Figura 11 dcha.)
- Control de dos ascensores
- Visualizador de matriz de puntos
- Conexión de un teclado externo
- Convertidor A/D: balanza electrónica
- Convertidor D/A
- Protocolo de comunicación I2C

En estas prácticas la motivación proviene del hecho de trabajar con sistemas casi reales. No es lo mismo simular, por ejemplo, el control de los semáforos de un cruce de calles, que observar realmente cómo éstos se encienden y apagan.

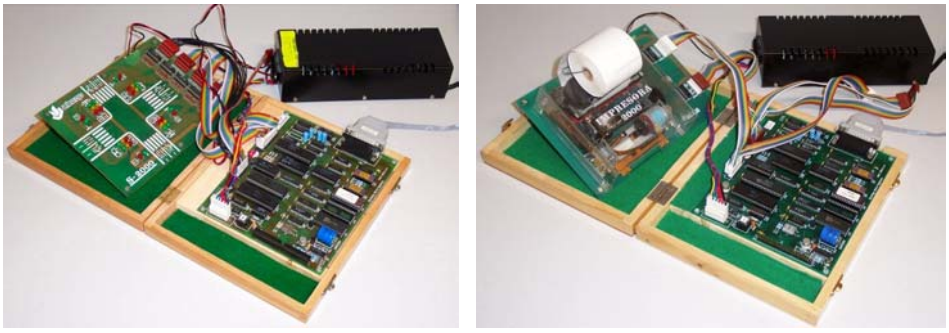


Figura 11. Equipos SBC-51 [6], de DMS, para prácticas de E/S con 8051, conectados a las tarjetas SEMAFORO-2000 (izda.) e IMPRESORA-2000 (dcha.) de Alecop.

3.6. Práctica con DSP

Esta práctica, que se realiza en ASBM en ITIG e ITIS, y PI en IE, consiste en la *Decodificación de sonido encriptado* [5]. El objetivo es decodificar en tiempo real un sonido procedente de una fuente encriptada, utilizando para ello un DSP y técnicas de tratamiento de señal. Al estudiante se le proporciona la señal de una cinta magnetofónica con sonido encriptado. A cada *kit* de desarrollo se le ha conectado un pequeño amplificador de audio con potencia suficiente para activar unos auriculares. Para descifrar el sonido se proponen cuatro formas de tratamiento de la señal codificada, cada una de las cuales produce una señal de salida con una fidelidad diferente. El estudiante desarrolla los cuatro algoritmos propuestos, verificando si el sonido reproducido corresponde a una señal perfectamente inteligible.

4. Conclusión

En el presente trabajo se ha descrito un conjunto de prácticas con hardware real que constituyen una poderosa herramienta motivadora en los estudiantes. Además, la nota adquirida en dichas prácticas les tranquiliza acerca del examen final (no se lo "juegan todo" en una única prueba).

Esta motivación puede incluso suscitar el interés por posibles asignaturas optativas que le mantengan en contacto con el área, facilitando probablemente la superación de las correspondientes troncales de cursos superiores.

Referencias

[1] Alecop, S. Coop.: "MCA-51: Manual de Usuario V1.1". Mondragón (Guipúzcoa), enero 1992. http://www.alecop.es

[2] Bueno, D., Garralón, J.; Pérez, J. M.; Maña, A.: "Aprendizaje Lúdico en Laboratorio de Programación", VII JENUI, pp. 346-350, Palma de Mallorca, 2001.

[3] Cañas, A.: "Enlaces sobre montaje de un PC", http://atc.ugr.es/~acanas/cad/#pc-build

[4] Cañas, A.; Díaz, A. F.; Rodríguez, M.; Bernier, J. L.; Prieto, A: "Development and Evaluation of a web-based tool to support university education and administration", Inf. Society and Education: Monitoring a Revolution. Serie Sociedad de la Educación nº. 9, tomo I, pp. 473-477, 2002.

[5] Díaz, A. F.; Rodríguez, M.; Parrilla, L.; Puntonet, C.: "Decodificación de sonido encriptado mediante la utilización de un procesador de señales digitales (DSP)", SAAEI'98, Univ. Pública de Navarra, pp. 127-128, 1998.

[6] Dynalog Micro-Systems: "SBC-51 User's Manual". http://www.dynalogindia.com/

[7] Enríquez de Luna, I.: "Simulador de un computador microprogramable", proy. fin de carrera Ing. Electrónica, Univ. Granada, 2002.

[8] González, E. J.; Alayón, S.; Piñeiro, J. D.; Sánchez, J. L.; Estévez, J. I.; Moreno, L.: "Combinación de Prácticas de Simulación y Reales para un Curso Anual de Arquitectura de Computadores", VII JENUI, pp. 80-85, Palma de Mallorca, 2001.

[9] Heidenstrom, K.: "FAQ/Application notes: Timing on the PC family under DOS", 1995. http://atc.ugr.es/~acanas/cad/programming/PC/PCTIM003.zip

[10] Miklós, T.: "AIDA-32: Worldwide Sysinfo Tool", http://www.aida32.hu/

[11] Pilgrim, A.: "Build Your Own Pentium III PC", 2nd Ed. Osborne McGraw-Hill, 1999.

[12] Pomares, H.; Rojas, I.; Prieto, A.; Gómez, F.; Cañas, A.: "A didactic environment for the study of microprogrammed processors", Inf. Society and Education: Monitoring a Revolution, Serie Sociedad de la Educación nº. 9, tomo I, pp. 17-22, 2002.

[13] Rebollo, M.: "Aprendizaje activo en el aula", VII JENUI, pp. 137-142, Palma de Mallorca, 2001.

[14] Rosenthal, M.: "Build your own PC", 3rd. Ed. McGraw-Hill, New York, 2002.

[15] Valero-García, M.: "Cómo conseguir que los alumnos hagan más ejercicios". VIII JENUI, pp. 343-349, Cáceres, 2002.

[16] Vicerrectorado de Planificación, Calidad y Evaluación Docente, Univ. de Granada: "Innovación Docente en la Universidad de Granada". http://www.ugr.es/~vic_plan/innovacion/innovacion.html

Automatización de prácticas en entornos masificados

Antonio García Dopico
Dpto. de Arquitectura y Tecnología de Sistemas Informáticos
Universidad Politécnica de Madrid
28660 Boadilla del Monte
e-mail: dopico@fi.upm.es

Santiago Rodríguez de la Fuente
Dpto. de Arquitectura y Tecnología de Sistemas Informáticos
Universidad Politécnica de Madrid
28660 Boadilla del Monte
e-mail: srodri@fi.upm.es

Francisco Javier Rosales García
Dpto. de Arquitectura y Tecnología de Sistemas Informáticos
Universidad Politécnica de Madrid
28660 Boadilla del Monte
e-mail: frosal@fi.upm.es

Resumen

La docencia en el área de Arquitectura de Computadores tiene una fuerte componente práctica, que se aborda a través de la realización de múltiples prácticas por parte de los alumnos. En la Facultad de Informática de la Universidad Politécnica de Madrid (UPM) estas prácticas se tienen que realizar para un elevado número de alumnos, que oscila de 500 a 700 por asignatura. El afán de realizar prácticas de calidad para todos estos alumnos ha obligado a desarrollar un entorno automatizado de prácticas cuyos componentes principales se presentan en este artículo.

1. Introducción

En general, la docencia se compone de una parte teórica y de otra práctica. La parte práctica es imprescindible para afianzar los conocimientos explicados en la teoría, esto se consigue enfrentado al alumno a un problema concreto que debe resolver, lo que le obliga a aplicar los conocimientos explicados a lo largo de la asignatura. Con frecuencia, a los alumnos le surgen dudas y problemas, en cuyo caso acuden a los profesores de prácticas para que les ayuden a resolverlos. Este trato directo con los profesores es positivo y hace más humana la docencia.

Desafortunadamente hay situaciones en las que no es posible el funcionamiento antes expuesto. Por ejemplo, en entornos de educación a distancia o allí donde el número de alumnos es muy elevado. En ambos casos, la posibilidad de un trato directo se ve mermada por el contexto en que se desarrollan las prácticas.

En el Departamento de Arquitectura y Tecnología de Sistemas Informáticos (DATSI) de la Facultad de Informática de la UPM, se tiene una media de 500 a 700 alumnos por asignatura troncal u obligatoria. Como se desean realizar varias prácticas a lo largo de las distintas asignaturas, la masificación representa un problema importante que hay que abordar.

Las prácticas propuestas no son de resolución trivial, dado que tratan en profundidad temas de cierta complejidad para los alumnos. Cada una es un pequeño proyecto, con una especificación, que se entrega al alumno y cuyo contenido se explica en clase. Esto, unido a que se exige la correcta realización de todas las prácticas, comprobando para cada una de ellas que se obtienen los resultados correctos, da una idea de la cantidad de trabajo que esto representa. La simple comprobación de todos los resultados sería una tarea casi imposible de abordar manualmente, ya que cada alumno dispone de varias oportunidades para su correcta realización. Esto ha obligado a automatizar una parte importante de la gestión y seguimiento de las prácticas.

La experiencia docente del departamento en la impartición de las prácticas permitió observar que los trabajos que los alumnos entregaban como prácticas completas adolecían de numerosas deficiencias:

- El alumno no había comprendido la especificación de la práctica.
- El alumno ni siquiera se había planteado el establecimiento de una batería de prueba que permitiera comprobar su trabajo de forma sistemática.
- El alumno no aprendía a depurar su código.

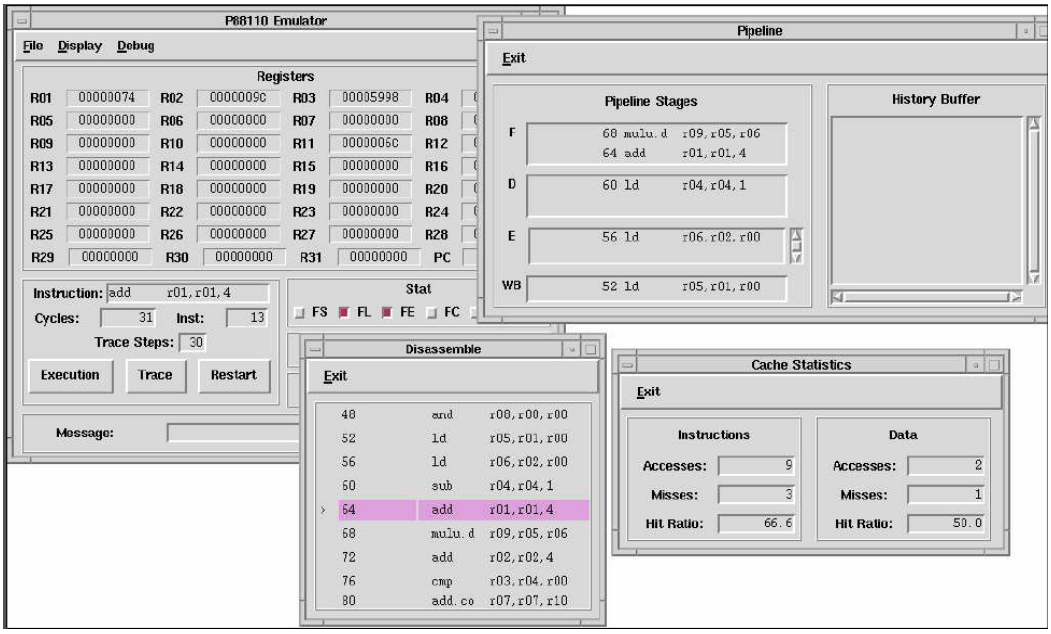


Figura 1. Interfaz gráfica del mc88110

Aunque la automatización reduce el contacto entre alumnos y profesores, en su momento se consideró que no había alternativas factibles a la misma. Las alternativas que se tenían: hacer menos prácticas, que fuesen mucho más sencillas o no comprobar su correcto funcionamiento, se consideraron poco didácticas, ya que todas ellas repercuten en un menor aprendizaje por parte de los alumnos.

Por tanto, se ha preferido mantener un alto número de prácticas, de complejidad media e incluso alta, y comprobar que todos los alumnos obtenían los resultados deseados en los tres apartados anteriores.

2. Prácticas en el DATSI

Como muestra del conjunto de prácticas de que se está hablando se enumeran a continuación algunas de ellas.

En el grupo de arquitectura se proponen las siguientes prácticas para su realización en grupos de dos alumnos:

- En Estructura de Computadores una práctica de microprogramación, en el tema de la unidad de control. Emplea un simulador basado en el i8080, el `p8080e`.
- En Laboratorio de Estructura de Computadores una práctica de ensamblador, dentro del tema de programación en ensamblador. Se emplea un simulador del mc88110 [4], cuya interfaz puede verse en la figura 1.
- En Arquitectura de Computadores hay dos prácticas, una de entrada/salida, en su tema correspondiente, y otra de memorias caché, en el tema de jerarquía de memoria. En la primera práctica se emplea un simulador del mc68000, el BSVC [3], cuya interfaz puede verse en la figura 2. En la segunda se emplea el simulador del mc88110 ya mencionado [5].

En el grupo de sistemas operativos las prácticas son individuales y se proponen las siguientes:

- En Fundamentos de la Programación de los Sistemas Operativos varias prácticas de programación en C.
- En Sistemas Operativos una práctica de *shell scripts* y otra de implementación de un pequeño intérprete de mandatos.

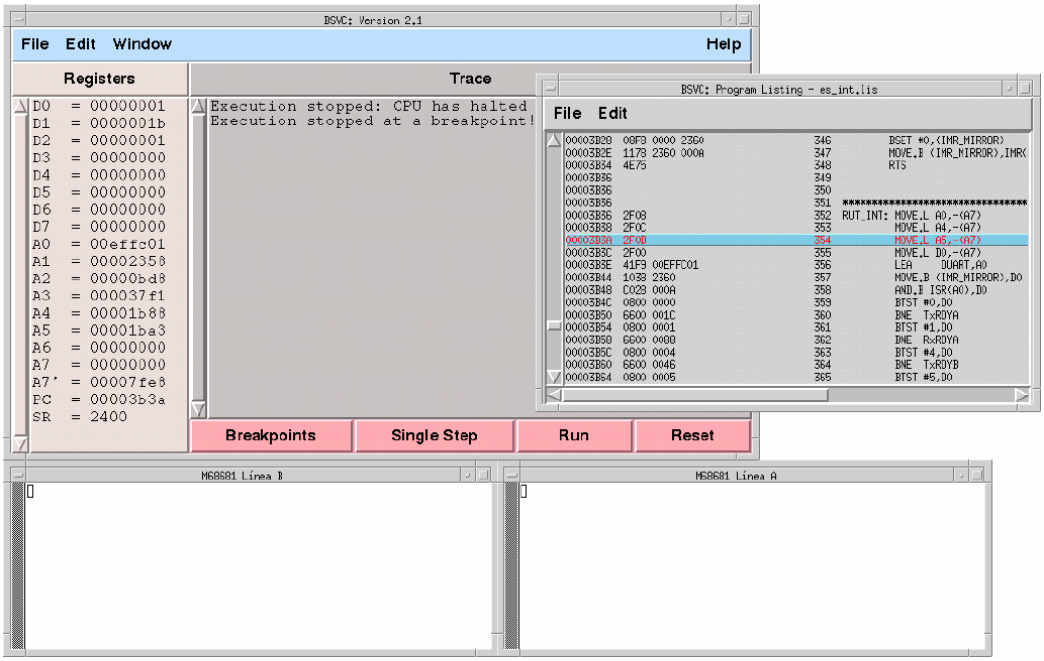


Figura 2. Interfaz gráfica del BSVC

- En Diseño de los Sistemas Operativos hay una práctica de análisis del comportamiento del gestor de memoria del S.O. y otra en la que se incorpora funcionalidad a un pequeño núcleo de sistema operativo.

3. Entorno de prácticas

Para posibilitar la realización de estas prácticas en el contexto descrito ha sido preciso el desarrollo de un entorno de prácticas, compuesto por un conjunto de herramientas y métodos para la automatización de la gestión de las mismas. Básicamente, este entorno de prácticas se basa en:

- Enunciados y documentación en la Web, así como en formato papel. Esto le permite a los alumnos acceder a la documentación de forma remota.
- Tutorías electrónicas vía e-mail, aunque, por supuesto, siguen atendiéndose tutorías presenciales. Así, aunque el alumno esté trabajando a distancia, se le resuelven las dudas que le surjan, ahorrándole tiempo. Esta vía de comunicación tiene la ventaja de que permite crear un archivo con las preguntas más frecuentes (FAQ). Este archivo ahorra muchas preguntas similares.
- Entregador de prácticas remoto. Permite a los alumnos enviar su práctica desde cualquier sitio, sin necesidad de desplazarse, así como consultar también de forma remota los resultados obtenidos tras la corrección.
- Corrector automático. Ejecuta una batería de pruebas sobre cada práctica de cada alumno o grupo. El objetivo del corrector es producir un resultado que proporcione al alumno información suficiente para determinar el origen del error detectado y sugerirle alternativas para su solución. De este modo los alumnos pueden ir corrigiendo su práctica sin acudir sistemáticamente al profesor.
- Detector de copia de prácticas. Asiste al profesor en una labor que manualmente sería imposible: detectar sobre el total de prácticas entregadas qué parejas contienen código plagiado una de la otra.

A continuación se describen en mayor profundidad el entregador, el corrector y el

detector de copias, que son la parte central de este entorno de prácticas.

4. Sistema de entrega remota

Permite que el alumno se conecte desde cualquier sitio a las máquinas del departamento. Para dicha conexión debe identificarse mediante un *login* y un *password*.

El entregador dispone de una lista con todos los alumnos que pueden realizar una práctica concreta. Para evitar accesos indeseados, sólo dichos alumnos podrán darse de alta en el sistema de entregas.

Una vez conectado, se puede enviar la práctica, consultar los resultados obtenidos en correcciones anteriores, ver las noticias más recientes relativas a esa práctica o anular alguna entrega realizada incorrectamente. A continuación se muestra el menú del entregador:

```
Practica de Lab. Estr. de Computadores
Septiembre 2002
Servidor de Practicas. Version 1.9.5
Introduzca su login (matricula): k0267
Password:

        Sesion iniciada correctamente

        Practica de Lab. Estr. de
Computadores (88110).
                Septiembre 2002

                OPCIONES:

        1. Mandar Ficheros.
        2. Consultar Resultados.
        3. Cancelar Entregas.
        4. Bloquear la Entrega.
        5. Ayuda !!!!
        6. Noticias. (11/03/02 18:17)
        q  Abandonar
>>
```

Figura 3. Menú del entregador

De cara al profesor encargado de la administración de la práctica, se guarda un histórico con todas las conexiones realizadas, detallando desde dónde, cuándo y quién la hizo, a modo de auditoría. También se almacena todo lo que los alumnos hacen una vez conectados. Dicho histórico se muestra en la figura 4.

Además, por cada alumno, se guardan todas sus entregas, incluso las que luego anula. De cada entrega realizada correctamente se guarda el resultado obtenido, en qué falla y por qué.

El hecho de guardar tanta información sobre lo que hace cada alumno se ha mostrado muy útil, ya que al ser un número tan alto de alumnos siempre hay algunos que se quejan por causas dispares. Por ejemplo, alumnos que afirman que le no se le ha corregido la práctica, cuando en realidad anuló su entrega, o que dicen disponer de correcciones cuando realmente las han agotado ya. La información recogida en este archivo histórico permite determinar qué sucedió en realidad.

5. Corrector automático

Recibe una batería de pruebas que ejecuta sobre todas las prácticas y para cada una de ellas almacena en un fichero qué casos pasan correctamente y cuáles fallan. Posteriormente, el alumno puede consultar, a través del entregador, dicho fichero para recoger los resultados de la corrección.

Cada prueba consiste en ejecutar la práctica del alumno con unos datos de entrada esperando obtener unos resultados concretos. En el grupo de sistemas operativos las prácticas se ejecutan directamente sobre máquinas Unix como un proceso. Pero en el caso del grupo de arquitectura, debido a los problemas que pueden surgir si se ejecutan las prácticas directamente sobre una máquina real, se emplean simuladores.

En cualquier caso el corrector automático es suficientemente flexible para poder tratar con diversos lenguajes de programación, ya sea ejecutando la práctica directamente sobre la máquina o sobre un simulador.

Todo lo que necesita es que la ejecución sea en modo *batch*. Es decir, en caso de ejecutar directamente en Unix será:

```
Practica < datos > resultados
```

En caso de utilizar un simulador será:

```
Simulador < practica+datos > result
```

El alumno tiene un número limitado de correcciones a su disposición, que establece el profesor de esa práctica. Esto es así para evitar que el alumno use el corrector como un depurador, ya que se deben alcanzar los siguientes objetivos:

- Comprensión de la especificación del trabajo que debe realizar el alumno.
- Desarrollo de aplicaciones en grupo.
- Diseñar baterías de pruebas para probar su propio trabajo.

```
04 Nov 2002 12:33 f980769 ALTA(ERROR) Cancelada [batman.fi.upm.es]
04 Nov 2002 12:33 f980769 ALTA(OK) [batman.fi.upm.es]
04 Nov 2002 12:33 f980769 LOGIN(OK) [batman.fi.upm.es]
04 Nov 2002 12:34 f980769 NEWS(OK)
04 Nov 2002 12:34 f980769 LOGOUT(OK)
05 Nov 2002 09:32 a920157 LOGIN(ERROR) Password erróneo [batman.fi.upm.es]
05 Nov 2002 09:32 a920157 LOGIN(OK) [batman.fi.upm.es]
05 Nov 2002 09:32 a920157 ENTREGA(ERROR). Fichero practica.s Error en formato
05 Nov 2002 09:32 a920157 CANCELACION(OK) Error en entrega
05 Nov 2002 14:13 960016  LOGOUT(OK)
05 Nov 2002 14:14 960016  LOGIN(OK) [batman.fi.upm.es]
05 Nov 2002 14:14 960016  ENTREGA(OK)
05 Nov 2002 14:14 960016  LOGOUT(OK)
```

Figura 4. Fichero histórico

- Aprender los métodos para depurar la práctica.

Cada vez que un alumno quiera corregir su práctica, para conocer qué fallos tiene aún, bastará con que envíe los ficheros de la misma mediante el entregador. Quedará registrado que dicho alumno ha hecho una entrega y cuando de nuevo se ejecute el corrector se corregirá su práctica.

Si el alumno detecta algún fallo antes de la corrección siempre puede anular la entrega, para evitar que le cuente dicha corrección.

El corrector ejecuta un número fijo de veces al día, a unas horas concretas, que establece el profesor y son conocidas por los alumnos.

El profesor puede lanzar cuando quiera alguna corrección extra. Incluso puede decidir si quiere que cuente o no sobre el número total de correcciones que tiene cada alumno.

Cada práctica tiene una batería de casos de prueba. Cada caso de prueba consta de cuatro ficheros:

- Prueba.cf: Configuración del caso de prueba. Indica en qué fijarse tras la ejecución, dónde estarán los resultados y en cuáles se debe centrar la atención.
- Prueba.ens: Fichero fuente, que contiene los datos con los que se invocará la práctica del alumno.
- Prueba.ofl: Resultado correcto, lo que debería dar la práctica del alumno tras la ejecución.
- Prueba.res: Mensaje de error que recibe el alumno. Le indica qué ha fallado y le proporciona información para corregir su error.

Una vez que el alumno ve sus fallos, intenta corregirlos con la información que le proporciona el corrector. Si no es capaz puede preguntar al profesor, ya sea personalmente o por correo electrónico. Muchas veces, cuando sus problemas se deben a que no entienden conceptos importantes es preferible el trato directo y acuden a las tutorías.

El conjunto de prácticas se pueden englobar en dos grupos:

- La mayor parte de ellas son prácticas funcionales en las que la calidad de los resultados se mide en función de si son conformes a la especificación dada sin hacer especial hincapié en las prestaciones proporcionadas por el trabajo del alumno.
- En el segundo tipo, al que pertenece la práctica de jerarquía de memoria, se exige que el alumno modifique un programa para que proporcione mejores prestaciones (menor número de accesos a memoria, menor número de fallos de cache, etc.). En este caso las herramientas (corrector de prácticas) deben ser capaces de comparar el resultado obtenido por el alumno con el peor de los resultados admisibles.

6. Detección de copias

El alto número de alumnos que realizan cada práctica tiene otra consecuencia inevitable aparte de las ya vistas. Aumenta mucho la probabilidad de que los alumnos se dejen las prácticas unos a otros, confiados en que el alto número de prácticas hará casi imposible detectar los casos de copia.

Para evitar que algunos alumnos aprueben las prácticas gracias al esfuerzo de otros, se ha incorporado a este entorno un detector de copias.

Esta herramienta hace una búsqueda exhaustiva, es decir, compara cada práctica con

todas las demás. Para un alto número de alumnos pueden generarse miles de comparaciones, pero los recursos actuales permiten realizar toda esta tarea en pocas horas.

El detector de copias sirve para diferentes lenguajes de programación, incluso para ensamblador o prácticas de microprogramación. Para ello, en una fase inicial procesa cada fichero de entrada identificando ciertos patrones predefinidos (por ejemplo, palabras reservadas del lenguaje) componiendo con ellos una firma o resumen de cada práctica. De esta forma, se obvian cambios elementales, como renombrado de variables o funciones, y se centra el estudio principalmente en la estructura lógica del programa, que en definitiva es aquello que quien copia no sabe cambiar.

A continuación se compara cada una de las firmas con todas las demás, evaluándose cuatro criterios distintos de similitud por cada pareja. Estos criterios cuantifican secuencias de patrones repetidos independientemente de su posición relativa. Esto permite la detección de casos en los que sólo se copia una rutina o un fragmento de la misma, incluso si se ha movido el código respecto del original.

El resultado producido es un archivo tal que, ordenado según cada uno de los criterios de similitud, permite determinar qué parejas de prácticas son más parecidas total o parcialmente.

Debido a la gravedad del tema, esta herramienta debe considerarse sólo un asistente a la detección de copias. La decisión final la han de tomar los profesores responsables, inspeccionado detenidamente cada posible caso de copia.

7. Evaluación remota con Aulaweb

Como extensión al entorno de prácticas visto, y siguiendo una filosofía similar, se ha empleado en la asignatura de Sistemas Operativos una herramienta de evaluación remota [6] a través de Internet denominada Aulaweb, desarrollada en la Escuela de Ingenieros Industriales de la UPM [1].

Esta herramienta permite programar exámenes periódicos que los alumnos deben resolver, conectándose a la misma vía Internet. A pesar de

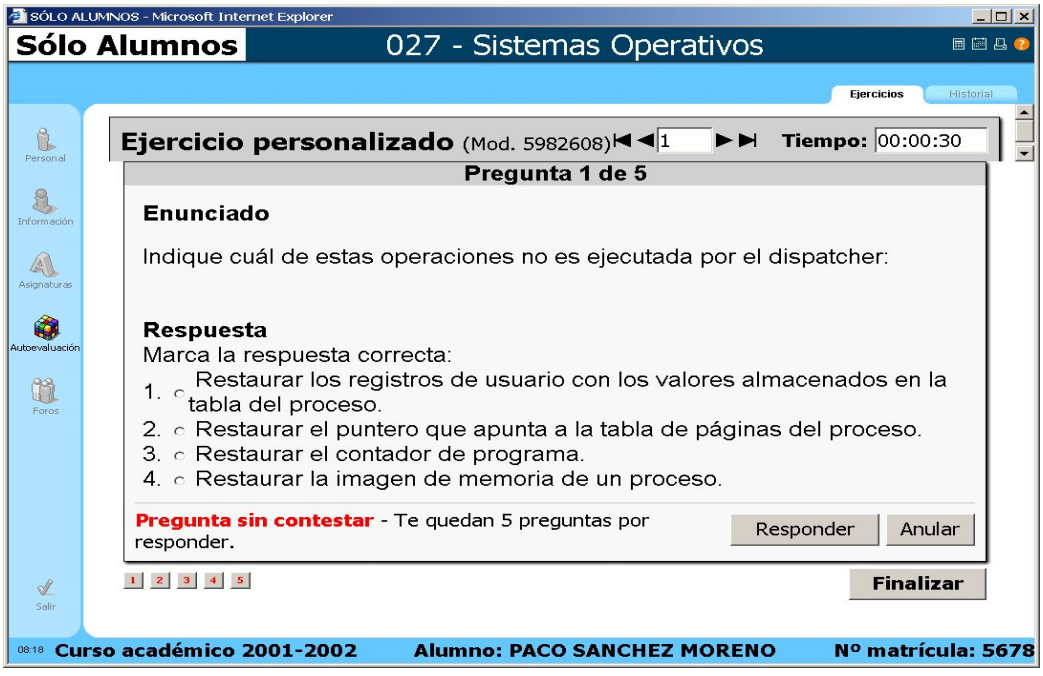


Figura 5. Examen de Aulaweb

poder tener los apuntes y libros delante, este método les obliga a llevar la materia al día y a estudiar de forma continuada.

Dado que a la hora de hacer el examen no se tiene ningún control presencial, no se puede asegurar la identidad del alumno ni evitar que compañeros suyos le ayuden. Por ello, estos exámenes tienen poco peso en la nota final, aunque deben realizarse obligatoriamente para poder acceder al examen final de la asignatura.

Cuando un alumno se conecta a través de un navegador a esta herramienta, se construye automáticamente un nuevo examen para dicho alumno. Se obtienen de forma aleatoria "n" preguntas de una gran base de datos de preguntas de la asignatura. El número y tipo de las preguntas es configurable. En la figura 5 se muestra una pregunta tipo test de esta herramienta.

De esta forma cada alumno se enfrenta a preguntas diferentes, por lo que no le sirve de nada mirar como un compañero resuelve otro examen. Se permite que un alumno repita su examen si no está satisfecho con la nota obtenida, ya que el objetivo no es tanto evaluarlo (la nota cuenta muy poco) como obligarle a estudiar. Si repite el examen estará profundizando más en la materia.

8. Conclusiones

Se ha presentado el entorno de prácticas que se emplea en el Departamento de Arquitectura y Tecnología de Sistemas Informáticos de la Facultad de Informática de la Universidad Politécnica de Madrid. Así mismo se han presentado sus componentes principales: un entregador remoto, un corrector automático y un detector de copias.

Este entorno de prácticas se ha mostrado apto para entornos masificados, donde un pequeño grupo de profesores tiene que llevar varias prácticas, cada una de ellas con un elevado número de alumnos.

Gracias a este entorno se ha podido mantener un alto número de prácticas, de complejidad media e incluso alta, y comprobar que todos los alumnos obtenían los resultados deseados. Esto obligó a automatizar parte de la gestión y mantenimiento, para agilizar el proceso y disminuir la carga de trabajo de los profesores.

El entregador remoto se ha mostrado muy útil tanto para los alumnos, que pueden realizar la práctica en cualquier lugar, como para los profesores, ya que permite centralizar y auditar toda la actividad que los alumnos realizan con sus prácticas.

El corrector automático permite diseñar prácticas complejas que pueden ser verificadas en todos sus detalles. Los alumnos no sólo deben realizar la práctica sino que esta debe funcionar correctamente según las especificaciones entregadas. El corrector automatiza esta verificación, generando información sobre los fallos detectados. Esto permite a los alumnos corregir sus errores de modo autónomo, sin precisar una ayuda constante del profesor.

La experiencia nos ha demostrado que siempre hay un porcentaje de alumnos que copia las prácticas. El detector de copias se ha mostrado tan efectivo en la detección de estos casos que incluso ha disuadido a muchos alumnos de su intención. Esto lo demuestra el hecho de que el porcentaje de alumnos que copian se ha reducido considerablemente en los últimos años.

En definitiva, la experiencia nos ha demostrado la utilidad de este entorno, ya que, a pesar de la masificación en que se desarrollan las prácticas, éstas siguen cumpliendo correctamente su cometido original: afianzar los conocimientos teóricos básicos que deben tener los alumnos. Esto lo demuestra el hecho de que un elevado porcentaje de alumnos pasan todas las pruebas satisfactoriamente, asegurándonos que han aprendido los conceptos en los que se apoya la práctica. Para confirmar esta conclusión se muestran unos cuantos datos de dos prácticas concretas:

- Práctica de ensamblador de Laboratorio de Estructura de Computadores en la convocatoria de junio de 2002. El corrector empleaba una batería de 54 pruebas, que debían pasarse en su totalidad.
  - 90% de aprobados entre los grupos que empezaron en fecha (64 grupos aprobados de 71)
  - 50% de aprobados entre los grupos que empezaron tarde (43 aprobados de 86)
  - En total 68% de aprobados (107 de 157).
- Práctica de entrada/salida de Arquitectura de Computadores en la convocatoria de febrero

de 2003. El corrector empleaba una batería de 45 pruebas, que también debían pasarse en su totalidad.

- 79% de aprobados entre los grupos que empezaron en fecha (112 grupos aprobados de 142)
- 49% de aprobados entre los grupos que empezaron tarde (36 aprobados de 73)
- En total 69% de aprobados (148 de 215).

En definitiva, este entorno se ha mostrado muy adecuado para tratar con el problema de la masificación, consiguiendo que los alumnos sean mucho más autónomos de lo que antes eran, a pesar de la complejidad y variedad de las prácticas propuestas.

El hecho de que varías prácticas, muy distintas entre sí, empleen este entorno, muestra su flexibilidad. De la misma forma que se emplea para prácticas con shell scripts, C, ensamblador, entrada/salida o microprogramación, se podría emplear para prácticas con Ada, Java o cualquier otro lenguaje de programación.

Por último cabe mencionar que la mayoría de las herramientas aquí mencionadas son accesibles vía Web (em88110, BSVC, Aulaweb) aunque algunas no lo son ya que no pueden estar accesibles a los alumnos, como puede ser el detector de copias (podrían intentar camuflar sus copias) o el corrector automático, ya no desarrollarían su propia batería de pruebas y usarían la del corrector. Para mayor información de dichas herramientas contacte vía e-mail con los autores del artículo.

Referencias

[1] García-Beltrán, A.; Martínez, R. *The role of self-assessment in Aulaweb e-learning system.* European Distance Education Network, Annual Conference, Granada, Spain, junio 2002.

[2] *MC88110: Second Generation RISC Microprocessor. User's Manual.* Motorola Inc. 1991.

[3] Mott, B.W. *BSVC A microprocessor simulation framework.* http://www.redlinelabs.com/bsvc

[4] Pérez, J.M.; Rodríguez, S.; Méndez, R.; García M.I. *The em88110: Emulating a superscalar processor.* ACM SIGCSE Bulletin, 29(4): 45-50, Septiembre 1997

[5] Rodríguez de la Fuente, S.; García Clemente, M.I.; Méndez Cavanillas, R. *Teaching Computer Architecture with a New Superscalar Processor Emulator.* Proceedings of the 4th annual SIGCSE/SIGCUE Conference on Innovation and Technology in Computer Science Education, ITiCSE'99, 99-102, Cracow, Poland.

[6] Sánchez Moreno, F.M.; García Dopico, A. *Experiencia docente con Aulaweb en la asignatura de Sistemas Operativos.* XIII Jornadas de Paralelismo, 87-90 Lleida, 2002.

Atención a la diversidad (género, discapacidad, proyección social)

Aspectos de Género y Enseñanza de la Informática en Alemania

Dr. Esther Ruiz Ben
Institut für Informatik und Gesellschaft
Albert-Ludwigs-Universität Freiburg
Friedrichstr. 50
79098 Freiburg i. Br.
http://mod.iig.uni-freiburg.de
e-mail: ruizben@modell.iig.uni-freiburg.de

Resumen

En este artículo se comentan algunos aspectos de género interrelacionados con la ensenanza de la informática a nivel universitario en el contexto de la profesionalización de la informática en Alemania. Asimismo se comentan las medidas que se han tomado a nivel estatal y por parte de las universidades, especialmente en el departamento de Modelación y consecuencias sociales del Instituto de Informática y Sociedad de la Universidad de Freiburg para incrementar la participación de las mujeres en este ámbito.

1. Introducción

Con la comercialización y diversficación de internet durante los años 90 la ya creciente demanda de especialistas en informática se incrementa a un ritmo que las universidades no son capaces de responder. Los contenidos curriculares se muestran cada día más obsoletos respecto al ritmo de innovación industrial y el número de graduados no es suficiente para cubrir las necesidades del mercado de trabajo en el sector. Como consecuencia existe una presencia de trabajadores en el sector del software sin formación académica de tercer grado en informática ("side step workers") del 60% [5]. Numerosos autores hablan de la desprofesionalización que este proceso ha traido consigo, dejando a un lado la consolidación de la informática como profesión con eco a nivel académico y de planes de estudio [8,3]. Sin embargo, dados los actuales síntomas de normalización y especialización en la demanda de trabajadores derivada de la enorme crisis que sufre el sector informático y caracterizada por la mayor exclusividad de adquisición de personal altamente cualificado, especialmente absolventes de informática, [7] habla de una tendencia hacia la profesionalización.

Si bien varias crisis en el sector del software, como la denominada "software crisis" de los 80 caracterizada por la dificultad de comunicación entre especialistas informáticos y usuarios/as o la "crisis de proyecto" derivada de la creciente complejidad de los proyectos, ya habia sido diagnosticada por diversos autores en años anteriores [17], dada la enorme dispersión de las aplicaciones y la necesidad en la gran mayoría de las ramas industriales de soluciones informáticas orientadas a la participación en internet, la crisis del software queda matizada actualmente por una enorme aceleración de la producción. La demanda de cualificaciones por parte del sector informático queda determinada por esta evolución, de manera que dependiendo de la rama de producción en la que nos situemos, una mayor o menor competencia técnica es necesaria por parte de los trabajadores/as, combinada con una mayor o menor habilidad social y comunicativa. Desde una perspectiva general, es importante tener en cuenta que flexibilidad y capacidad de adaptación relativa a una necesidad continua de renovación de conocimientos es la característica más valorada por las empresas a la hora de contratar nuevos empleados. Este factor viene acompañado por la juventud de la mano de obra del sector, debido a la confianza de las empresas en su competencia para impulsar la innovación productiva y a su menor coste. Por otro lado, dada la relativa juventud de la informática como profesión, y

concretamente en el caso de Alemania, no existen standards de cualificación que sean reconocidos ampliamente y suscritos por asociaciones profesionales consolidadas, ya que el enorme número de trabajadores del sector procedentes de otras disciplinas académicas y la enorme diversificación de la presencia de la informática en la mayor parte de los sectores industriales impide la homogeneización profesional. Cuáles son las estrategias que se han seguido en Alemania para confrontar esta situación de falta de mano de obra en el sector informático? Por un lado, a nivel oficial se organiza en el año 2000 el programa "Green-Card" para la adquisición de trabajadores extranjeros de paises extra-comunitarios altamente cualificados en informática. Paralelamente se inician en el año 97 diversos planes de estudio en academias profesionales orientados a formar personas con diversas cualificaciones como especialistas en informática. Asimismo, las empresas de mayor tamaño optan por el establecimiento de programas internos de formación continua, con el fin de adaptar su fuerza de trabajo a las demandas del mercado. A nivel académico, las universidades reaccionan lentamente intentando adaptar sus programas académicos y sobre todo apoyando una mayor orientación práctica, asi como la creación de títulos "Bachelor" de duración más reducida que los tradicionales "Diplom". Por otro lado, de manera más concreta y reducida, la antigua GMD (Gesellschaft für Mathematik und Datenverarbeitung) y el Fraunhofer Institut se fusionan creando uno de los centros más prestigiosos en el país de transferencia de conocimientos entre la industria informática y el ámbito académico, principalmente por medio de la Universidad de Bonn.

Por otra parte, otra de las estrategias para paliar la demanda de especialista informáticos en los pasados años fue intentar atraer a las mujeres hacia este ámbito en el que en Alemania representan una reducida minoría. Numerosos programas a nivel académico o de adaptación profesional se llevan a cabo por parte estatal y también en cooperación con grandes empresas del sector (IBM, Alcatel, deutsche Telekom) [12].

Desde los años 80, momento en el que se consolidan los primeros planes de estudio de informática en las universidades alemanas, la participación de las mujeres en estos estudios se ha reducido desde alcanzar en 1987 casi un 30% hasta la presencia actual de cerca de un 10%. En el ámbito laboral se refleja también esta situación minoritaria: según datos del "Mikrozensus" tan solo un 14% de trabajadores de la rama de creación de software son mujeres [12]. Aspectos estructurales derivados de la situación en el sector, sobre todo relacionados con la dificultad de combinar las responsabilidades familiares con la vida laboral, asi como simbólicos, implícitos en la cultura académica de la informática, en la que prevalecen aspectos técnicos tradicionalmente asociados con la masculinidad [15,18] constituyen el origen de esta situación.

En las siguientes páginas ofrecemos un breve repaso de la situación de la informática a nivel académico en Alemania considerando sus implicaciones profesionales y la atención a aspectos de género dentro de la disciplina.

2. La Enseñanza universitaria de la Informática en Alemania y su orientación profesional

La informática como disciplina académica de tercer grado se imparte en Alemania en Facultades o en Escuelas Universitarias. En facultades de Informática la carrera se divide en 13 semestres y en ella se ofrecen las siguientes ramas:

- Informática general,
- Informática técnica o ingeniería informática,
- Informática teórica,
- Informática aplicada.

En general los estudios de informática en las facultades universitarias se hallan más influidos por su destacada exigencia teórica, es decir matemática. Precisamente a esta exigencia atribuye la iniciativa industrial y estatal D21 la elevada cuota de abandono de los estudios, que supera en esta disciplina el 50% [9]. Por otro lado, se critica también la falta de orientación hacia la praxis profesional y hacia la actual evolución productiva en los diversos sectores industriales en los que interviene la informática, asi como el exceso de contenidos teóricos irrelevantes y la excesiva presencia de

especialistas o en otras palabras la escasez de generalistas[1].

En las Escuelas universitarias los estudios tienen una duración de 8 a 10 semestres con dos semestres adicionales obligatorios de prácticas, que por lo general suele ser la puerta de entrada en el mundo laboral para graduados/as. El contenido curricular se diferencia del facultativo principalmente por su concentración en aspectos prácticos de la informática y la menor exigencia de contenidos teóricos. Las ramas que se ofrecen en estas instituciones son similares a las de las universidades con la diferencia de que en las Escuelas universitarias existen una oferta más especializada. De este modo se ofrecen especialidades como informática médica, producción multimedia, u otras denominadas "informática de guión" ("Bindestrichinformatik"), orientadas hacia la combinación de contenidos informáticos y de otras disciplinas (medicina, biología, geografía, etc.). Este tipo de estudios son muy demandados por parte de la industria, sobre todo en empresas de tamaño medio, en las que la menor complejidad de los productos en comparación con los de las empresas grandes, los requerimientos de cualificación son más modestos.

Desde el punto de vista teórico se pueden distinguir tres amplias tendencias, con diferente peso en el terreno académico de la informática:

- Orientación hacia la Ingeniería. Centrada en el desarrollo de productos sobre una base científica y con el fin de la optimización de costes, calidad y tiempo invertido [1].
- Matemático-formal. Orientada hacia la formalización y resolución "correcta" de problemas [4] rechazando además la responsabilidad social o ética derivada de la introducción social de productos informáticos [Wedekind].
- Orientación crítica-social, que considera como principales tareas de la informática el análisis y la reorganización de procesos de aprendizaje y de trabajo y su apoyo técnico [2], asi como en la creación medial de accesos a la información [3].

Sin embargo, estas tres orientaciones se encuentran combinadas, si bien la orientación hacia la ingeniería es predominante y tiene un mayor eco en el ámbito profesional.

Por otra parte, esta perspectiva es también la que prevalece en la principal asociación de informática de Alemania GI (Gesellschaft für Informatik – Sociedad para la Informática). Esta asociación esta representada principalmente por personas pertenecientes al ámbito académico. Sin embargo, es una de las instituciones más activas en el ámbito de la informática en cuanto a la transformación curricular de cara a la situación en el plano industrial. De este modo, la GI ha formulado una serie de directrices para la mejora de la disciplina de informática a nivel académico, entre las que se pueden distinguir:

- La introducción de grados de Bachelor (6 Semestres) y Master (3 o 4 semestres) de corta duración.
- La acreditación de estudios de informática de otras instituciones[2], que sean aceptados por la industria y evaluados adecuadamente.

Aspectos de carácter socioeconómico y jurídico , asi como competencias sociales y comunicativas, son consideradas primordiales por esta asociación para las escuelas universitarias, mientras que para las universidades la GI tan solo recomienda considerar estos aspectos [19].

3. Espacios monoeducativos para la enseñanza universitaria de la informática

En el año 1999 tiene lugar por vez primera en Bremen la primera "Informática Feminale" con el fin de acercar particularmente a las alumnas de informática y de otras ramas académicas afines, a temas centrales de la disciplina y del mundo laboral, teniendo en cuenta su situación minoritaria en las facultades alemanas de informática. Dada dicha situación, esta oferta monoeducativa surge como medida de motivación y para ofrecer seguridad a las mujeres en la

[1] Littig, Peter: Berufe in der Informationstechnik. In: Blätter zur Berufskunde, Bundesanstalt für Arbeit, Nürnberg, 1999, S. 165f

[2] La oferta de formación en informática extrauniversitaria se ha disparado ha partir de 1999 coincidiendo con la escasez de especialistas y la enorme demanda de fuerza de trabajo en el sector del software.

informática, asi como para paliar los estereotipos de género que prevalecen en la cultura de esta disciplina [14,13]. Principalmente se persiguen tres objetivos:

- Motivación para la discusión entre mujeres de temas curriculares
- Creación de campos de experimentación para nuevos conceptos de ensenanza
- Desarrollo de nuevos conceptos para la formación continua de científicas.[3]

En ese mismo año se realizó el primer "call for papers" para el que hubo una enorme acogida. Los tres bloques temáticos tratados fueron:

- Redes informáticas
- Desarrollo de Software como proceso
- Interacción y medios

Asimismo, se desarrollaron diversos cursillos sobre programación o administración de sistemas y conferencias gratuitas acerca de la situación de las mujeres en la industria del software o sobre las perspectivas profesionales de la informática. Otra característica de este evento es la participación internacional de las docentes, provinientes de Dinamarca, Holanda, Austria, Gran Bretaña y Estados Unidos.

En los años posteriores, Informatica Feminale se ha consolidado como "Estudios de Verano" de la Universidad de Bremen y se han organizado durante el verano del 2001 y del 2002 posteriores ediciones paralelas en Baden-Württemberg (Fachhochschule Furtwangen y Institut für Informatik und Gesellschaft – Universidad Freiburg). El éxito de estos estudios de verano ha quedado reflejado en el enorme número de participantes que se incrementa cada año, asi como en la evaluación del evento.

4. Aspectos sociales y de género en la enseñanza universitaria de la informática en el Institut für Informatik und Gesellschaft

El Departamento "Modelación y consecuencias sociales" del Instituto de Informática y Sociedad de la Universidad de Freiburg tiene como fin la integración de aspectos sociales en la disciplina de informática y con ellos especialmente la consideración del género en el curriculum de informática.

Teniendo en cuenta las barreras de carácter estructural y simbólico a las que se enfrentan las mujeres en su carrera académica y profesional en el ámbito de la informática en Alemania [12,14], el departamento "Modelación y consecuencias sociales" pretende sensibilizar a los/as estudiantes acerca de la construcción y reproducción del género a distintos niveles, asi como impulsar la investigación del género en y sobre la informática. Los ámbitos principales de actuación son:

- La participación de las mujeres en el ámbito profesional y académico de la informática y la construcción del género en organizaciones de este ámbito.
- El análisis de productos de software desde una perspectiva de género (principalmente relacionados con E-Learning en la Informática).
- La creación de plataformas virtuales para la enseñanza de la informática.

Especialmente en relación con este último aspecto, el departamento colabora en tres iniciativas: Universidad virtual Oberrhein VIROR, Derecho e Informática en Internet RION, enseñanza y aprendizaje móvil de la Informática en Freiburg F-MOLL. En cada uno de estos proyectos han sido integrados aspectos relativos al género a través de los contenidos curriculares.

Por un lado, se tienen en cuenta factores estructurales relativos a la participación de las mujeres en el mercado de trabajo relacionado con las tecnologías de la información y la comunicación. En Alemania esta participación es minoritaria, sobre todo en aquellos campos cuya denominación, contenido y atribución social se hallan más cercanos a "la tecnología". Factores relacionados con esta minoritaria representación de las mujeres en el ámbito profesional de la informática, como son su participación en los estudios de informática y la caracterización cultural androcéntrica de la disciplina son también analizados [11] y transferidos en seminarios al alumnado. Una de las características principales de la transferencia de conocimeintos relacionados con el género en el departamento de Modelación y consecuencias sociales es la interdisciplinariedad. Dado que la situación minoritaria de las mujeres o

[3] http://www.informatik.uni-bremen.de/~oechteri/Texte/Top99.html

en otras palabras, la situación mayoritaria de los hombres en la informática en Alemania responde a factores culturales, sociales y psicológicos, estas disciplinas se hallan representadas junto con la informática, las ciencias naturales y la ingeniería asi como la matemática en este departamento.
Algunos de los temas de género que se imparten en seminarios son:

- Introducción a los estudios de género en la técnica y las ciencias naturales
- Inter-Action: Aspects of communication at work
- Género y desarrollo sostenido en la sociedad de la información
- Usability y Género relacionados con Groupwaretools
- Aspectos de Género del aprendizaje virtual

Los seminarios tienen un marcado carácter interdisciplinar en relación con los contenidos, tanto como con los docentes y los/as estudiantes, que no solo pertenecen a la facultad de informática, sino también a la de matemáticas, o a la de sociología principalmente. Asimismo, algunos de los docentes pertenecen a la industria o a institutos especializados con el fin de garantizar una transferencia más efectiva y continuada entre la teoría y la praxis.

5. Conclusión

En este artículo hemos realizado un breve resumen de la situación de la enseñanza universitaria de la informática en Alemania. La evolución de la disciplina se halla marcada por la situación de la industria de las tecnologías de la información y la comunicación y sus cortos ciclos de innovación. Dado el enorme dinamismo de este sector industrial y su carácter integrativo, las universidades y escuelas universitarias se encuentran en una situación de competencia frente a otras instituciones educativas que ofrecen títulos de informática. Como hemos señalado, la acreditación es una de las medidas que se recomiendan como seguro de calidad. Asimismo, aspectos sociales, económicos y relacionados con competencias comunicativas son recomendados para ser introducidos en el curriculum. Entre ellos se halla también la atención a la diversidad y al género. Por otra parte hemos mostrado algunos ejemplos de cómo se introducen aspectos de género en el curriculum universitario de informática en Alemania. Este interés se encuentra también influido por la situación minoritaria que sufren las mujeres en la informática en Alemania, la cual tanto la Informática Feminale como el departamento de Modelación y consecuencias sociales del Instituto de Informática y Sociedad de la Universidad de Freiburg contribuyen a paliar.

Referencias

[1] Broy, M./Schmidt, J. W.: Informatik: Grundlagenwissenschaft oder Ingenieursdisziplin? In: Informatik-Spektrum, 1999, 22, S. 206-209.

[2] Coy, W.: Brauchen wir eine Theorie der Informatik? In: *Informatik- Spektrum* 1989, 12, 256-266

[3] Coy, W. : Defining Discipline. In Freksa, Ch. /Jantzen, M./ Valk, R. (Hrsg.): In: *Foundations of Computer Science*, 1997, Berlin, Springer

[4] Dijkstra, E. W. (1989): On the Cruelty of Really Teaching Computer Science.. In: *Comm. ACM 32*, 1398-1404

[5] Dostal, W. Turbulenzen im IT-Arbeitsmarkt. In: Informatik Spektrum, 2001, 8, 35-52.

[7] Dostal, W.:IT Arbeitsmarkt. Chancen am Ende des Booms. *IAB Kurzbericht*, Ausgabe Nr. 19 / 21. 8. 2002.

[8] Endres, A.: Akademische und praktische Informatik – zwei Inseln ohne Brücken? in: *Informatik Spektrum* 2001, 12, 105-158.

[9] Lückefett, Hans-Jochen u. Dr. Thomas, Uwe (Projektleitung): Die Entwicklung des Arbeitsmarktes und der Hochschulplätze für IT-Fachkräfte in Deutschland. Zwischenergebnis der Arbeitsgruppe ”Bildung und Qualifikation”, Themenschwerpunkt Berufliche Bildung und Arbeitsmarkt. Initiative D21 u. Input Consulting, Frankfurt/Stuttgart Januar 2001

[10] Pasch, J. (1994) Kooperative Softwareentwicklung im Team. Berlin, Sprinter.

[11]Ruiz Ben, E., Schinzel, B. (2001) Die Professionalisierung der Informatik in Deutschland" in: *Forum Bildung und Beschäftigung NFP 43 des SNFs*

[12] Ruiz Ben, E.: Qualifikation, Erfahrung und Geschlecht. In: *FiFFKo,* 2002, 9, 37-41.

[13] Ruiz Ben, E.: Looking beyond the software boom. Gendered costs and benefits? In: Pasero, U. & Gottburgsen, A. (Hrsg.): *Gender from costs to benefits.* Opladen, Westdeutscher Verl. (to appear).

[14] Schinzel, B.; Kleinn, K.; Wegerle, A.; Zimmer, C.: Das Studium der Informatik: Studiensituation von Studentinnen und Studenten. Ziel ist die Stärkung des Selbstbewußtseins von Frauen in der Informatik. In: *Informatik-Spektrum*, Bd. 22, Heft 1, Februar 1999, S. 13-23.

[15] Wajcman, J.: *Feminism Confronts Technology* 1991 Pennsylvania: Penn State University Press.

[16] Wedekind, H. (1987) Gibt es eine Ethik der Informatik? In: *Informatik-Spektrum*, 10, 324-328.

[17] Weltz, F. & Ortmann, R.G.: *Das Softwareprojekt: Projektmanagement in der Praxis.* 1992, Frankfurt a. M., Campus

[18] Woodfield, R.: *Women, Work and Computing*. Cambridge; Cambridge Univ. Press.

[19]http://link.springer.de/link/service/journals/00287/papers/0023006/00230383.pdf

Calidad y evaluación de la docencia

Estudio del rendimiento académico de una asignatura con relación a asignaturas de cursos anteriores

Luisa Zúnica, Pedro Blesa, Rosa Alcover, Jorge Más, José M. Valiente
Escuela Técnica Superior de Informática Aplicada.
Universidad Politécnica de Valencia
46022 Valencia
e-mail: lrzunica@eio.upv.es , pblesa@dsic.upv.es , ralcover@eio.upv.es , jmas@fis.upv.es, jvalient@disca.upv.es

Resumen

Se intenta estudiar la relación entre el rendimiento de una asignatura tipo de segundo curso con respecto al rendimiento de las asignaturas del curso anterior, con el fin de detectar si existe una relación más fuerte con la/s asignatura/s posibles prerrequisitos. No se detecta que sea significativo este efecto, pero en el análisis estadístico realizado se obtienen dos factores o componentes relevantes en el rendimiento de dicha asignatura de segundo curso. Estos factores se pueden interpretar como el rendimiento global del alumno, y su "destreza" en asignaturas propiamente informáticas.

1. Motivación y objetivos

Desde hace algún tiempo han proliferado en la Universidad española diversos estudios sobre rendimiento académico, consecuencia de la creciente percepción que tenemos los docentes de que "algo" no funciona en la Universidad. Las quejas iniciales surgidas en aquellas Facultades y Escuelas con bajas notas de acceso han dado paso a quejas en Centros tradicionalmente considerados de "élite", donde ya se percibe, también, sensación de fracaso académico. El recurso fácil de achacar todas las culpas al alumnado se entremezcla con otras voces más autocríticas que ven deficiencias tanto en los Centros como en el profesorado y en los planes de estudio vigentes. Consecuencia, en buena medida, de estas preocupaciones, ha sido la contrareforma de los planes de estudio, que están implantándose actualmente. Ya con la idea de acometer esta mencionada reforma con cierto conocimiento de causa, el Consejo de Universidades recomendó la realización de este tipo de estudios para evaluar los planes de estudio vigentes [1].

En particular, en la Universidad Politécnica de Valencia se puso en marcha, hace ya una docena de años, el Proyecto de Innovación Educativa (P.I.E), con la intención de transformar las técnicas docentes. Acogidos a este Proyecto, un grupo de profesores de la UPV relacionados con la Escuela Técnica Superior de Informática Aplicada o ETSIA (antigua Escuela Universitaria de Informática) iniciamos un Proyecto [2] con el fin de elaborar una herramienta informática que permitiera extraer, de manera cómoda, sencilla, y sin necesidad de conocimientos informáticos avanzados, datos de rendimiento académico de la abundante base de datos de la que dispone la Universidad [3].

Mediante esta aplicación se han elaborado algunos estudios relativos a diferentes aspectos que pensamos tienen influencia sobre el rendimiento académico, factores tales como la nota de acceso de los alumnos a la Universidad, procedencia de los alumnos (COU, FP.....), organización de los estudios, y otros [4,5]. En la mayoría de ellos hemos estudiado, de manera especial, la problemática existente en primer curso, que nos parece especialmente grave. Pero también tenemos la sensación de que en segundo y tercer curso, donde el fracaso académico también es elevado, la problemática no es la misma que en primero, y pueden afectar factores distintos y de distinta manera.

En este trabajo pretendemos abordar esta última cuestión: analizar, de manera rigurosa, factores que inciden en el rendimiento académico en primer curso y en segundo curso tomando como ejemplo la titulación de Ingeniero Técnico

en Informática de Sistemas en la E.T.S. de Informática Aplicada de la UPV.

2. Estudio realizado

El primer problema que se presenta al intentar hacer estudios de rendimiento académico es el de definir de forma clara qué es el rendimiento académico. En este sentido, y a partir de otros estudios sobre la cuestión [6], definimos el rendimiento académico de un alumno en una asignatura en un curso como un parámetro comprendido entre 0 y 100 en el que se tiene en cuenta la calificación obtenida, la convocatoria en la que se aprobó la asignatura, y los años transcurridos desde la primera matriculación del alumno en la asignatura, según la fórmula:

$$R_{ij} = \frac{0.4^{n_{ij}-1} k_{ij} C_j}{C_j} * 40$$

donde R_{ij} es el rendimiento del alumno i en la asignatura j, n_{ij} es el año en el que el alumno i aprobó la asignatura j (referido al año en el que el alumno se matriculó en la asignatura j por primera vez), C_j es el número total de créditos de la asignatura j y k_{ij} es un coeficiente que depende tanto de la calificación obtenida por el alumno i en la asignatura j, como de la convocatoria en la que se aprobó, pudiendo tomar los valores siguientes:

Calificación	Convocat. Ordinaria	Convocat. Extraordinaria
<5	0	0
≥5 y <7	1,5	1
≥7 y <9	2	1,5
≥9	2,5	2

A partir de esta definición, se pueden definir rendimientos referidos a distintos ámbitos; para este estudio hemos utilizado el rendimiento de una asignatura j durante un período de tiempo P, entendido como la suma de los rendimientos, en la asignatura j, de todos los alumnos que durante el período P estuvieron matriculados en dicha asignatura:

$$R_j(P) = \frac{\sum_{i \in j(P)} 0.4^{n_{ij}-1} k_{ij}}{\sum_{i \in j(P)} i} * 40$$

El segundo problema que se nos presenta para llevar a cabo el estudio pretendido, es el de definir de forma clara la población a estudiar. En el caso de estudios sobre primer curso es generalizada la técnica de elegir una cohorte de alumnos que ingresan en una titulación un curso determinado. Pero el hacer un estudio sobre segundo curso plantea el problema de definir quiénes son alumnos de segundo curso, máxime en un Centro donde no existen incompatibilidades de matrícula entre asignaturas, y es común que muchos alumnos cursen, de forma simultánea, asignaturas de varios cursos.

Por ello, el procedimiento seguido fue el siguiente:

- Elegimos una asignatura de segundo curso (primer cuatrimestre) que, a nuestro juicio, fuera representativa en cuanto a resultados académicos; dicha asignatura fue Tecnología de Computadores (TCO)
- Seleccionamos todos aquellos alumnos que durante el curso 2000-2001 estuvieron matriculados en dicha asignatura, calculando el rendimiento obtenido por cada alumno en dicho curso.
- Buscamos, en los cinco cursos anteriores al 2000-2001, para cada uno de los alumnos señalados, el rendimiento obtenido en cada una de las asignaturas de primer curso (0 en caso de que la asignatura no hubiera sido aprobada, o su valor en caso contrario).
- Hicimos un estudio estadístico de toda la información obtenida.

Durante el curso 2000-2001 hubo matriculados en la asignatura TCO 295 alumnos. Al consultar las calificaciones obtenidas por dichos alumnos en las asignaturas de primer curso a 110 de estos alumnos les faltaban algunos datos, por lo que no pudieron ser incluidos en el análisis. Así pues, los alumnos estudiados fueron un total de 185.

3. Análisis Estadístico

En principio, un primer análisis que podría responder a nuestro objetivo consistiría en plantear directamente un modelo de regresión lineal múltiple que ligara el rendimiento obtenido por los alumnos en la asignatura Tecnología de Computadores (RTCO) con los correspondientes a la totalidad de las asignaturas de primer curso (variables explicativas). Sin embargo, este análisis presenta el problema de que las variables explicativas están muy relacionadas entre sí (por ejemplo, los alumnos buenos en general obtienen buenos rendimientos en la mayor parte de las asignaturas). En estas situaciones puede resultar estadísticamente imposible diferenciar a partir de los datos los efectos de las diferentes variables, pudiendo incluso resultar que ninguna de ellas aparezca como estadísticamente significativa, pese a que su efecto conjunto sí que lo es claramente.

En consecuencia, se ha optado por plantear, en primer lugar, un análisis de componentes principales (ACP) sobre las doce variables correspondientes a los rendimientos de las asignaturas de primer curso [7,8]. El objetivo de este primer análisis es el de condensar o resumir la información contenida en estas 12 variables en un conjunto reducido de nuevas variables, no observables directamente e incorrelacionadas entre sí. A estas nuevas variables se les denomina componentes. Una vez obtenidas estas componentes, se ha planteado un modelo de regresión tomando RTCO como variable dependiente y las nuevas variables (componentes), obtenidas en el análisis previo, como variables explicativas o independientes.

Para el análisis de datos se ha utilizado el programa *Statgraphics*. La siguiente tabla, en donde cada componente se identifica en la primera columna por su "Factor Number", muestra los resultados del ACP, apreciándose (en la cuarta columna) que las cinco primeras componentes explican el 70.464% de la variabilidad total generada por los rendimientos en los alumnos de las doce asignaturas de primer curso. Se puede observar como las tres primeras componentes son las más relevantes, y las únicas para las que los valores propios asociados son mayores que 1.

Factor Analysis

Factor Number	Eigenvalue	Percent of Variance	Cumulative Percentage
1	3,79314	31,610	31,610
2	1,87026	15,586	47,195
3	1,03054	8,588	55,783
4	0,930574	7,755	63,538
5	0,831145	6,926	70,464
6	0,747084	6,226	76,690
7	0,656801	5,473	82,163
8	0,575508	4,796	86,959
9	0,502572	4,188	91,147
10	0,431711	3,598	94,744
11	0,371239	3,094	97,838
12	0,259428	2,162	100,000

En la tabla siguiente se recogen las correlaciones de las doce variables estudiadas (que corresponden a los rendimientos de las doce asignaturas, y se indican en la primera columna, así, RALG es Álgebra) con los cinco factores, o componentes obtenidos:

	Factor 1	Factor 2	Factor 3	Factor 4	Factor 5
RAD1	0,6257	-0,60099	0,0005748	0,025003	-0,130839
RAD2	0,461046	-0,42729	-0,022731	0,561262	-0,025720
RALG	0,608918	0,299588	-0,229924	-0,021365	0,453123
RAM1	0,672737	0,433403	0,0829508	-0,150341	-0,146342
RAM2	0,518438	0,615221	0,0705987	-0,0059330	-0,181611
REC1	0,654986	-0,15231	-0,106565	-0,172309	-0,239751
RES1	0,64444	0,3198	-0,101065	0,0458893	-0,395488
RFCO	0,532464	-0,3688	0,0060651	-0,415436	0,413077
RFFI	0,544929	0,080442	0,0367246	0,581528	0,24955
RINT	0,110721	0,040769	0,966623	0,0078179	0,0621313
RIPR	0,502559	-0,60229	0,083306	-0,186428	-0,200165
RMAD	0,638234	0,195028	0,0321449	-0,120271	0,25084

A partir de la tabla anterior se puede observar que para la primera componente todos los coeficientes

de las diferentes asignaturas son positivos y del mismo orden de magnitud (en torno a 0.5-0.6), salvo el coeficiente de la asignatura Inglés Técnico (RINT) que es del orden de 0.11 . Esta primera componente o factor estaría indicando que los alumnos que son buenos estudiantes obtienen en general buenos rendimientos en todas las asignaturas. Por sus características especiales la asignatura Inglés Técnico (INT) se diferencia de las demás puesto que los alumnos pueden llegar a la ETSIA con diferentes niveles de inglés, y esto no está necesariamente relacionado con lo buen o mal estudiante que pueda ser un alumno.

Respecto a la segunda componente o factor, se observan coeficientes tanto positivos como negativos. Si nos fijamos en aquellos cuyo valor absoluto es mayor (RIPR, RAD1, RAD2, RFCO, RAM1, RAM2, RES1) pueden diferenciarse, para un mismo nivel promedio global recogido por el primer factor, dos grupos de estudiantes: aquellos que presentan en primer curso cierta destreza en las asignaturas relacionadas con la programación y materias propias de la Informática (Introducción a la Programación, Algoritmos y Estructuras de Datos 1 y 2, Fundamentos de Computadores) y aquellos alumnos que la presentan en las asignaturas más relacionadas con las matemáticas como pueden ser Análisis Matemático 1 y 2 y Estadística 1. Opinamos que el primer grupo de alumnos puede proceder de FP2 mientras que el segundo puede proceder de COU, aunque habría que hacer otros estudios para comprobar esta afirmación.

La tercera componente pone de manifiesto la diferencia entre el rendimiento en la asignatura Inglés Técnico (con un coeficiente de 0.97) y el del resto de las asignaturas de primer curso, con coeficientes marcadamente inferiores.

A efectos de claridad y con el fin de mostrar gráficamente las afirmaciones anteriores, en la figura 1 se han representado las correlaciones de las doce variables estudiadas con las dos primeras componentes o factores.

Nótese como, respecto a la primera componente, todos los puntos representados se sitúan a la derecha (correlaciones positivas). La segunda componente está diferenciando rendimientos en asignaturas propiamente informáticas (correlaciones negativas) de aquellos obtenidos en asignaturas más matemáticas (correlaciones positivas), quedando el rendimiento de la asignatura Inglés Técnico en una situación intermedia.

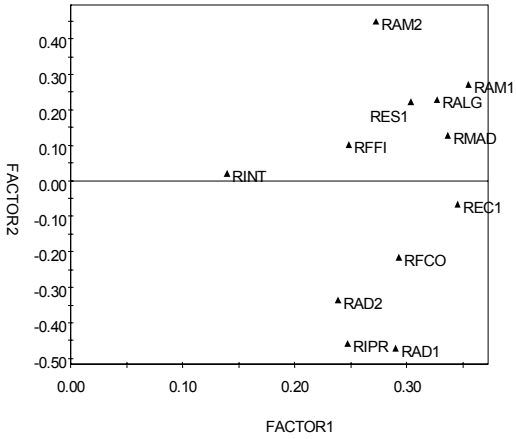


Figura 1. Gráfico de las correlaciones de las doce variables con las dos primeras componentes

Una vez obtenidas las componentes principales el siguiente paso ha consistido en plantear un modelo de regresión que permita analizar el efecto de éstas sobre el promedio del rendimiento en la asignatura de segundo curso TCO. El modelo es el siguiente: $E(RTCO)= \beta_0+ \beta_1*FACTOR1+\beta_2*FACTOR2+\beta_3*FACTOR3+ \beta_4*FACTOR4+ \beta_5*FACTOR5$, donde los parámetros β_i recogen y cuantifican el efecto de las componentes ($FACTOR_i$) sobre el rendimiento medio en TCO. La siguiente tabla recoge los resultados de este análisis de regresión:

```
Multiple Regression Analysis
----------------------------------------
Dependent variable: RTCO
----------------------------------------
                    Standard    T
Parameter Estimate Error Statist.P-Value
----------------------------------------
CONSTANT    13,46   1,36   9,92   0,0000
FACTOR1     1,467   0,36   4,09   0,0001
FACTOR2    -2,319   0,73   -3,19  0,0017
FACTOR3     0,113   1,32   0,08   0,9317
FACTOR4     0,923   1,46   0,63   0,5285
FACTOR5     1,932   1,64   1,18   0,2394
----------------------------------------
R-squared = 13,6246
```

En el análisis se observa que únicamente las dos primeras componentes resultan estadísticamente significativas (valores en la columna *P-value* inferiores a 0.05). Además la estimación de los coeficientes de las dos primeras componentes (columna *Estimate*) presenta signos contrarios. Respecto al efecto de la primera componente, el signo positivo obtenido implica la conclusión, bastante esperable por cierto, de que los alumnos que al ser buenos estudiantes obtienen buenos rendimientos en primer curso también obtienen buenos rendimientos en la asignatura TCO de 2º curso.

En cuanto al efecto de la segunda componente, el signo negativo obtenido para β_2 indica que en general los alumnos con valores negativos en la misma obtienen mejores calificaciones en TCO. Estos alumnos serían aquéllos en los que predomina la destreza en las materias más propias de la Informática frente a la correspondiente a materias de carácter más matemático.

Sin embargo, el valor obtenido para el parámetro *R-squared* indica que el 13.6% de la variabilidad del rendimiento de los alumnos en la asignatura TCO viene explicado por las variables consideradas en el modelo planteado, lo que resulta un porcentaje bastante reducido. El modelo es, por tanto susceptible de mejora puesto que otros factores no contemplados en él ayudarían a explicar mejor dicha variabilidad.

4. Conclusiones

En su origen, este trabajo estaba enfocado principalmente a estudiar si existía una relación importante entre una determinada asignatura, y cualquier otra que fuese prerrequisito, y diferente en comparación con la de otras asignaturas previas. Para ello se escogió una asignatura típica de segundo curso de nuestra escuela (TCO: Tecnología de Computadoras) y tomamos como población todos los matriculados en dicha asignatura en un curso, obteniendo el rendimiento académico de cada individuo en las asignaturas previas hasta los cinco cursos anteriores.

En esta situación, un modelo de regresión múltiple no es directamente aplicable a las variables manejadas (rendimiento de las diferentes asignaturas), dada la elevada relación entre las variables explicativas. Por ello se ha planteado, un análisis de regresión tomando como variables explicativas las componente principales obtenidas en un análisis previo (análisis de componentes principales).

Las dos primeras componentes resultan estadísticamente significativas, y tienen, en nuestra opinión, una interpretación clara. La primera componente estaría indicando que los alumnos que, en general, son buenos estudiantes obtienen también buenos rendimientos en la asignatura TCO. En cuanto a la segunda componente, se ha visto que separa claramente dos grupos de estudiantes: los alumnos que presentan ciertas habilidades en asignaturas relacionadas directamente con la informática, tales como programación y estructura de computadoras, y los que las muestran en las asignaturas relacionadas con las matemáticas. El primer grupo de alumnos podría proceder de formación profesional, y el segundo de COU . Esta segunda componente influye en el rendimiento de TCO en el sentido de que los alumnos que muestran una mayor destreza en las asignaturas informáticas obtienen también mejores rendimientos en TCO.

Dado el escaso porcentaje de variabilidad del rendimiento de TCO explicado por el modelo pensamos que este es susceptible de mejora puesto que otros posibles factores no contemplados en él ayudarían a explicar mejor dicha variabilidad. Es nuestra intención seguir con el análisis, y ofrecer en un futuro nuevas conclusiones.

Agradecimientos

A Hanane Tachafait y Jesús Lafuente por su esfuerzo en el desarrollo de la aplicación informática, y al Instituto de Ciencias de la Educación (ICE) de la Universidad Politécnica de Valencia por su financiación.

Referencias

[1] B.O.E. de 17 de Enero de 1997
[2] P.I.D. 9052 de la UPV. Libro de resúmenes de los P.I.D. de la UPV. 2001.
[3] Más, Alcover, y otros. *Una herramienta informática para un estudio multidimensional del rendimiento académico en la EUI de la*

UPV. Libro de resúmenes del VII CUIE. Huelva. 1999.

[4] Benlloch, J.V; Bonet, E. y otros. *Estudio comparado del rendimiento de los alumnos de primer curso procedentes de COU frente a los alumnos procedentes de FP*. Libro de resúmenes de las IV Jornadas de enseñanza universitaria de Informática. Andorra. 1998.

[5] Más, J. Meseguer, J.Mª *Estudio sobre la heterogeneidad de conocimientos básicos en alumnos de primer curso de universidades politécnicas*.. Libro de resúmenes de las VI Jornadas de enseñanza universitaria de Informática. Alcalá de Henares. 2000.

[6] González, R. M. *Rendimiento académico en la UPM: estudio longitudinal en primer ciclo, Vol. 1 y 2*. I.C.E. de la UPM, 1993.

[7] Mardia, K., Kent, J. Y Bibby, J. (1979). *Multivariate Analysis*. Ed. Academic Press, London

[8] Sharma, S. (1996). *Applied Multivariate Techniques*. Ed Wiley

E-valuate: un modelo de autoevaluación para la mejora de la enseñanza y el aprendizaje

Mª Ángeles Díaz, Ana Belén Martínez, Miguel Riesco, Carlos Iglesias
Depto. de Informática
Universidad de Oviedo
e-mail: fondon@correo.uniovi.es
{belen, albizu}@lsi.uniovi.es
i1632776@petra.euitio.uniovi.es

Resumen

En el modelo educativo actual de este país el proceso de evaluación se usa prácticamente de forma exclusiva como mecanismo de rendimiento de cuentas del alumno de cara al profesor y se concede poca importancia a la evaluación como mecanismo de valoración para la mejora del proceso de aprendizaje por parte del alumno.

Por otra parte, no existen herramientas al servicio del profesorado que permitan a éste realizar una autoevaluación de su propia docencia, para así tener conocimiento de la eficacia de su manera de proceder en la impartición de la disciplina, y poder corregir aquellos aspectos que han resultado peor parados.

En este artículo se describen las características de una herramienta, e-valúate!, que hace posible una evaluación individualizada, continua y adaptativa en la que cada alumno consigue información elaborada de sus errores con respecto a objetivos, contenido de la materia y causas del error.

Asimismo, e-valúate! registra y compara resultados a nivel de grupo de alumnos, lo que permite extraer conclusiones acerca del grado de cumplimiento de objetivos alcanzados en la materia. Por lo tanto, se proporcionará también al profesor información elaborada relativa a la calidad de la transferencia de información, sacando a la luz los puntos débiles del proceso de enseñanza.

1. Importancia de la autoevaluación de la docencia

En la literatura se encuentran múltiples referencias a la influencia de la evaluación en el aprendizaje, es decir, la evaluación para la mejora y no sólo para la calificación.

La autoevaluación es, por tanto, un proceso compuesto de realización de una prueba y obtención de información acerca de los errores cometidos, el por qué de los mismos y los efectos de esos errores en cuanto a su grado de importancia y respecto a la consecución de objetivos en el proceso de aprendizaje.

Es posible clasificar el proceso de autoevaluación en niveles diferentes:

- *Autoevaluación del discente.* El alumno es el primer interesado en poder valorar su grado de conocimiento, comprensión y dominio de la materia para poder atacar los aspectos que aún no están maduros. El proceso de evaluación para la mejora es un elemento básico dentro del proceso de aprendizaje, por lo que una herramienta que le permita de forma sencilla realizar pruebas de autoevaluación podría resultar muy interesante.
- *Autoevaluación del docente.* El profesor, en cuanto que profesional de la enseñanza, debe llevar a cabo una valoración de su de trabajo para la mejora continua de la calidad docente. Si bien son diversos los mecanismos que puede llevar a efecto [2] es evidente que uno de ellos consiste en hacer una valoración de los resultados del aprendizaje obtenidos por los alumnos. La información con la que cuenta habitualmente a este respecto es simplemente el conjunto de calificaciones de las pruebas. El análisis de información más específica, como puede ser calcular el grado de cumplimiento de los objetivos específicos por parte del grupo puede ayudar a sacar conclusiones acerca de los puntos fuertes y débiles de la explica-

ción y realizar un proceso de realimentación positiva de la docencia.

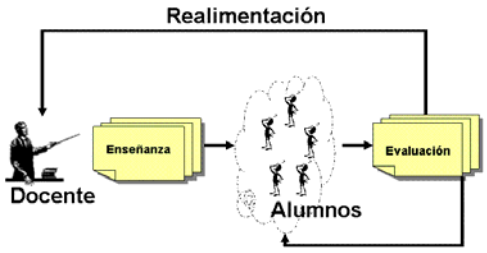


Figura 1. Valoración de los resultados del aprendizaje de los alumnos

Este segundo nivel, si bien no es tan estudiado como el primero, resulta fundamental en el nuevo entorno en el que se desarrolla la educación superior, en el que la calidad de la docencia constituye el primer elemento diferenciador [4].

2. Antecedentes

Con el uso generalizado de Internet por parte de la población en general, comienzan a aparecer las plataformas de tele-enseñanza sobre la red (llamadas también enseñanza virtual on-line), que proporcionan un paso de gigante a un tipo de enseñanza mucho más antigua y con gran demanda como es la educación a distancia. [1] [3] [5] y[10] son ejemplos de este tipo de plataformas.

La aceptación ha sido tan grande que hoy en día prácticamente todas las universidades ofrecen asignaturas o incluso titulaciones completas basadas en entornos de formación que han dado en llamarse "Sistemas de Tutor Inteligente".

Tras la realización de un proceso de análisis acerca de cuáles son las características destacables de los sistemas de este tipo disponibles en la actualidad, se puede calificar la situación de heterogénea e incompleta. Conclusión que por otro lado, parece bastante lógica dado que la introducción de estas herramientas ha sido veloz en el tiempo, y no ha dado lugar a excesivas reflexiones ni análisis de resultados.

Asimismo, el profesorado, empieza a darse cuenta de la necesidad de aplicación de las tecnologías en el proceso de enseñanza, como elementos complementarios al esfuerzo personal y al componente humano, por otra parte irreemplazable. Así surgen las primeras herramientas específicas de apoyo a la docencia, ligadas en los mejores casos a proyectos de innovación docente, alentados éstos en los últimos años por las universidades, sabedoras de que la mejora de la calidad docente será clave en los procesos de acreditación universitaria, no sólo a nivel nacional, sino también europeo y mundial.

Si nos centramos en el proceso de automatización de la evaluación, cabe decir que la mayoría de los tutores inteligentes ofrecen algún tipo de herramienta de evaluación. Así por ejemplo, Learning Space [9] es una herramienta que permite una evaluación y auto-evaluación del alumno. El proceso de evaluación puede ser individualizado, permitiendo que no todos los usuarios reciban las mismas preguntas, y su resultado será comunicado al profesor. Por el contrario, los resultados de las auto-evaluaciones, que podrán ser realizadas un número indefinido de veces, no serán proporcionadas al profesor impidiendo así su realimentación.

Algo similar ocurre con WebCT [11]. Esta herramienta proporciona un aprendizaje significativo, y permite tanto pruebas de evaluación, que proporcionan información a los alumnos a causa de sus fallos (feedback), como controles de evaluación. Estos últimos suponen el método de autoevaluación proporcionado por la herramienta que puede ser añadido a cualquier módulo de contenidos. No obstante, los resultados de estos controles no son proporcionados tampoco al profesor, únicamente se le comunica la calificación al alumno.

Además de plataformas completas como las anteriores, existen herramientas encaminadas únicamente a la evaluación. Tal es el caso, por ejemplo, de SIETTE [8] que supone una implementación de la teoría de test adaptativos, generando dinámicamente las preguntas en función del nivel de conocimientos que ofrezca el alumno evaluado. Sin embargo, no proporciona ninguna herramienta que pueda ayudar al profesor, ni al alumno, a comprobar la evolución seguida en el proceso de aprendizaje.

Como conclusión principal que se puede extraer tras el análisis de la situación actual respecto a lo que ofrece el "mercado web", y centrándonos en el aspecto de las herramientas de autoevaluación, podemos decir que si bien son relativamente importantes en cantidad, no resultan completas en prestaciones, sino que desarrollan ciertos aspectos

del conjunto de los abordables en el tema de la autoevaluación.

Sería por tanto interesante la elaboración de un modelo de autoevaluación donde se especifiquen los requisitos deseables para un sistema de autoevaluación completo.

3. El modelo de autoevaluación

En la elaboración del análisis de requisitos para el modelo de sistema de autoevaluación, se han tenido en cuenta los "indicadores para evaluar entornos integrados para la formación en Internet" [7].

Así pues, se consideran deseables las siguientes características para el modelo:

3.1. Características generales

- *Facilidad de uso*. Es quizás el requisito por excelencia que debe tener cualquier servicio Web.
- *Adaptabilidad*. Se debe diseñar una arquitectura escalable de forma que permita una mejor adaptación a las necesidades del momento y su posible transformación a necesidades futuras.
- *Compatibilidad.* La herramienta debe tener carácter universal y no propietario de la Plataforma, de forma que permita utilizarse sin necesidad de adaptaciones con los equipos y sus configuraciones que son estándar de facto entre los usuarios de Internet. La compatibilidad debe tenerse en cuenta en requisitos software y hardware, soporte de formato de ficheros multimedia, estándares para el almacenamiento de información y protocolos de comunicación o accesibilidad a discapacitados.
- *Robustez.* Referida a la fiabilidad de uso de la plataforma tanto desde el punto de vista de su consistencia como de su protección contra usos indebidos, sean o no malintencionados.

3.2. Edición de elementos de evaluación

- *Riqueza de plantillas*. Para generar e interconectar elementos de evaluación y actividades de refuerzo. Diversos tipos de preguntas: test simple, test múltiple, ejercicios de respuesta corta, ejercicios con solución numérica en base a la generación aleatoria de cuestiones calculadas (admitiendo una tolerancia en la respuesta), ejercicios cuya solución es proporcionada de forma gráfica (por ejemplo cronogramas), tablas, etc.
- *Riqueza de recursos.* Variedad de recursos que se puede incluir en cada pregunta. En concreto, la posibilidad de incluir gráficas, imágenes, ecuaciones, ayudas o realimentaciones, pudiendo éstas últimas ser más o menos inteligentes, de forma que según la respuesta nos lleven a la parte de los contenidos donde se aborda la cuestión.
- *Gestión de las Bases de datos.* Se debe incluir la capacidad de crear cuestionarios a partir de una base de datos de preguntas, la incorporación de cuestiones implementadas en otros lenguajes, y el manejo simultáneo de diversas bases de datos.
- *Tipos de cuestionarios.* La plataforma debe ser capaz de llevar a cabo distintos modelos de evaluación, con objetivos claramente diferenciados. Concretamente, pruebas de autoevaluación, evaluación parcial, evaluación global. Asimismo, ha de proporcionar una capacidad de desarrollo off-line.

3.3. Proceso de enseñanza – aprendizaje

La herramienta ha de ser capaz de permitir el diseño de pruebas que se adapten a los diferentes alumnos. Es importante incorporar funciones disponibles para el profesor que le permitan efectuar una monitorización del trabajo de los alumnos de forma individual y en su conjunto. Análogamente, deberá incorporar funciones dirigidas al alumno para la realización de sus actividades de aprendizaje.

Concretamente, podemos destacar las siguientes:

- *Evaluación adaptativa.* Generación de pruebas adaptadas a las distintas características e intereses de los estudiantes.
- *Flexibilidad.* Respecto al diseño de pruebas de evaluación con capacidad para controlar aspectos tales como: puntuación, número de intentos, duración de la prueba, etc.

- *Supervisión y control de las pruebas***.** Seguimiento automatizado y monitorización del trabajo y progreso de los alumnos.
- *Generación de un expediente del alumno.* Integración de las herramientas de evaluación (cuestionarios, trabajos, calificación del profesor, etc.) en un único expediente para cada alumno. Dicho expediente (o un resumen) puede remitirse al alumno al finalizar el curso.
- *Control de su progreso,* con información de su aprendizaje y comparativa con otros alumnos.

3.4. Otras características

Dado que el sistema puede ser utilizado por múltiples profesores y múltiples alumnos, deben incluirse las características típicas de gestión de usuarios, como altas, bajas, gestión de perfiles, gestión de contraseñas, etc.
Otra parte importante del sistema constituye la *generación de informes*. En base a los resultados de la evaluación de los alumnos es interesante realizar distintas estadísticas, tanto sobre los alumnos como sobre las propias preguntas. Datos como el grado de consecución de los objetivos o el índice de fallos de preguntas por temas pueden contribuir decisivamente a mejorar la docencia de la asignatura.

4. Principales características de e-valúate!

A continuación se describen los principales aspectos desarrollados en la herramienta, siguiendo los requisitos del modelo de evaluación descrito.

4.1. Tipo de pruebas de evaluación

El tipo de prueba elegido para la evaluación es el test de conocimientos adquiridos, aunque el sistema estará abierto a la incorporación posterior de otros tipos de pruebas.

Los tests constarán de un número variable de preguntas con un número también variable de opciones de respuesta, de las cuales solo una podrá ser verdadera, siendo las demás falsas. Habrá un mínimo de tres y un máximo de cinco opciones de respuesta posibles por pregunta.
Existen tres niveles de elaboración de pruebas

- *Nivel básico.* Representa la plataforma de edición para la generación de exámenes de tipo test, incorporando formatos específicos para la pregunta y las respuestas posibles, generación automática de modelos, ocultación de respuestas correctas, movimiento automático de número de pregunta y apartados, generación de cabeceras, etc.
- *Nivel medio.* Supone la incorporación de información adicional al nivel básico. Así por ejemplo, cada pregunta puede llevar asociadas características como sobre qué contenido trata la pregunta.
- *Nivel avanzado.* Otro de los atributos asociado a las preguntas consiste en definir la lista de objetivos que cubre su resolución así como qué grado de dificultad conllevan (según el tipo de objetivo).

Cada respuesta, podrá tener asociada una pequeña explicación o aclaración sobre el por qué de su veracidad o falsedad que podrá ser mostrada como ayuda al alumno después de la corrección de las pruebas. Esta explicación debe ser introducida en el sistema por el profesor. Las preguntas también podrán tener asociadas una aclaración similar.

4.2. Generación de pruebas

Las pruebas podrán ser generadas manualmente por un profesor para valorar el rendimiento, o bien automáticamente por el sistema en función de la elección de algunos parámetros de dificultad y objetivos a cubrir. Esta información servirá al alumno para realizar una autoevaluación, pudiendo éste elegir entre los diferentes niveles (básico, medio o avanzado) de complejidad, lo que permitirá establecer diferentes grados de dominio de la materia y evitar la desmotivación del alumno.

Los distintos niveles podrían ser generados a partir de los tipos de objetivos a cubrir, según sean de mayor o menor complejidad.

4.3. Proceso de autoevaluación del aprendizaje

Una vez generado un examen el alumno podrá resolverlo con la propia herramienta, obteniendo inmediatamente los resultados de la evaluación.

El resultado obtenido será en forma de calificación al modo tradicional (nota numérica), pero

además el alumno podrá conocer otros parámetros fundamentales para la mejora de su aprendizaje, tales como, los apartados o subapartados de la materia que no domina, o lo que es más importante, los objetivos que aún no ha logrado en el proceso de aprendizaje.

Como ilustración, el sistema debe proporcionar mensajes informativos del tipo: "Este alumno puede tener problemas de compresión en el objetivo 'x'. Su rendimiento en este objetivo es bastante menor que en los demás objetivos".

- *Evaluación continua y adaptativa.* La herramienta facilita datos estadísticos relevantes acerca de los progresos realizados, sobre diferentes pruebas, dando consejos personalizados para mejorar los resultados o elaborando automáticamente ejercicios suplementarios personalizados para aquellas tareas en las que cada individuo presenta mayores dificultades.
- *Evaluación comparativa.* Además, los alumnos podrán hacer un seguimiento de su evaluación en comparación con el resto, mediante distintas estadísticas accesibles a través de su perfil personal en el sistema

4.4. Proceso de autoevaluación de la enseñanza

- *Análisis de resultados de una prueba***.** El mecanismo que proporciona la herramienta de cara a la valoración y mejora de la docencia impartida consiste en el análisis de los resultados de las evaluaciones de los alumnos. Extrae información acerca de aquellos objetivos o apartados que tienen un índice bajo de comprensión. Como ilustración, el sistema proporcionará mensajes informativos como el siguiente: "El rendimiento global del grupo para el objetivo 'x' es muy bajo. Puede haber problemas de comprensión. El objetivo debería ser reforzado".
- *Análisis de resultados de múltiples pruebas.* Asimismo, deberán poder realizarse análisis evolutivos del conocimiento adquirido a lo largo del curso, e incluso sobre la evolución del aprendizaje en años sucesivos a medida que se modifican aspectos docentes

4.5. Supervisión y control de pruebas

Para lograr una evaluación lo suficientemente fiable de las pruebas realizadas la herramienta utilizará un modelo matemático para la evaluación, caracterización y diagnóstico de problemas educativos, basado en el modelo de comunicación de Shannon, ampliado posteriormente por Weaver, y adaptado para los sistemas telemáticos de educación por profesores de la Universidad Carlos III de Madrid [6]. Gracias a este modelo obtendremos una medida del rendimiento de los alumnos en función del grado de cumplimiento de los distintos objetivos requeridos.

Respecto al docente, el sistema permitirá evaluar exámenes escritos, a través de los datos recibidos en un formato obtenido tras la lectura de los mismos mediante una lectora óptica. Del mismo modo, la valoración irá desde la calificación numérica clásica hasta una valoración en función de objetivos.

4.6. Gestión de usuarios

La herramienta contempla la existencia de distintos tipos de usuarios. Para que sea posible un seguimiento del alumno este deberá registrarse como usuario del sistema. Cualquier usuario no registrado podrá utilizar el sistema pero sufriendo una serie de restricciones:

- No podrá beneficiarse de la evaluación continua, ya que al no poseer un perfil personal no es posible el seguimiento de su evolución.
- No podrá recibir recomendaciones del sistema por no estar siendo evaluado de forma continua.
- No podrá realizar aquellas pruebas propuestas por los profesores que requieran una contraseña.
- No podrá acceder a las estadísticas.

4.7. Adaptabilidad, compatibilidad y robustez

Se pretende que e-valúate! forme parte de un entorno integrado de formación discente y también docente, por lo que dispone de una arquitectura escalable que permita una mejor adaptación a las necesidades del momento y su posible transformación a necesidades futuras.

Para mantener compatibilidades, se ha tenido en cuenta el cumplimiento de un conjunto de protocolos y normas para el entendimiento común entre plataformas de teleformación y contenidos didácticos como se manifiesta en su capacidad para importar y exportar información en formato XML.

La herramienta proporciona control de acceso de usuarios mediante protocolos seguros con cifrado de canales de comunicación, como SSL.

5. Conclusión

Uno de los mecanismos de autoevaluación para la mejora de la docencia consiste en la realización de un análisis estadístico de los resultados del aprendizaje de los alumnos.

Hasta ahora la funcionalidad que se ha ido incorporando dentro de los Entornos Integrados de Formación, está dirigida a la ayuda al aprendizaje del alumno, bien como complemento a la docencia presencial o bien incluso como plataforma para la educación a distancia.

El uso de un entorno de formación vía web constituye un marco ideal para la extracción de información y tratamiento estadístico del proceso de aprendizaje del alumno. Esto propiciará la aparición de herramientas que utilicen los resultados de la interacción con el alumno para la mejora del proceso de la enseñanza.

Un ámbito inmediato sería la creación de una herramienta dirigida al análisis de los resultados de las pruebas objetivas (exámenes), si bien existen otros ámbitos aplicables, como el uso de tutores inteligentes de compilación que almacenen información de los errores cometidos por los alumnos y obtengan estadísticas de la frecuencia de los mismos.

El uso de estas nuevas herramientas ligadas al Web abre un nuevo campo en el ámbito educativo, y concretamente en el proceso de autoevaluación de la docencia.

E-valúate constituye un sistema de autoevaluación que surge tras un proceso de estudio y reflexión haciendo un paréntesis en la frenética carrera de generación de aplicaciones web en la que nos vemos inmersos.

Referencias

[1] Blackboard. http://www.blackboard.com/.

[2] Brown George, Atkins Madeline, Atkins Madeleine. *Effective Teaching in Higher Education*. Methuen Drama, 1998.

[3] Carro M, Pulido E., Rodríguez P. *TANGOW: a Model for Internet Based Learning. International Journal of Continuing Engineering* Education and Life-Long Learning, IJCEELL, Pub. UNESCO. Special Issue on "Internet based learning and the future of education". Vol. 11, Nos. 1/2, 2001.

[4] Cátedra UNESCO de gestión de educación superior de la UPC. *Calidad de la docencia y formación del profesorado*. Boletín de Educación Superior Nº 1, 2001.

[5] Docent. http://www.docent.com/.

[6] Fernández M.C., Fernández J., Marín A. y Pardo A. *Extensión al modelo de Shannon para la caracterización y diagnóstico de problemas educativos*. Actas del Tercer Congreso de Interacción Persona Ordenador. Madrid, 2002.

[7] Portaencasa Raquel, Arriaga García, J. y otros *Sindicadores para evaluar entornos integrados para la formación* http://hermes.gate.upm.es/platafomas/herramientastele/index.htm.

[8] Ríos, A. Pérez de la Cruz, J.L., Conejo, R. *SIETTE: Intelligent Evaluation System using Test for TeleEducation*. 4th International Conference on Intelligent Tutoring System.ITS'98. Workshops papers, San Antonio, Texas, 1998.

[9] Sponberg Hilding. *The Visual, Virtual Learning Space*. The Lisbon 2000 European Conference, 2000.

[10] WBT Systems (TopClass) http://www.wbtsystems.com/.

[11] WebCT. http://www.webct.com/.

Directrices éticas y legislación informática

Gestión académica y protección de datos

Xavier Canaleta, David Vernet
Dpto. de Informática
Enginyeria i Arquitectura La Salle
Universidad Ramon Llull
08022 Barcelona
e-mail: {xavic, dave}@salleurl.edu

Resumen

En esta ponencia pretendemos analizar si los métodos y acciones que son práctica habitual en la evaluación y la gestión académica del alumnado están afectados por la Ley Orgánica de Protección de Datos de Carácter Personal [1] (en adelante LOPD) y la Ley de Servicios de la Sociedad de la Información y Comercio Electrónico [2] (en adelante LSSICE).

Con ello pretendemos realizar un análisis de la gestión que nos permita identificar las acciones que conllevan un claro incumplimiento de los reglamentos vigentes, incidiendo especialmente en aquellas que pueden ser tipificadas como faltas graves o muy graves. Pero, además, queremos sugerir posibles actuaciones a llevar a cabo para ajustarse a la normativa y evitar así posibles sanciones de los organismos pertinentes.

Pudiera parecer que ciertos aspectos aquí tratados son más propios del área de Derecho que del ámbito de las Tecnologías de la Información. Pero el camino recorrido desde la entrada en vigor de la LOPD hasta la actualidad, nos hace ver que es imprescindible la colaboración de profesionales de ambos sectores para una correcta interpretación y aplicación de esta legislación que tiene un marcado carácter interdisciplinar.

1. Antecedentes

Antes de proceder al desarrollo de los temas centrales de la ponencia, creemos necesario hacer una breve descripción cronológica de la normativa relacionada con la protección de datos.

El primer referente legal en el estado español sobre protección de datos lo hallamos en la Ley Orgánica 5/1992 [3], de 29 de octubre, de Regulación del Tratamiento Automatizado de los Datos de carácter personal, más conocida como LORTAD. La legislación española iba con cierto retraso respecto al resto de los países de Europa. Recordemos que el Convenio 108 del Consejo de Europa (donde se recogían una serie de principios para la protección de las personas en la utilización de sus datos personales incluidos en los diversos ficheros automatizados) es del año 1981; y países como Alemania, Austria, Francia e Inglaterra desarrollaron leyes sobre protección de datos e informática a finales de los años 80.

En 1994 aparece, en el Real Decreto 1332/1994 de 20 de junio [4], un reglamento donde se desarrollan ciertos aspectos de la LORTAD, exceptuando uno muy importante: la seguridad y las medidas a adoptar.

El 24 de octubre de 1995 se publica la Directiva Europea 95/46/CEE [5], donde, en su artículo 17, se recogen una serie de obligaciones respecto al tratamiento de datos de carácter personal realizado por terceros. Estas van desde el establecimiento en un contrato que detalle esta responsabilidad hasta la implantación de las medidas técnicas que garanticen la seguridad de dichos datos.

Y no es hasta 1999, concretamente el 25 de junio, cuando sale publicada en el BOE el Real Decreto 994/1999 de 11 de Junio [6], por el cual se aprueba el Reglamento de medidas de seguridad de ficheros automatizados que contengan datos de carácter personal. Este reglamento es una pieza clave ya que hace que la LORTAD sea operativa.

Y el 14 de diciembre del mismo año, se publica la Ley Orgánica de Protección de Datos de Carácter Personal (LOPD) que deroga la LORTAD. Esta ley es una actualización de la

NIVEL	TIPO DE DATOS	MEDIDAS A ADOPTAR
Básico	• Nombre y apellidos • Direcciones (física y electrónica) • Teléfono (fijo y móvil)	• Documento de Seguridad • Registro de incidencias • Identificación y autenticación de usuarios • Control de acceso • Gestión de soportes • Copias de respaldo y recuperación
Medio	• Información de Hacienda Pública • Infracciones administrativas • Infracciones penales • Información financiera	• Medidas de seguridad de nivel básico • Creación del Responsable de Seguridad • Auditoría bianual • Medidas adicionales de autenticación • Medidas adicionales de identificación
Alto	• Ideología y creencias • Religión • Origen racial • Salud	• Medidas de nivel básico y medio • Seguridad en la distribución de soportes • Registro de accesos • Cifrado en las telecomunicaciones

Tabla 1. Niveles de protección de datos y medidas de seguridad

antigua LORTAD. Quizás la novedad más importante en relación a la LORTAD es la extensión en su ámbito de aplicación a los ficheros no automatizados. Otro aspecto a tener en cuenta es la diferenciación en tres niveles de protección de datos, dependiendo de la naturaleza de los mismos, y la descripción de las medidas de seguridad a adoptar en cada caso (véase Tabla 1). Y finalmente recordar que la LOPD no deroga el Reglamento de la LORTAD sino al contrario: en su disposición transitoria tercera explicita la subsistencia de las normas preexistentes a la LOPD que no la contradigan.

La consecuencia de lo expuesto en el párrafo anterior es la vigencia de los plazos de implantación de las medidas de seguridad que el Real Decreto 994/1999 establecía: 26 de diciembre de 1999 para las medidas de nivel básico, 26 de junio de 2000 para las de nivel medio y 26 de junio de 2001 para las de nivel alto.

En relación a estos plazos cabe mencionar que el Real Decreto 195/2000 [7], de 11 de febrero, amplía el plazo para implantar las medidas de seguridad para el nivel básico. La nueva fecha pasa a ser el 26 de marzo de 2000. Y la Resolución de 22 de junio de 2001 [8] acaba fijando la fecha definitiva para la implantación de medidas de nivel alto el 26 de junio de 2002.

Y la última Ley española relacionada con los sistemas de información y la protección de datos sale publicada en el BOE número 166, de 12 de julio de 2002. La Ley 34/2002 es la que se conoce como la LSSICE: Ley de servicios de la sociedad de la información y comercio electrónico. Esta Ley pretende regular jurídicamente ciertos aspectos que se derivan del uso de las nuevas tecnologías de la información, en especial Internet, y muy directamente lo que se conoce como el comercio electrónico. La LSSICE, además de cubrir un vacío jurídico evidente en lo que se refiere a legislación informática en Internet, pretende dar protección de los intereses de los consumidores en el comercio electrónico.

2. Análisis del entorno académico

En primer lugar vamos a definir nuestro entorno de trabajo para poder determinar así en qué situación nos encontramos y qué normas nos afectan.

Queremos desvincular, en una aproximación inicial, el entorno académico del entorno administrativo. Por entorno académico nos referiremos a los datos relacionados con los alumnos y su currículum universitario. De este modo, podríamos enumerar ciertos contenidos que pueden ser considerados información académica. Esta lista no pretende especificar todos los datos

exhaustivamente sino dar una idea descriptiva del tipo de contenidos de esta información:

- Datos personales del alumno: como pueden ser el nombre y apellidos, la fecha de nacimiento, el domicilio, los teléfonos de contacto, la dirección electrónica, etc.
- Datos académicos: incluiríamos aquí toda la información referente al historial académico del alumno, donde constan, entre otros datos, las asignaturas que ha cursado y sus correspondientes calificaciones.
- Datos logísticos: en este apartado hallaríamos información que nos indique a qué grupo ha asistido dentro de una determinada asignatura, cuál era su horario de clases, qué profesores impartían los créditos a los que estaba matriculado, etc.

No consideramos que pertenezca al ámbito académico aquella información de naturaleza económica relacionada con los alumnos, como pueden ser los importes de matriculación y créditos cursados, los datos de la domiciliación bancaria para satisfacer dichos importes, el importe de las becas que le hayan sido concedidas, etc. Este tipo de datos del alumno los ubicamos dentro del entorno administrativo y no serán objeto de nuestro estudio.

Una vez especificado nuestro ámbito de trabajo podemos proceder al análisis del mismo y determinar qué leyes descritas anteriormente le afectan y en qué medida.

Los datos personales que se pueden encontrar en ficheros, informatizados o en soporte papel, referentes al entorno académico podemos concluir que son de nivel bajo. No se especifica en ninguno de los documentos en vigor nada referente a los datos académicos de una persona física por lo que debemos considerarlos estrictamente datos de carácter personal, tipificados estos de nivel básico. De este modo hemos de tener en cuenta que la información académica se halla afectada por la LOPD y también por el Reglamento 994/1999 en vigor sobre las medidas de seguridad.

Otro punto a analizar es la Ley 34/2002, más conocida como la LSSICE y su posible aplicación a la gestión académica. Parece lógico pensar que actualmente la gran mayoría de la información académica se almacena en soporte informático. Por lo que podemos intuir que su tratamiento irá más allá del mero almacenamiento en ficheros automatizados y, con toda probabilidad, se aplicarán las nuevas tecnologías de la información y comunicaciones (Internet) para dicha gestión académica. Pero aunque así sea y se usen los servicios de la sociedad de la información para agilizar y facilitar las actualizaciones y consultas de datos académicos, también es cierto que estos no se utilizan para generar comunicaciones comerciales ni para la contratación electrónica. Los servicios académicos que se suelen ofrecer a través de campus virtuales a los alumnos no pueden considerarse en modo alguno dentro del campo de aplicación de esta Ley y todos los datos de carácter personal que se gestionen deben regirse por las normas de la LOPD, que tienen como objetivo proteger el derecho a la intimidad y la privacidad de datos.

En resumen, la información académica y las operaciones que realicemos con ella deberán adecuarse a las normas establecidas por la LOPD y el Reglamento LORTAD vigente. Además, tendrán que adoptarse las medidas de seguridad especificadas para datos de nivel básico.

De todos modos, nos gustaría dejar constancia que una ley permite diferentes lecturas y, consecuentemente, distintas interpretaciones. Con ello pretendemos decir que si bien parece evidente que los datos académicos son, en sí mismos, de nivel básico, no es menos cierto que a través de un expediente académico se puede inducir un perfil de personalidad. ¿No es lógico pensar que si analizamos el historial académico de un alumno con calificaciones brillantes en su expediente y, de repente, encontramos un descenso drástico y global en sus notas, deduciremos que se encuentra en un momento de crisis personal y nos será más sencillo intuir que ese alumno tiene alguna problemática (ya sea laboral, sentimental o familiar) en su entorno? ¿No es verdad que los tutores utilizan los datos académicos de los alumnos para poder analizar su evolución, sus preferencias e inclinaciones profesionales? ¿Y de este modo poder evaluar múltiples aptitudes como pueden ser la tenacidad, capacidad de trabajo, constancia, extroversión, etc., que van más allá del mero perfil académico? Y, llegados a este punto, ¿podríamos concluir que estos datos podrían ser considerados de nivel alto?

3. Medidas de seguridad

Dado que los ficheros con datos académicos se ven afectados por la LOPD, estos deben ser declarados a la Agencia de Protección de Datos. Esta notificación puede hacerse por correo ordinario. También puede realizarse casi en su totalidad vía Internet a través de la página web https://www.agenciaprotecciondatos.org.

Seguidamente se procederá a la adopción de las medidas de seguridad de nivel básico para los ficheros afectados. Vamos a detallar en qué consiste cada una de ella y las acciones que deben llevarse a cabo en el caso que nos ocupa.

3.1. Documento de seguridad

Se ha de redactar un documento donde se detallen las normas a seguir por todo el personal que tenga acceso a los datos académicos con el fin de garantizar la protección de los mismos. Este documento también debe de contener la descripción de todos los procedimientos exigidos para las medidas de seguridad para el nivel básico. Algunos ejemplos serían: un procedimiento que especifique cómo, cuando y quién realiza las copias de seguridad; otro procedimiento que determine cómo se realiza la recuperación de datos si hay una caída del sistema, etc. Este documento debe estar actualizado, por lo que se ha de realizar una revisión del mismo con cierta periodicidad. La universidad también debe garantizar que el documento de seguridad sea del conocimiento de todo el personal involucrado en el tratamiento de datos académicos.

3.2. Registro de incidencias

Debe existir un formulario que ha de cumplimentarse en el caso que se produzca alguna incidencia que afecte a los ficheros con información académica. Así que cada vez que haya una interrupción en el suministro eléctrico que provoque una parada de los servidores de datos, o se produzca una avería en los soportes magnéticos o exista algún problema con las comunicaciones que no permita acceder a la información sensible (por poner tres ejemplos bastante habituales), dicha incidencia deberá quedar registrada en un formulario.

3.3. Identificación y autenticación de usuarios

El centro universitario debe asegurar que el acceso a los datos académicos sólo lo realizarán aquellas personas que estén habilitadas para ello. De este modo el acceso a los sistemas de información que permitan la consulta o actualización de información académica debe estar controlado. El mecanismo más común se basa en la existencia de nombres de usuario y contraseñas. Se acostumbra a adoptar un conjunto de normas que permitan garantizar la inequívoca identificación y autentificación de los usuarios en el sistema de información. Quizá alguna de las normas que citaremos conlleve un grado de protección más exigente que las estrictamente requeridas en el nivel básico, pero de todos modos aconsejamos que se implanten:

- no están autorizados usuarios genéricos que permitan accesos al sistema. Así pues, el usuario "profesor" no sería válido ya que permitiría el acceso al sistema a diversas personas y no se podría conocer qué persona en concreto está accediendo a la información protegida.
- activación de la caducidad de *passwords*: habitualmente con una periodicidad mensual o bimensual se fuerza que el usuario cambie su contraseña. De este modo limitamos el uso de contraseñas que hayan podido ser sustraídas de sus dueños.
- guardar un histórico de contraseñas, con el fin de evitar repeticiones en los cambios y obligar a los usuarios a cambiar realmente la palabra de paso.
- bloquear la cuenta de usuario a partir de un cierto número de intentos fallidos, con el objeto de limitar posibles accesos no autorizados de forma reiterada.
- No permitir *passwords* triviales: no se deben usar como contraseñas datos obvios del usuario como pueden ser sus iniciales, fecha de nacimiento, nombre de sus familiares cercanos, etc. Estas son las palabras habituales que se usan para intentar acceder al sistema.

3.4. Control de acceso

Esta medida determina que el personal sólo tendrá acceso a los datos académicos que le sean necesarios para desempeñar su trabajo. Del mismo modo únicamente se autorizarán las operaciones estrictamente requeridas sobre estos datos. Así pues, un profesor sólo debe tener acceso a los datos académicos de sus alumnos y, siendo extremadamente estrictos, a los referentes a su asignatura. Un profesor de la asignatura de Programación no debería tener acceso a las calificaciones de sus alumnos en la asignatura de Electrónica. En cambio un tutor debe tener acceso a todos los datos académicos de sus alumnos para poder desarrollar correctamente sus funciones. Y no sólo esto sino que el tutor de un alumno, evidentemente, podrá tener acceso a los datos de ese alumno pero en modo consulta y no actualización ya que no es el responsable de evaluar a ese alumno en las diferentes materias de las que este se haya matriculado.

Todas las acciones anteriores tienen como objetivo limitar el acceso lógico a los datos. En este sentido creemos necesario hacer notar que a veces controlamos correctamente los accesos lógicos a los datos y olvidamos el posible acceso físico. Por ello queremos insistir en el estricto control de acceso físico a la sala de servidores y también en un hábito que parece una nimiedad pero que acostumbra ya a aparecer como norma para los usuarios del sistema de información en las empresas: la obligatoriedad de bloqueo de la estación de trabajo en ausencia del trabajador y la activación del salvapantallas en un tiempo no superior a 5 minutos y su posterior desbloqueo con contraseña. De este modo evitaremos posibles descuidos de bloqueo del terminal y podremos garantizar la no vulnerabilidad física del sistema.

El control de acceso descrito hasta el momento es un control lógico.

3.5. Gestión de soportes

Los soportes que contengan información académica deberán estar correctamente identificados, tienen que estar inventariados y almacenados en un lugar de acceso restringido al personal autorizado. La salida de soportes informáticos con datos académicos fuera de los locales de ubicación de los ficheros debe quedar registrada y tiene que estar autorizada por el responsable del fichero. Si se pretende hacer una copia de las calificaciones de alumnos en un disquete o CD-ROM para poder trabajar con ella desde otro lugar, esta acción ha de estar autorizada explícitamente por el responsable de los datos y, además, ha de quedar constancia escrita de esta salida de información. Al ser información sensible, estos datos no han de ser accesibles por cualquier persona fuera del centro.

3.6. Copias de respaldo y recuperación

Es necesario realizar copias de seguridad de los datos académicos periódicamente. Esta operación es habitual hoy en día en la mayoría de centros. La periodicidad de las copias (diaria, semanal, etc.) variará según la volatilidad de la información. Muchos centros acostumbran a realizar copias de *backup* todos los días. Sin embargo no es tan frecuente verificar que estas copias permitan la recuperación de los datos en caso de pérdida de los mismos en el sistema. Esta medida de seguridad nos obliga a realizar pruebas de recuperación periódicamente. Las empresas que cumplen dicha medida hacen estas verificaciones trimestral o semestralmente.

4. Prácticas punibles

Una vez descritas las acciones necesarias para adoptar las medidas de seguridad exigidas por la LOPD, podríamos pensar que ya estamos cumpliendo correctamente las disposiciones de la Ley y también del Reglamento. Lo cierto es que hay prácticas habituales que se realizan en la gestión académica en entornos universitarios que son claras infracciones a la LOPD e, incluso, vulneran el artículo 18.4 de la Constitución Española, que no es otro que el derecho a la intimidad. Citamos textualmente: “La ley limitará el uso de la informática para garantizar el honor y la intimidad personal y familiar de los ciudadanos y el pleno ejercicio de sus derechos”.

No pretendemos aquí realizar una persecución inquisidora de estas prácticas. Nuestro objetivo se limita a detectar y exponer ciertos hábitos que pueden ser constitutivos de infracción y proponer acciones alternativas que nos permitan actuar

dentro de las normas establecidas y evitar así posibles sanciones.

Empezaremos con una de las prácticas más habituales en el mundo académico como es la publicación de resultados de unos exámenes ya sean parciales o finales. El modo más usual de informar a los alumnos de sus calificaciones es que cada asignatura publique listas con sus notas. Si nos fijamos esta acción vulnera el principio de privacidad ya que se están haciendo públicos datos de carácter personal de los alumnos. Los artículos 6 y 11 de la LOPD informan que para comunicar datos personales a un tercero se necesita el consentimiento previo del afectado. De este modo todos los alumnos deberían hacer constar, o no, si permiten la notificación pública de calificaciones. Esto crearía un primer problema ya que ciertos alumnos aceptarían dicha publicación pero otros no. Un segundo problema viene dado por el derecho que tiene el alumno, según los mismos artículos, a revocar dicha autorización en cualquier momento. Creemos que este camino no lleva a ninguna solución viable. Se ha optado por soluciones intermedias como puede ser la publicación de listas de notas donde sólo existe el número de expediente de cada alumno y su nota. Esta solución tampoco cumple el objetivo de privacidad si existen sistemas accesibles al público que permitan a partir del nombre de un alumno obtener su expediente. La propuesta que parece más lógica es la de comunicar los resultados de las diferentes asignaturas personalmente a cada interesado. En este aspecto las tecnologías de la información nos facilitan dicha tarea ya que es relativamente sencillo diseñar un entorno donde permita el acceso personalizado al expediente de cada alumno según su nombre de usuario y contraseña. Nuestra experiencia de utilizar el campus virtual de nuestra escuela de ingeniería (*e-campus*), donde los alumnos pueden consultar sus notas de forma individualizada, ha tenido gran éxito, no tan solo por la privacidad de datos sino por la facilidad de acceso vía Internet desde cualquier punto geográfico. En resumen, hablando claramente podemos concluir que las listas de notas publicadas ya sea en los tablones de anuncios son una práctica ilegal según la LOPD. Y soluciones intermedias como la publicación de un documento pdf con la misma lista de notas en una intranet del campus universitario virtual al cual tienen acceso sólo los alumnos matriculados de esa asignatura continua infringiendo la Ley de Protección de Datos.

Otra práctica habitual, realizada sin ningún ánimo perverso, es la divulgación a través de la Red de imágenes personales. Sea a través de Internet o en una *intranet* de un centro universitario, la divulgación de fotografías personales está prohibida sin el consentimiento explícito de la persona.

Actualmente la mayoría de universidades, y en especial las que tienen estudios científico-técnicos, poseen desde una simple página web hasta un completo campus virtual donde el alumno encuentra multitud de servicios de gran utilidad para su vida académica. El problema está en que, pretendiendo ser una ayuda para el alumno, se ofrezcan servicios que difundan datos de carácter personal. Una utilidad habitual en los campus virtuales es el buscador. Esta herramienta permite a cualquier usuario de Internet realizar consultas de otros alumnos a partir de su nombre y apellidos o su número de expediente. La búsqueda nos permite obtener información de aquel alumno; nos facilita su dirección electrónica y su URL si esta existe. Este servicio nuevamente vulnera la Ley ya que permite la obtención a terceros de datos de carácter personal sin el consentimiento expreso del alumno.

La solución tecnológica para estos casos de consentimiento del alumno y derecho a una posible revocación de este sería poder implantar un pequeño aplicativo en la web, con acceso por *login* i *password*, que le permita al alumno la configuración de su perfil. De este modo podría actualizar sus datos personales, podría activar o no el consentimiento para la difusión de su fotografía, *e-mail* y URL a través de la web o a través de la *intraweb*.

5. Conclusiones

Se han presentado en esta ponencia las medidas de seguridad que deben adoptarse para los ficheros de datos académicos, dada su condición de datos de carácter personal de nivel básico. También se han descrito acciones habituales en la gestión de la información académica que constituyen una infracción a la LOPD y se han propuesto soluciones efectivas encaminadas a solucionar

este conflicto sin que por ello se deba renunciar a los objetivos que tenían dichas acciones.

Con todo, creemos que sería fácil de constatar que en muchos entornos universitarios las medidas de seguridad que hemos descrito no se han aplicado o si se han aplicado no están siendo seguidas. Adicionalmente, se siguen utilizando frecuentemente métodos en la gestión académica que son claras infracciones a la LOPD.

Llegados a este punto, creemos que es necesario hacerse la siguiente reflexión: ¿por qué no se han implantado las medidas de seguridad que marca la ley? ¿Por qué sigue siendo tan habitual la gestión académica mediante acciones que suponen infracciones tipificadas como graves por la LOPD? Sería ingenuo pensar que es debido a los altos costes que suponen la implantación de las medidas de seguridad exigidas y las soluciones alternativas a los métodos usados. Pensamos que ha quedado constancia que ni dichas medidas suponen un coste elevado para las universidades ni la adopción de nuevos métodos de trabajo tiene una complicación tecnológica excepcional.

La realidad es que convergen un conjunto de factores que hacen que nos encontremos en esta situación. Un primer factor determinante es la falta de información y formación: creemos que hay un gran desconocimiento en el sector informático, como en muchos otros, de las obligaciones y normas que afectan a los ficheros con datos de carácter personal. Para ello no hallamos otra solución que informar a los formadores y, posteriormente, informar y formar a los futuros profesionales del sector, nuestros alumnos. Existen diferentes sistemas para conseguir este objetivo. Una posibilidad que debería plantearse seriamente sería la incorporación de créditos obligatorios dentro de los planes de estudios que desarrollen esta temática que, por supuesto, afecta a todos los profesionales de la informática, sea cual sea su especialidad. De todos modos, nosotros abogamos más por una solución quizá más compleja de ejecutar pero más efectiva: creemos que la información sobre la legislación que afecta al sector debe darse con más continuidad e insistencia. La solución de añadir una asignatura, aunque no nos parece mal, no creemos que sea del todo efectiva. Evidentemente cumpliríamos el objetivo de informar pero no el de concienciar. Esta tarea es interdisciplinar y debe tener un pequeño hueco temporal reservado dentro de cada una de las asignaturas del plan de estudios. La manera de concienciar a futuros profesionales de la importancia de la protección de datos, que tiene su arraigo en el derecho constitucional a la intimidad, es informar desde distintas ópticas y eso sólo es posible hacerlo con la colaboración de todo el estamento académico. De hecho, existen iniciativas en ese aspecto (por ejemplo Créditos de Libre Configuración, cursos de postgrado, etc.) pero nos ha sido realmente difícil hallar propuestas que tengan un ámbito de aplicación transversal como proponemos en este artículo.

Un segundo factor que influye decisivamente es el aspecto cultural. Por naturaleza el ser humano es reticente al cambio. Así que el hecho de adoptar nuevos métodos de trabajo y adquirir nuevos hábitos siempre encuentra cierta oposición. Por experiencia propia sabemos que la implantación de nuevas metodologías de trabajo es costosa, aunque esta tenga las mismas funcionalidades que la anterior e incorpore mejoras considerables. Nadie duda hoy en día que tener las calificaciones de los alumnos en una hoja de cálculo o base de datos es infinitamente más cómodo y eficaz que mantenerlas en soporte papel: agilidad de cálculo de promedios, posibilidad de actualización de datos, rapidez en las consultas, etc. Así mismo, ¿alguien del sector tecnológico puede cuestionar que tener un entorno en Internet donde se puedan publicar, actualizar y consultar datos académicos tiene las mismas ventajas y adicionalmente otras como pueden ser la accesibilidad, movilidad y protección de datos?

Finalmente también querríamos hacer notar la falta de implicación de la dirección en este asunto. Argumentamos los mismos motivos que en los docentes: desconocimiento, falta de formación, etc. Esto nos llama la atención sobretodo si tenemos en cuenta las grandes sanciones económicas que suponen las infracciones de la LOPD. Mientras que, mayoritariamente, las directivas de las empresas ya han tomado las medidas adecuadas para cumplir la normas que se derivan de la LOPD, pensamos que los en centros universitarios no se ha llegado aún a ese punto. Hechos como que no existan cláusulas relativas al cumplimiento de la LOPD en los contratos de los profesores universitarios, que no se divulgue entre estos una Normativa de Seguridad Informática o que ni exista esta constatan esta situación.

Agradecimientos

Queríamos aprovechar la oportunidad para agradecer al jefe del Departamento de Informática de Ingeniería y Arquitectura La Salle (Universidad Ramon Llull), la motivación que nos ha dado para redactar la presente ponencia. Nos ha transmitido su denodado interés por estos temas, así como la preocupación por la falta de concienciación que existe por ellos.

Referencias

[1] "Ley Orgánica 15/1999, de 13 de diciembre", *BOE* nº 298, 14 de diciembre de 1999.
[2] "Ley 34/2002, de 11 de julio", *BOE* nº 166, 12 de julio de 2002.
[3] "Ley 5/1992, de 29 de octubre", *BOE*, de octubre de 1992.
[4] "Real Decreto 1332/1994, de 20 de junio", *BOE*.
[5] "Directiva 95/46/CE, de 24 de octubre", *DOCE*.
[6] "Real Decreto 994/1999, de 11 de junio", *BOE* nº 151, 25 de junio de 1999.
[7] "Real Decreto 195/2000, de 11 de febrero", *BOE* nº 49, 26 de febrero de 1999.
[8] "Resolución de 22 de junio de 2001", *BOE* nº 151, 15 de junio de 2001.

Evaluación del alumnado

Algoritmo para la evaluación de exámenes tipo test en sistemas e-learning avanzados

R. Barchino, J. M. Gutiérrez, J. Macías, S. Otón
Dpto. de Ciencias de la Computación
Universidad de Alcalá
28871 Madrid
e-mail: {roberto.barchino, josem.gutierrez, javier.macias, salvador.oton }@uah.es

Resumen

El objeto de este documento es presentar ideas sobre la corrección automática de tests generados por ordenador en el contexto de sistemas de e-learning. Más en detalle, se presentan trabajos realizados entorno a las normas del IMS (Instructional Management Systems) en su apartado QTI (Question & Test Interoperability). Estos trabajos, orientados a la implementación de las normas en un sistema real y su uso para obtener una medida de sus posibilidades, han desembocado en la necesidad de tener en cuenta escenarios más complejos que los planteados por IMS y la necesidad de desarrollar algoritmos adaptados a estos escenarios.

El panorama final estudiado, comprende la realización de pruebas en varias fases donde las preguntas tendrán asociados un nivel y sólo podrán ser seleccionados para una fase de su nivel o superior. Un estudiante accederá a una pregunta de mayor nivel sólo si se ha superado en cierto grado la fase anterior.

También se trata, brevemente, los ajustes realizados en las diversas fases de funcionamiento del sistema para dotar al escenario de un funcionamiento lo más similar posible al que tendrá si no fuese ejecutado por un sistema automático. En particular, se describe la forma de calificar con una cierta relevancia a las preguntas que forman la base de conocimiento y cómo calcular la relevancia de forma sencilla, correcta y representativa del nivel real que representa esa pregunta.

1. Motivación

En estos últimos años se han desarrollado en el ámbito universitario y empresarial distintos sistemas de e-learning, que se centran en crear entornos virtuales de aprendizaje mediante una serie de herramientas tecnológicas que originalmente no estaban pensadas para este fin pero se han integrado correctamente en estos sistemas. Un componente fundamental de estos sistemas es el encargado de la evaluación de los avances de los alumnos en la asimilación de contenidos docentes.

Existen distintos estándares internacionales que proponen algoritmos de corrección de exámenes tipo test [1], pero no existe en la actualidad ningún sistema que permita la corrección automática de otro tipo de preguntas más abiertas que las de tipo test, debido a los lógicos problemas de análisis del lenguaje natural.

En el Departamento de Ciencias de la Computación de la Universidad de Alcalá hemos trabajado en estos últimos años con distintos sistemas comerciales de e-learning como LUVIT [2], WebCT [3] y Learning Space [4], pero aun cuando son suficientemente flexibles para trabajar con ellos decidimos crear nuestro propio sistema para experimentar e investigar de una forma más profunda.

En el año 2001 desarrollamos un primer sistema de e-learning para la generación aleatoria y corrección de exámenes tipo test [5], al año siguiente se creó una nueva versión del sistema que cumplía el estándar IMS-QTI para el almacenamiento e intercambio de test y en la actualidad se ha desarrollado una tercera versión del sistema que incorpora los distintos algoritmos

de evaluación de exámenes del estándar QTI de IMS.

En esta última versión de nuestro sistema hemos desarrollado nuevos algoritmos de evaluación que extienden a los del estándar QTI y que siguiendo su especificación almacenaremos en XML integrados en la propia estructura QTI. El motivo de desarrollar estos nuevos algoritmos es la búsqueda de mayor precisión en la evaluación de los conocimientos de los alumnos.

Una vez centrados en la evaluación de contenidos debemos tener presente una serie de cuestiones que nos van a servir como punto de partida en el desarrollo de los nuevos algoritmos de evaluación del aprendizaje.

Estas cuestiones son, en primer lugar que los exámenes deberán abarcar los contenidos fundamentales de la asignatura, que las preguntas de todos los exámenes sean claras y no exista ambigüedad, también que se debe dar a conocer a los usuarios del sistema el criterio o algoritmo de evaluación, y por último la calidad del propio examen tipo test [6] [7]

2. Algoritmo Propuesto

Los algoritmos disponibles en IMS [1] se han desarrollado teniendo en cuenta una serie de casos de uso significativos con la finalidad de que sean suficientemente flexibles como para cubrir otros muchos casos.

Algunos de estos casos de uso se enumeran en la tabla 1. No se enumeran todos los casos de uso que propone el estándar, debido fundamentalmente a que algunos de ellos no son de aplicación a nuestro sistema, por ser de propósito general como Inglés, Biología o materias orientadas a la impartición del método científico.

Nuestro sistema en su versión actual permite la realización de test como los incluidos en los tres primeros casos de uso y prepara, aunque no implementa aún, el cuarto tipo, esto es, permite almacenar pesos para preguntas pero no utilizarlos en la corrección. Todos estos casos de uso están pensados para la realización de la prueba de evaluación una vez terminado el aprendizaje y que se presenten los resultados de la misma de forma inmediata.

Caso de Uso	Caracterísitcas
Test de elección múltiple	Múltiples opciones Una sola correcta Todas las preguntas valen lo mismo
Test verdadero/falso	Dos opciones (verdadero/falso) Todas las preguntas valen lo mismo
Test de respuesta múltiple	Múltiples opciones Más de una correcta Todas las preguntas valen lo mismo
Test fin de temario	Múltiples opciones Una sola correcta Cada pregunta tiene diferente valor

Tabla 1. Casos de uso del estándar QTI

Los algoritmos de evaluación contenidos en la especificación QTI [1] se enumeran en la tabla 2.

Algoritmo
Number Correct
Number Correct (Attempted)
Weighted Number Correct
Weighted Number Correct (Attempted)
Parameter Weighted Number Correct
Parameter Weighted Number Correct (Attempted)
Sum Of Scores
Sum Of Scores (Attempted)
Weighted Sum Of Scores
Weighted Sum Of Scores (Attempted)
Parameter Weighted Sum Of Scores
Parameter Weighted Sum Of Scores (Attempted)
Best K from N
Guessing Penalty
Weighted Guessing Penalty

Tabla 2. Algoritmos del estándar QTI

En nuestro caso, queremos plantear un caso de uso diferente, de mayor flexibilidad y complejidad que puede ser usado para la evaluación final o para auto evaluaciones periódicas por parte del alumno. Este caso de uso planteará preguntas de

distinto peso en función del grado de acierto anterior del alumno y la corrección del examen deberá tener en cuenta esta situación. Por este motivo, los algoritmos existentes en QTI no serán suficientes.

La utilización de pesos para valorar las preguntas será una herramienta fundamental a la hora de permitir a los educadores obtener del sistema los exámenes de calidad buscados. En los siguientes apartados analizaremos la generación y significado de estos pesos y el caso de uso y algoritmo planteados.

2.1. Preguntas con pesos

El peso asociado a una pregunta va a representar la relevancia de esa pregunta y del acierto o fallo en su respuesta. Por este motivo, es muy importante y difícil elegir correctamente este valor. La única fuente de información para obtener este valor a priori es la experiencia y conocimientos del docente que genera la base de preguntas a partir de la que se generan los tests.

Una forma de simplificar el establecimiento de la relevancia de una pregunta es la de atender a varios aspectos sobre la misma de forma independiente para luego reunir todos los valores en uno mediante un operador de agregación.

En nuestro caso, hemos considerado dos características de relevancia de cada cuestión, la importancia y la dificultad. Para cada una de ellas hemos establecido un rango de valores enteros de 1 a 5 que luego se combinan dando una matriz que se puede reducir por varios operadores de agregación. En la Figura 1A podemos observar la matriz de posibles valores, en la Figura 1B el resultado del operador de agregación media aritmética.

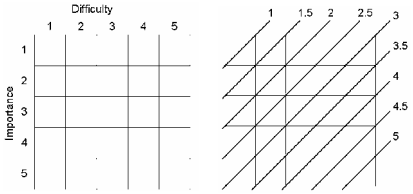


Figura 1. Relevancia de las Preguntas

En la actualidad, utilizamos el operador media aritmética como operador de agregación porque posee muy buenas características para nuestros fines no siendo la menor de ellas la facilidad de implementación.

2.2. Algoritmo de Evaluación

A continuación vamos a describir en detalle el caso de uso sobre el que hemos trabajado, así como ejemplos del mismo que nos den pie a presentar el algoritmo de evaluación utilizado.

El objetivo es que el sistema pueda realizar exámenes tipo test y evaluar a un alumno sin intervención del profesor. Pero, ¿qué le interesa al profesor?, que el alumno conteste de forma correcta a las preguntas de una determinada materia en varias fases con preguntas de relevancia creciente para dilucidar así su verdadero conocimiento en la materia no sólo en cantidad sino también en nivel del mismo. Para conseguir estos objetivos, el examen tipo test se realizará dividido en un número de fases concreto, en nuestro caso ese número será de cinco. Los exámenes, por tanto, estarán divididos en cinco fases con cuatro preguntas por fase. El funcionamiento será el siguiente: en la primera fase se harán preguntas de nivel 1, si el alumno responde correctamente una pregunta, en la siguiente fase se la harán preguntas de nivel 2, es decir, un nivel superior a la pregunta realizada con anterioridad y si el alumno no contesta o contesta incorrectamente una pregunta, en el nivel siguiente se le volverá a realizar una pregunta del mismo nivel de la pregunta que falló o no contestó. En cada nueva fase, el número de preguntas que avanzan de nivel será igual al número de preguntas que contestó correctamente y el número de preguntas que serán del mismo nivel será igual al número de preguntas que no superó. Esto puede dar lugar a una fase que incluya preguntas de cuatro niveles distintos en el caso de un alumno de conocimientos dispersos y de profundidad irregular en las distintas áreas de la materia.

2.3. Realizando un Test

El alumno que realice un test recibirá una nota numérica, con valor entre 0 y 10, que el sistema generará mediante una serie de operaciones aritméticas. Este valor será generado por el algoritmo de evaluación que

hemos desarrollado. Antes de presentar el algoritmo, vamos a introducir una serie de ejemplos de posibles situaciones que nos han servido, entre otros, para crear el algoritmo. El algoritmo, estará compuesto por varias fases. En una primera fase, se calcula un valor para el conjunto de respuestas de un alumno en el que se utiliza el peso asignado a cada una de las preguntas presentadas en la prueba. La fórmula para obtener estos valores se presenta en la figura 2.

$$\sum_{i} item_i \left(\frac{relevancia_i + 1}{2} \right)$$

Figura 2. Fórmula del Algoritmo de Evaluación

Como se puede apreciar en la fórmula, los pesos cuyo valor estaban entre 1 y 5, se modifican para que queden entre 1 y 3, suavizando el peso de los elementos de mayor nivel. La asignación de pesos se puede ver en la tabla 3.

Importancia	Peso
1	1
2	1.5
3	2
4	2.5
5	3

Tabla 3. Normalización Pesos/Importancia

Empezaremos por presentar una serie de ejemplos de situaciones que se pueden producir en el sistema tras la realización de un test por parte de un alumno. En primer lugar presentamos las situaciones más sencillas que marcan los tres límites principales en toda evaluación, el máximo, el mínimo y el valor frontera entre superado y no superado.

1 ✗	1 ✗	1 ✗	1 ✗	1 ✗
1 ✗	1 ✗	1 ✗	1 ✗	1 ✗
1 ✗	1 ✗	1 ✗	1 ✗	1 ✗
1 ✗	1 ✗	1 ✗	1 ✗	1 ✗

Figura 3. Resultado con 0 puntos.

1 ✓	2 ✓	3 ✓	4 ✗	4 ✗
1 ✓	2 ✓	3 ✓	4 ✗	4 ✗
1 ✓	2 ✓	3 ✓	4 ✗	4 ✗
1 ✓	2 ✓	3 ✓	4 ✗	4 ✗

Figura 4. Resultado con 18 puntos.

1 ✓	2 ✓	3 ✓	4 ✓	5 ✓
1 ✓	2 ✓	3 ✓	4 ✓	5 ✓
1 ✓	2 ✓	3 ✓	4 ✓	5 ✓
1 ✓	2 ✓	3 ✓	4 ✓	5 ✓

Figura 5. Resultado con 40 puntos.

Los ejemplos de la figuras 3, 4 y 5 muestran los resultados de 0 puntos (mínimo), 18 puntos (valor frontera) y 40 puntos (máximo) del algoritmo para un determinado test.

A continuación, se presentan una serie de ejemplos que ilustran situaciones potencialmente conflictivas, por su alineamiento con el valor frontera ya sea por exceso o por defecto.

1 ✓	2 ✓	3 ✗	3 ✗	3 ✗
1 ✓	2 ✓	3 ✗	3 ✗	3 ✗
1 ✓	2 ✓	3 ✗	3 ✗	3 ✗
1 ✓	2 ✓	3 ✗	3 ✗	3 ✗

Figura 6. Resultado con 10 puntos.

1 ✗	1 ✗	1 ✓	2 ✓	3 ✓
1 ✗	1 ✗	1 ✓	2 ✓	3 ✓
1 ✗	1 ✗	1 ✓	2 ✓	3 ✓
1 ✗	1 ✗	1 ✓	2 ✓	3 ✓

Figura 7. Resultado con 18 puntos.

1 ✓	2 ✗	2 ✓	3 ✓	4 ✗
1 ✓	2 ✓	3 ✗	3 ✓	4 ✗
1 ✓	2 ✓	3 ✓	4 ✗	4 ✓
1 ✗	1 ✓	2 ✓	3 ✗	3 ✓

Figura 8. Resultado con 20,5puntos.

En los ejemplos de la figuras 6, 7 y 8 se muestran los resultados numéricos de 10 puntos, 18 puntos y 20,5 puntos del algoritmo para un determinado test. No se incluyen otras situaciones no cercanas a la frontera porque su evaluación no ha aportado datos significativos en el análisis que permitió crear el algoritmo.

2.4. Evaluación de un test.

En algunos de los ejemplos nos hemos encontrado con un problema a la hora de dar la nota final al alumno, en la Figura 7, la puntuación obtenida es de 18 puntos, en teoría, una puntuación final de cinco. Pero vemos, que aún cuando el alumno ha contestado los tres niveles, éste ha necesitado más preguntas del nivel mínimo, fallando hasta 8 de estas, en este tipo de casos se desea penalizar la puntuación del ejercicio porque se aprecia que el alumno no posee conocimientos de base suficientes para superar la prueba. Esta penalización se realizará mediante la siguiente norma: si el alumno comete 8 o más fallos en los niveles 1,2 y 3 e independientemente de la nota que la fórmula matemática le otorgue, esa prueba será calificada como no superada.

Con la penalización incorporada al algoritmo, se ha conseguido plasmar dos ideas subyacentes en la mente del docente cuando corrige un test, la puntuación depende del número de respuestas correctas, pero marcadas con un cierto peso, y la necesidad de superar un mínimo del conocimiento de base en la materia, también representado por los pesos.

Una vez obtenida la valoración intermedia de una prueba, se procederá a su normalización para obtener la nota final entre 0 y 10. Esta normalización se realizará mediante el ajuste de los dos tramos de pendiente desigual de valores intermedios. Los dos tramos y el ajuste a realizar se presentan en las tablas 4 y 5.

Evaluación intermedia	Evaluación final
0	0
18	5

Tabla 4. Normalización, primer rango.

Evaluación intermedia	Evaluación final
18	5
40	10

Tabla 4. Normalización, segundo rango.

3. Conclusión

Las iniciativas de establecimiento de normativas de desarrollo de los distintos componentes de sistemas e-learning son muy importantes y el trabajo en los límites de las mismas más aún porque establece solidez a lo ya desarrollado. El escenario planteado y el algoritmo de evaluación aplicado, trabajan en esta zona del conocimiento para abrir nuevas vías de desarrollo a la normativa existente. Los resultados obtenidos hasta la fecha y las ideas presentadas aparecen claramente como uno de los caminos a seguir por las nuevas versiones de la normativa y por nuestros propios sistemas.

4. Agradecimientos.

Nos gustaría dar las gracias a nuestros compañeros del departamento de Ciencias de la Computación por la ayuda recibida para el desarrollo de este trabajo, especialmente al grupo de trabajo de e-learning y al de operadores de agregación.

Referencias

[1] IMS Global Learning Consortium. *IMS Question & Test Interoperability,* 2002

[2] LUVIT, 2002. *http://www.luvit.com*

[3] WebCT, 2002. *http://www.webct.com*

[4] Learning Space, 2002 *http://www.lotus.com/learningspace*

[5] Barchino, R et al. *EDVI: Un sistema de apoyo a la enseñanza presencial basado en Internet*. VII Jornadas de Enseñanza Universitaria de la Informática. Mallorca, España, 2001

[6] Aiken, L.R. *Test psicologógicos y evolución*. Prentice Hall Hispanoamericana, México, 1996

[7] Croket, L. and Algina J. *Introduction to Classical and Modern Test Theory.* Holt, Rinehart and Winston, 1986

Sistema para la (auto) evaluación de los alumnos a través de la Web

Sergio Luján-Mora
Departamento de Lenguajes y Sistemas Informáticos
Universidad de Alicante
Carretera de San Vicente del Raspeig s/n
E-03080 San Vicente del Raspeig, Alicante
e-mail: sergio.lujan@ua.es

Iván Mingot Latorre
Servicio de Informática
Universidad de Alicante
Carretera de San Vicente del Raspeig s/n
E-03080 San Vicente del Raspeig, Alicante
e-mail: ivan.mingot@ua.es

Resumen

En este artículo se presenta el *Examinador*, un sistema integrado dentro de la aplicación web de gestión universitaria *Campus Virtual* de la Universidad de Alicante que permite crear (el profesor), realizar (el alumno) y corregir automáticamente (el sistema informático) ejercicios de tipo test a través de Internet. Este sistema, que ya se encuentra implantado y disponible para toda la comunidad académica de la Universidad de Alicante, se puede emplear de dos maneras: por un lado, como sistema de auto evaluación para apoyar el aprendizaje de los alumnos; por otro lado, también se puede emplear como herramienta de evaluación por parte del profesor. En este artículo presentamos sus principales características, los beneficios que aporta, su arquitectura y dos experiencias de aplicación reales junto con la apreciación de los alumnos que lo han empleado.

1. Introducción

La expansión del uso de los ordenadores y la creciente implantación y abaratamiento de las comunicaciones de banda ancha (ADSL y cable) permiten la incorporación de nuevas técnicas y tecnologías a la enseñanza, ya sea presencial o a distancia (aunque esta última se puede beneficiar mucho más que la presencial). Prueba del interés que suscitan estos temas en la comunidad educativa es el hecho de que en JENUI 2002 uno de los temas estratégicos fue "Formación a distancia y entornos virtuales", con 11 ponencias presentadas, donde se trataron aspectos relacionados con la formación a distancia y el empleo de las nuevas tecnologías en la educación.

En la Universidad de Alicante (UA), aunque su carácter es presencial, se lleva potenciando desde hace más de seis años el empleo de Internet con el fin de mejorar el proceso educativo. Fruto de este esfuerzo es el desarrollo del *Campus Virtual* (CV), "un servicio de complemento a la docencia y a la gestión académica y administrativa, cuyo entorno es Internet y está dirigido tanto al profesorado como al alumnado y al personal de administración de la Universidad de Alicante" [6].

El CV de la UA presenta tres perfiles: profesor, personal de administración y alumno. A través del perfil de profesor se pueden realizar tareas de gestión (obtener listados de alumnos junto con la ficha particular de cada uno, publicar su horario de tutorías, gestionar anuncios de interés, etc.) como de docencia (publicar materiales, contestar tutorías, publicar noticias y avisos, proponer y moderar debates, etc.). Las últimas funcionalidades que se han incorporado al CV son el relleno electrónico de las preactas y la realización de exámenes a través de la Web.

En este artículo presentamos el subsistema del CV que permite realizar exámenes a través de la Web (a partir de ahora, *Examinador*). Mostraremos su arquitectura, sus principales características, su uso, los beneficios que puede aportar tanto al profesor como al alumno y dos experiencias de aplicación reales junto con la apreciación manifestada por los alumnos que lo han empleado. Este sistema, al estar integrado dentro del CV, veremos que ofrece una mayor flexibilidad y facilidad de uso que un sistema independiente y además incorpora una serie de características que permiten lograr una gran seguridad, lo que garantiza la autoría del examen. El Examinador se ofrece al profesorado como una

herramienta más de ayuda en la tarea docente y en ningún caso pretende sustituir otros métodos docentes.

El resto del artículo se ha dividido de la siguiente forma: en la sección 2 se comentan algunas propuestas relacionadas con la presentada en este artículo; en la sección 3 se explican las principales características del sistema; en la sección 4 se describe la arquitectura del sistema a grandes rasgos; en la sección 5 se comentan los principales beneficios del sistema; en la sección 6 se muestra la opinión de los alumnos que han empleado el sistema; en la sección 7 se comentan algunos de los trabajos futuros que nos hemos planteado; finalmente, la sección 8 cierra el artículo con las conclusiones.

2. Trabajos relacionados

En los últimos tres años, el uso de Internet como herramienta docente en la universidad ha aumentado espectacularmente. Prueba de ello son las numerosas ponencias presentadas en JENUI en los últimos años que emplean Internet, y en especial la Web, como una herramienta de apoyo a la docencia. Algunas de las propuestas relacionadas con el sistema que presentamos en este artículo son:

- En [1] se presenta un sistema que permite gestionar ejercicios tipo test con contenido multimedia. Este sistema es un programa autónomo que genera a partir de una base de datos ficheros HTML estáticos con el contenido de los ejercicios. Por tanto, no se trata realmente de una aplicación web.
- En [2] se presenta un sistema que permite publicar materiales docentes y, además, los alumnos pueden verificar los conocimientos adquiridos en cada tema, mediante la realización de exámenes de tipo test generados por el sistema. Sin embargo, presenta varias carencias como son: todos los test tienen 10 preguntas elegidas de forma aleatoria, todas las preguntas tienen que tener cuatro respuestas, etc.
- En [4] se presenta un sistema que permite al alumno acceder a listas de problemas generadas de forma aleatoria a partir de una base de datos. Sin embargo, esta propuesta carece de un módulo para preguntas-respuestas de tipo test con corrección automática: el alumno se descarga los ejercicios, los resuelve y posteriormente puede consultar la solución para auto evaluarse.
- En [5] se presenta un sistema parcial, ya que el examen está impreso y únicamente se emplea un formulario web como hoja de respuesta que se corrige automáticamente.
- Nosotros mismos presentamos el sistema AWAM en JENUI 2002 [3]. Este sistema tiene un sentido lúdico, ya que el objetivo es motivar a los alumnos a través de una competición que se establece entre ellos: para responder cada pregunta se dispone de un tiempo limitado y se pueden cometer como máximo un número dado de errores. Según el número de preguntas contestadas cada alumno recibe una puntuación que se refleja en una tabla con las puntuaciones de todos los alumnos participantes.

Todos los sistemas que se han comentado presentan dos inconvenientes comunes: son soluciones aisladas, donde el profesor tiene que realizar todas las tareas (por ejemplo, dar de alta a los alumnos) y las medidas de seguridad son muy escasas, lo que impide su empleo como sistema de evaluación. Sin embargo, el sistema que presentamos en este artículo está integrado con la herramienta de docencia CV, por lo que el profesor se ve descargado de la mayoría de las tareas administrativas, y cuenta con una gran cantidad de medidas de seguridad.

3. Descripción del sistema

Inicialmente, el Examinador nació como medio de auto evaluación para que fuese empleado por los alumnos de forma voluntaria cuando quisieran. Sin embargo, con una pequeña extensión se ha logrado un sistema que también permite al profesor evaluar a sus alumnos mediante exámenes de tipo test.

El interfaz del Examinador está preparado, al igual que todo el CV, para trabajar con varios idiomas. En la actualidad, según las preferencias que manifieste cada alumno, se puede emplear en castellano o valenciano, los idiomas oficiales en la UA.

El Examinador posee dos perfiles: profesor y alumno. Según el perfil que se emplee, estarán disponibles unas opciones u otras. A la hora de crear un test, el profesor dispone de diversos

parámetros que permiten su configuración. Por último, el sistema cuenta con varias medidas de seguridad que permiten su uso como sistema de evaluación.

3.1. Perfil del profesor

El profesor puede realizar las siguientes tareas:

- Añadir un nuevo ejercicio (examen).
- Editar y borrar un ejercicio ya existente.
- Asignar un ejercicio a un grupo de alumnos.
- Añadir una pregunta con sus respuestas a un ejercicio.
- Editar y borrar una pregunta y sus respuestas.
- Ver los ejercicios (sin y con solución) creados hasta el momento.
- Ver los resultados de un ejercicio, por pregunta o por alumno.

3.2. Perfil del alumno

El alumno puede realizar las siguientes tareas:

- Realizar un ejercicio.
- Rehacer un ejercicio (si lo permite el profesor).
- Consultar los resultados obtenidos en un ejercicio, con la posibilidad de ver o no ver su propia solución y la solución correcta.

3.3. Parametrización

El Examinador dispone de diversos parámetros que permiten al profesor configurar de forma individual cada ejercicio (examen). Los principales parámetros son (Figura 1 y Figura 2):

- *Número de preguntas a visualizar por página*: las preguntas se pueden mostrar todas juntas en una única página o separadas en varias páginas.
- *Número de preguntas mal respondidas… restan número de preguntas bien respondidas*: se puede fijar una penalización para evitar que se responda de forma aleatoria.

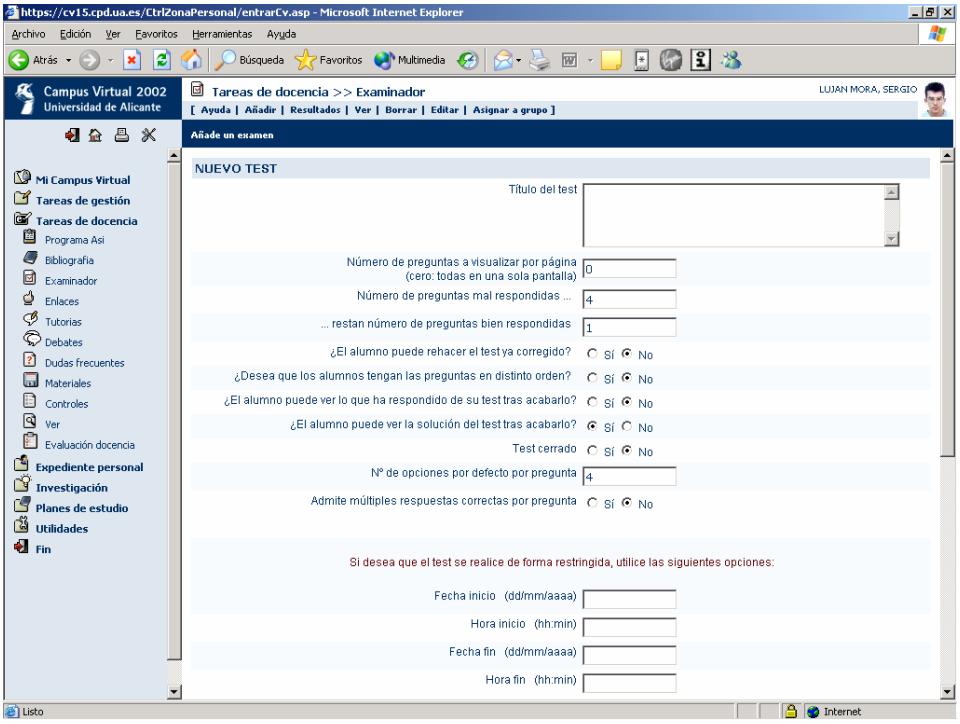


Figura 1. Parametrización de un ejercicio (1)

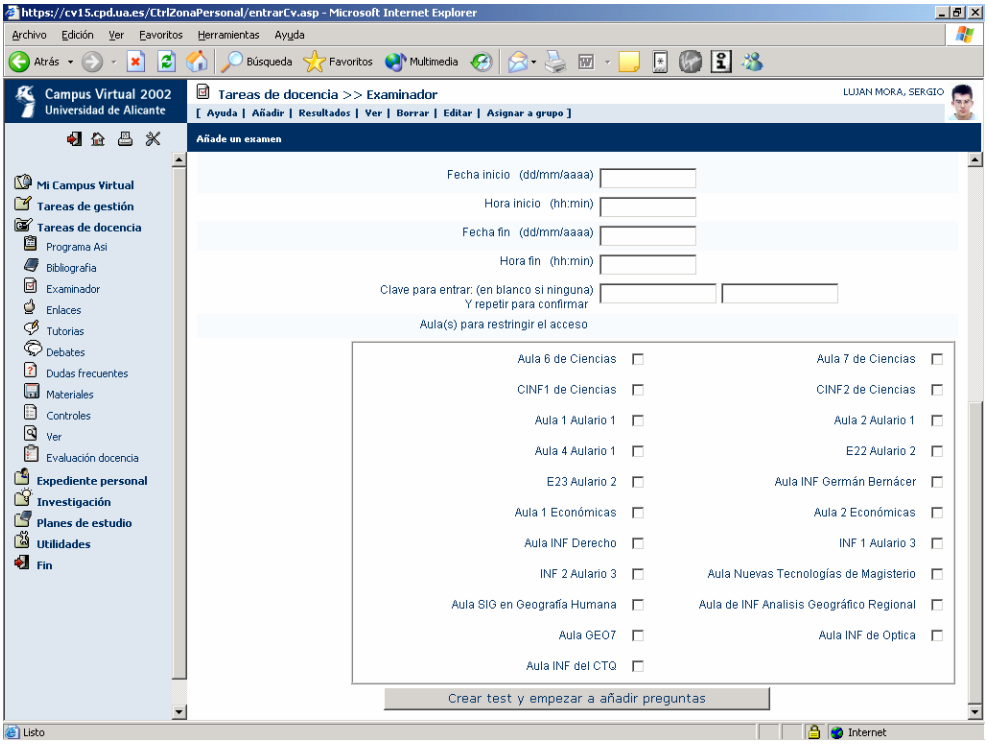


Figura 2. Parametrización de un ejercicio (2)

- *¿El alumno puede rehacer el test ya corregido?*: tiene sentido cuando se emplea el sistema como auto evaluación.
- *Nº de opciones por defecto por pregunta*: inicialmente todas las preguntas tienen el mismo número de respuestas (opciones), pero luego se pueden modificar de forma individual.
- *Admite múltiples respuestas correctas por pregunta*: pueden haber preguntas con varias respuestas correctas.

3.4. Medidas de seguridad

En el caso de emplear este sistema como medio de evaluación es necesario garantizar la legitimidad del proceso de evaluación. Para ello el sistema contempla varias medidas de seguridad que evitan la realización de un examen de forma indebida (copias y suplantación de identidad):

1. Toda la comunicación a través de Internet entre el navegador del usuario y el servidor web que aloja el CV se realiza por medio del protocolo SSL (*Secure Socket Layer*) que garantiza la seguridad de las transmisiones[1].
2. El acceso al CV para realizar un examen se realiza previa identificación del usuario, mediante su DNI y un código personal secreto (modificable por el usuario) que garantiza la privacidad.
3. El profesor debe de comprobar la identidad de cada alumno[2]. En la pantalla del ordenador le aparece la fotografía del alumno, que puede emplear para comprobar que el alumno es quien dice que es.
4. Se puede fijar una fecha y una hora de inicio y

[1] Se emplea una clave pública de 512 bits.

[2] Evidentemente, esta comprobación se "debería" de hacer en cualquier tipo de examen. Pero en los exámenes escritos, siempre cabe la posibilidad de realizar un estudio grafológico del examen de un alumno para comprobar su autoría en caso de duda.

finalización del examen, de forma que sólo se pueda contestar durante ese período, tal como se ve en la Figura 1.

5. Además, el profesor puede mantener un examen cerrado y sólo abrirlo cuando lo considere indicado (Figura 1).
6. También se puede establecer una clave para que los alumnos puedan entrar a realizar el examen (Figura 2). Esta clave se la comunica el profesor a los alumnos en el momento de comenzar el examen.
7. También se puede restringir el acceso a la realización del examen a una serie de aulas (direcciones IP de los ordenadores), de modo que sólo se pueda realizar desde ellas (Figura 2).
8. Durante la realización del examen el profesor dispone de una opción que le permite consultar en cualquier momento la lista de alumnos que están realizando un examen.
9. Respecto al examen en sí, el profesor puede asignar a cada grupo de alumnos un modelo de examen distinto.
10. Finalmente, el orden de presentación de las preguntas a cada alumno se puede fijar para que sea aleatorio (Figura 1), de modo que sea más difícil que se copien las respuestas.

3.5. Resultados

Al finalizar un ejercicio, el profesor puede consultar los resultados de los alumnos de dos formas: por pregunta o por alumno. En el primer caso (Figura 3), para cada pregunta se muestra la respuesta correcta, el número de alumnos que han respondido la pregunta y el número de alumnos que han seleccionado cada una de las respuestas; además, también se muestra la máxima nota, la nota mínima y el promedio. En el segundo caso, se muestra la nota que ha obtenido cada alumno y se pueden visualizar las respuestas de un alumno concreto.

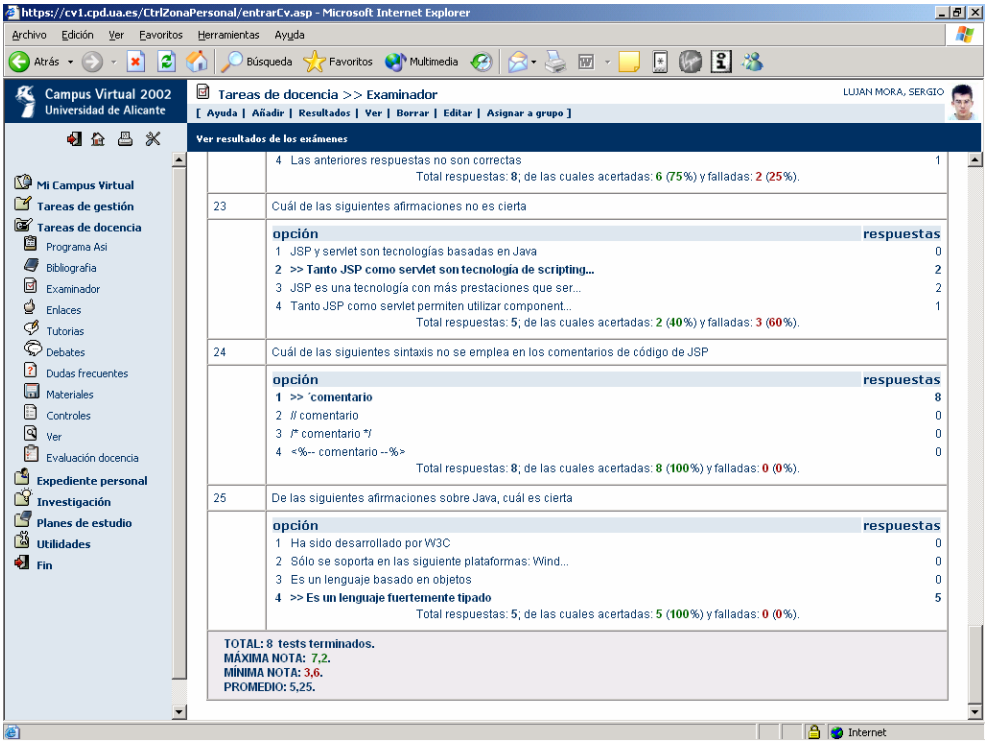


Figura 3. Resultados de un ejercicio

4. Arquitectura de la aplicación

La arquitectura del Examinador comprende tres niveles: lógica de presentación, lógica de negocio (aplicación) y lógica de datos.

La lógica de presentación, encargada de interactuar con el usuario, se ha programado mediante HTML, CSS y JavaScript. Esta parte de la aplicación se ejecuta en el cliente (navegador web) y se ha intentado hacer lo más compatible posible. Su correcto funcionamiento se ha verificado en los navegadores más conocidos: Microsoft Internet Explorer, Netscape Communicator y Opera.

La lógica de negocio, encargada de controlar el acceso a los exámenes, seleccionar las preguntas, verificar las respuestas, etc., se ha programado mediante ASP en un servidor web Microsoft Internet Information Server 5.0 en el sistema operativo Microsoft Windows 2000 Server. Una parte importante de la lógica de negocio también reside en el sistema gestor de bases de datos (SGBD): una serie de procedimientos almacenados (*stored procedures*) y de disparadores (*triggers*) existentes en la base de datos realizan el control de las reglas de negocio (por ejemplo, un alumno sólo puede hacer un examen si está matriculado en un grupo asociado al examen, si el examen está cerrado nadie lo puede realizar, etc.).

Por último, para la lógica de datos, encargada de gestionar los datos y una parte de las reglas de negocio, se ha empleado el SGBD Oracle 8i en el sistema operativo IBM AIX 4.3.

5. Beneficios del sistema

Los beneficios que creemos que se pueden lograr con este sistema los hemos clasificado según como se emplee:

1. Como sistema de auto evaluación.
   - De cara al alumno, puede ayudar a la mejora de la satisfacción y del rendimiento efectivo de los estudiantes, ya que la realización de ejercicios corregidos automáticamente permite la correcta evaluación de los conocimientos que posee y, por tanto, el descubrimiento por sí mismo de aquellas carencias que presenta. Además, si el profesor plantea los ejercicios de auto evaluación como un medio de entrenamiento previo a la realización del examen, también puede ayudar al alumno a aumentar la seguridad con la que se enfrenta a un examen al conocer el nivel de conocimiento que se le va a exigir a la hora de examinarse. Por otro lado, las ventajas de emplear Internet como medio de publicación de ejercicios son de sobra conocidas: los alumnos puede resolver los ejercicios cuando ellos quieran (la aplicación está disponible las 24 horas del día) y desde donde ellos quieran (se puede acceder a la aplicación desde cualquier ordenador con acceso a Internet).
   - De cara al profesor, puede ayudarle a realizar un seguimiento de sus alumnos más continuo y personalizado: número de ejercicios de auto evaluación realizados por cada alumno, resultados obtenidos, etc. A partir de estos datos, el profesor puede reorientar su docencia para repasar aquellos aspectos que susciten más dudas, modificar el ritmo de las explicaciones, citar a aquellos alumnos con peores resultados en su horario de tutorías, etc.
2. Como sistema de evaluación.
   - De cara al alumno, es un sistema cómodo y rápido, que le permite conocer la calificación obtenida nada más acabar el examen (si así lo desea el profesor).
   - De cara al profesor, reduce en gran medida las actividades que tiene que realizar para evaluar a sus alumnos, ya que una vez introducido un conjunto de preguntas le permite crear exámenes en pocos minutos y la corrección de las respuestas de los alumnos es automática, instantánea y sin posibilidad de cometer errores. De este modo, puede aumentar su productividad y dedicar más tiempo a otras tareas (preparar materiales, responder tutorías, etc.).
   - De cara a la administración, supone una considerable reducción de costes, ya que se evita el empleo de hojas de respuesta de lectura óptica y el posterior trabajo de lectura y corrección.

6. Apreciación de los alumnos

El Examinador se ha empleado con éxito en dos exámenes de la convocatoria de diciembre 2002 en la UA: "Programación en Internet" (PI) con 8 alumnos, asignatura optativa de Ingeniería en Informática, y "Gestión de la Información V" (GIV) con 14 alumnos, asignatura obligatoria del Título propio de primer ciclo en Estudios Inmobiliarios. Se han elegido estas dos asignaturas para observar el comportamiento de los alumnos con distinto perfil: en el primer caso, se trata de alumnos expertos en el manejo de ordenadores y con avanzados conocimientos en programación; en el segundo caso, se trata de alumnos con un nivel de experiencia medio o bajo en el manejo de ordenadores.

En el primer examen (PI) se detectaron varios errores (*bugs*) que impidieron que los alumnos pudiesen ver la calificación obtenida al final del examen (se la tuvo que comunicar el profesor de forma individual). En el segundo examen (GIV), no hubo ningún problema, ya que los errores detectados habían sido corregidos.

En ambos casos, al finalizar el examen se les pasó una encuesta anónima para recabar su opinión sobre el Examinador. En la Tabla 1 se recogen las preguntas de la encuesta junto con los porcentajes de respuesta. En ambos casos los valores obtenidos son muy similares, aunque se aprecian diferencias en la pregunta 3 (debido a los errores detectados en el primer examen) y en la pregunta 8 (los alumnos de GIV creen que es más fácil copiarse). En general, la opinión de los alumnos sobre el sistema es muy favorable: les gusta, lo consideran fácil de emplear y les interesa que se pueda emplear el sistema como auto evaluación.

	Pregunta	Sí (%)		No (%)		No sabe (%) No contesta	
		PI	GIV	PI	GIV	PI	GIV
1	¿Te ha gustado el sistema?	87,5	92,3	12,5	7,7	0	0
2	¿Consideras que es sencillo de emplear?	100	84,6	0	7,7	0	7,7
3	¿Has tenido alguna dificultad en la realización del examen?	25	7,7	75	92,3	0	0
4	¿Consideras interesante que se generalice el empleo de este sistema a otras asignaturas?	75	84,6	25	15,4	0	0
5	¿Te gustaría que existieran exámenes de auto evaluación para poder realizarlos cuando quieras (en esta o cualquier otra asignatura)?	100	100	0	0	0	0
6	Si existieran exámenes de auto evaluación, ¿crees que los emplearías periódicamente?	100	100	0	0	0	0
7	¿Te da garantías este sistema de evaluación "sin papeles"?	87,5	84,6	0	15,4	12,5	0
8	¿Crees que es fácil "copiarse" durante la realización de un examen con este sistema?	0	30,8	100	69,2	0	0

Tabla 1. Apreciación de los alumnos

Por último, también podían escribir cualquier comentario o crítica que nos quisieran hacer llegar. Muchos de los comentarios se refirieron a la pregunta 8: si existe la suficiente separación entre los alumnos y se cambia el orden de las preguntas, la única forma de copiarse es buscando la información a través de Internet. También nos indicaron que era más cómodo y rápido responder a través del ordenador. Como curiosidad, algunos alumnos nos indicaron que preferían no conocer la nota nada más acabar el examen (!).

7. Trabajos futuros

El sistema que se ha desarrollado se encuentra ya en funcionamiento y está disponible para todo el personal docente de la UA. Sin embargo, en las primeras experiencias reales de evaluación en varias asignaturas, se han detectado algunas

mejoras en las que se está trabajando en la actualidad:

- Introducción de la misma pregunta en distintos idiomas, para que cada alumno pueda leerlas en el idioma que más le interese.
- Soporte de otro tipo de preguntas: de relleno de huecos, de asociación de conceptos, de selección de valores, etc.
- Posibilidad de enlazar cada pregunta con los temas teóricos donde se explican.
- Con el fin de reducir aún más las probabilidades de que los alumnos se copien en la realización de un examen, se va a añadir la posibilidad de que el orden de las repuestas de cada pregunta también varíe de un examen a otro. Sin embargo, en principio esto impedirá que se introduzcan respuestas del tipo "Las respuestas a) y b) son verdaderas"[3].
- Inclusión de imágenes en las preguntas y respuestas.
- Relleno automático de las preactas a partir de los resultados del examen. Desde hace dos años se pueden rellenar las preactas de forma electrónica a través del CV en la UA. Este método ahorra costes (no hay que imprimir las preactas en papel de lectura óptica), reduce tiempos (se elimina el envío y recepción de las preactas por medio del correo interno) y elimina problemas (pérdida de preactas, posibilidad de manipulación por personas no autorizadas, etc.).
- Estadísticas de resultados más avanzadas. Por ejemplo, se va a incorporar la posibilidad de que un alumno pueda comparar los resultados que ha obtenido en un ejercicio con la media de los resultados de sus compañeros. De este modo, podrá saber en qué preguntas ha acertado (o fallado) más que sus compañeros.

8. Conclusiones

En este artículo hemos presentado el *Examinador*, un subsistema del *Campus Virtual* de la Universidad de Alicante que permite crear, realizar y corregir automáticamente ejercicios tipo test a través de Internet. El Examinador se puede emplear de dos formas: como medio de auto evaluación o para realizar exámenes.

Este sistema ya ha sido empleado con éxito varias veces para realizar exámenes. Las principales ventajas de su uso son que reduce el tiempo necesario en la elaboración de los exámenes y elimina las fases de corrección y publicación de calificaciones. Por otro lado, las opiniones de los alumnos muestran que están muy interesados en su uso y perciben sus ventajas.

[3] Para resolver este problema se está contemplando el empleo de etiquetas que hagan referencia al número de respuesta en el momento de la definición de la pregunta.

Agradecimientos

Deseamos agradecer al programador Fulgencio Sanmartín Martínez del Servicio de Informática de la Universidad de Alicante su gran labor llevada a cabo durante el desarrollo del sistema de evaluación a través de la Web.

Referencias

[1] Antelm, J.M.; Mollá, R.; Vivó, R. y Vidal, V. *Programa Genérico de Evaluación*. En VII Jornadas de Enseñanza Universitaria de la Informática, páginas 384-389, 2001.

[2] Barchino, Roberto; Gutiérrez, J.M.; García, Elena y Hilera, J. Ramón. *EDVI: Un sistema de apoyo a la enseñanza presencial basado en Internet*. En VII Jornadas de Enseñanza Universitaria de la Informática, páginas 451-453, 2001.

[3] Luján-Mora, Sergio y Llopis, Fernando. *Resolución de ejercicios de programación en la web*. En VIII Jornadas de Enseñanza Universitaria de la Informática, páginas 29-36, 2002.

[4] Más, Ramón y Lacosta, Ignacio. *Aplicaciones de Internet a la Enseñanza: Un Sistema de Autoevaluación*. En VII Jornadas de Enseñanza Universitaria de la Informática, páginas 500-503, 2001.

[5] Pavón, Nieves; Cano, José Ramón; Márquez, Francisco y Sainz, Alfredo. *SCRAE'Web: Sistema de Corrección y Revisión Automática de Exámenes a través de la WEB*. En VIII Jornadas de Enseñanza Universitaria de la Informática, páginas 231-235, 2002.

[6] Servicio de Informática de la Universidad de Alicante. *Campus Virtual*. Disponible en: http://www.ua.es/es/univirtual/index.html.

SAM: Sistema de Autoevaluación Multimedia

J.M. Antelm, R. Mollá, R. Vivó, V. Vidal
Departamento de Sistemas Informáticos y Computación
Universidad Politécnica de Valencia
Camino de Vera, s/n
46022 – Valencia
rmolla@dsic.upv.es

A. Robles
Departamento de Informática de Sistemas y Computadores
Universidad Politécnica de Valencia
Camino de Vera, s/n
46022 – Valencia
arobles@disca.upv.es

M. L. Gil
Departamento de Idiomas
Universidad Politécnica de Valencia
Camino de Vera, s/n
46022 Valencia
mlgil@idm.upv.es

Resumen

SAM es una herramienta informática que permite trabajar tanto en un entorno colaborativo en la intranet de un centro educativo mediante un interfaz de ventanas como a través de Internet empleando un interfaz web.

SAM trabaja sobre información almacenada en una base de datos de conocimiento y sobre materiales multimedia depositados en Internet o en repositorios propios.

La herramienta permite al alumno acceder a la teoría de la asignatura y practicar sobre los contenidos propuestos adaptando dicha práctica a sus necesidades particulares. El sistema permite también seleccionar partes concretas de la asignatura, nivel de dificultad, tipos de recursos multimedia a utilizar... SAM cohesiona fuertemente teoría y práctica de la misma materia o entre materias afines en la propia base de datos de conocimiento. El mismo sistema puede soportar diferentes bases de datos con múltiples asignaturas cada una. También permite al alumno consultar sus estadísticas, constituyendo éstas una información de realimentación que utilizará para continuar planificando su propio aprendizaje.

1. Introducción

La introducción de las tecnologías de la información en el trabajo y en particular en el procesamiento de textos, ha conseguido incrementar la calidad visual del producto final, así como la facilidad de creación de los documentos. El abaratamiento de la tecnología ha contribuido a su popularización, consiguiendo en la práctica que casi toda la documentación que se produce actualmente sea confeccionada por medios electrónicos [1]. El mundo de la educación no ha sido una excepción a esta tendencia, permitiendo a muchos docentes universitarios confeccionar y emplear materiales en soporte electrónico tanto en el aula como a disponer de ellos en páginas web, transformarlos en documentos hipertexto, etc [2].

En la primera fase de popularización de Internet en el ámbito universitario hubo un esfuerzo muy importante por transcribir los contenidos docentes de su formato tradicional a un formato electrónico pasivo e igualmente inerte, desaprovechando el carácter dinámico del medio. La introducción de materiales multimedia, permitió ganar en dinamismo. Aunque más vistosos y atractivos, estos contenidos seguían adoleciendo de una interacción con el alumno poco flexible, evitando su participación activa en el proceso del aprendizaje. No obstante, estos sistemas, siguen esquemas rígidos de organización, que una vez conocidos, ya no dan pie a la innovación, quedando como meras versiones electrónicas más o menos sofisticadas de los apuntes y transparencias tradicionales [3].

Actualmente, la tendencia se encamina hacia la gestión automática, sin intervención directa y explícita del profesor, poniendo los contenidos a disposición de los alumnos de acuerdo a sus necesidades educativas personalizadas, evitando la monotonía en la forma de presentación con el fin

de incrementar la motivación, el interés y la curiosidad del alumno.

Un sistema que pretenda servir de orientación en el proceso de (auto)aprendizaje debe adaptarse a las necesidades de cada alumno, debe resultar motivador, y además debe poder informarle del nivel adquirido, proporcionándole al alumno una realimentación que le permita reorientar su práctica de aprendizaje [4].

Los procesos de enseñanza-aprendizaje han sufrido grandes modificaciones, especialmente los ligados a los cambios que se van originando conforme se va generalizando el uso de Internet [5]. Muchas instituciones educativas han comenzado a utilizar la web como herramienta de apoyo al proceso de enseñanza-aprendizaje, aplicando las nuevas tecnologías en áreas como la teleformación y el autoaprendizaje. La mayoría de ellas la utilizan como sistema de distribución de información y de formación, proporcionando al alumno acceso a los contenidos teóricos necesarios y material de apoyo de tipo práctico, así como la posibilidad de establecer comunicación con un profesor de consulta que pueda resolver sus dudas. En este sentido, cabe destacar las encomiables iniciativas que están llevando a cabo:

- La UOC (Universitat Oberta de Catalunya – http://www.uoc.es)
- El sistema SED de la Universidad Miguel Hernández de Elche (http://www.umh.es)
- El sistema de telenseñanza integral de la Univesitat Politècnica de Mataró
- La herramienta de evaluación a disposición del profesorado universitario de la Universidad de Navarra (http://www.uva.es/aufop/publica/revelfop/99-v2n1.htm)

Por otra parte, hay una gran cantidad de herramientas informáticas que asisten al docente en la realización de todo tipo de ejercicios para sus alumnos. Muchas son las posibilidades que éstas ofrecen, como el soporte de diferentes y variados tipos de ejercicios, la autocorrección, enviar respuestas a un instructor para que las corrija, soporte de archivos multimedia, etc. De entre todas estas iniciativas que han aparecido en el mercado se cita a modo de ejemplo:

- HotPotatoes. http://www.halfbakedsoftware.com y http://web.uvic.ca/hrd/hotpot
- Discovery School. http://school.discovery.com
- Interactive Exercise Makers. lang.swarthmore.edu/makers/indexold.htm
- Quia!. http://www.quia.com/
- Quiz Star
- Web Author. http://aware.hwg.org/tools/
- WebPractest. http://www.wm.edu/CAS/modlang/gasmit/webpractest/

La mayoría de las iniciativas universitarias, así como los programas comerciales, se decantan claramente por un entorno de trabajo que obliga a la conexión permanente a Internet para poder realizar la autoevaluación. Por otro lado, las herramientas comerciales presentan una tecnología propietaria que no permiten su desarrollo ni el control de los fuentes para poder adaptarlas al entorno o las características de sus usuarios.

SAM es una herramienta en la que las fuentes han sido desarrolladas en la UPV y, por tanto, los usuarios somos los propietarios de la tecnología. SAM permite trabajar tanto a través de Internet como a través de la intranet de un centro educativo o directamente en modo local, fuera de línea, en casa, empleando los mismos soportes de conocimiento.

El mismo programa puede ser empleado tanto por el profesor como por los alumnos (versión con menos funcionalidad) tanto en clase como en casa, siendo en este caso una herramienta que complementa y evalúa los contenidos adquiridos.

2. SAM

SAM es el resultado de la evolución del proyecto anterior denominado MGA (Motor Genérico de Autoevaluación) [6] al que se le ha dotado de un interfaz web que amplía la forma en la que los alumnos pueden acceder al contenido de información.

SAM pretende facilitar al alumno la evaluación autónoma. Para ello, se exige al docente organizar dichos contenidos en temas o módulos concretos que a su vez pueden dividirse en subtemas de forma jerárquica e indefinida. Así mismo, los contenidos prácticos se asocian a esos subtemas junto con la teoría.

SAM está dividido en tres módulos o programas que interaccionan de forma semejante entre sí y que comparten una misma base de datos y recursos multimedia. Estos tres módulos son el MGA versión completa, el MGA versión simplificada y el WebMGA. Véase la Figura 1.

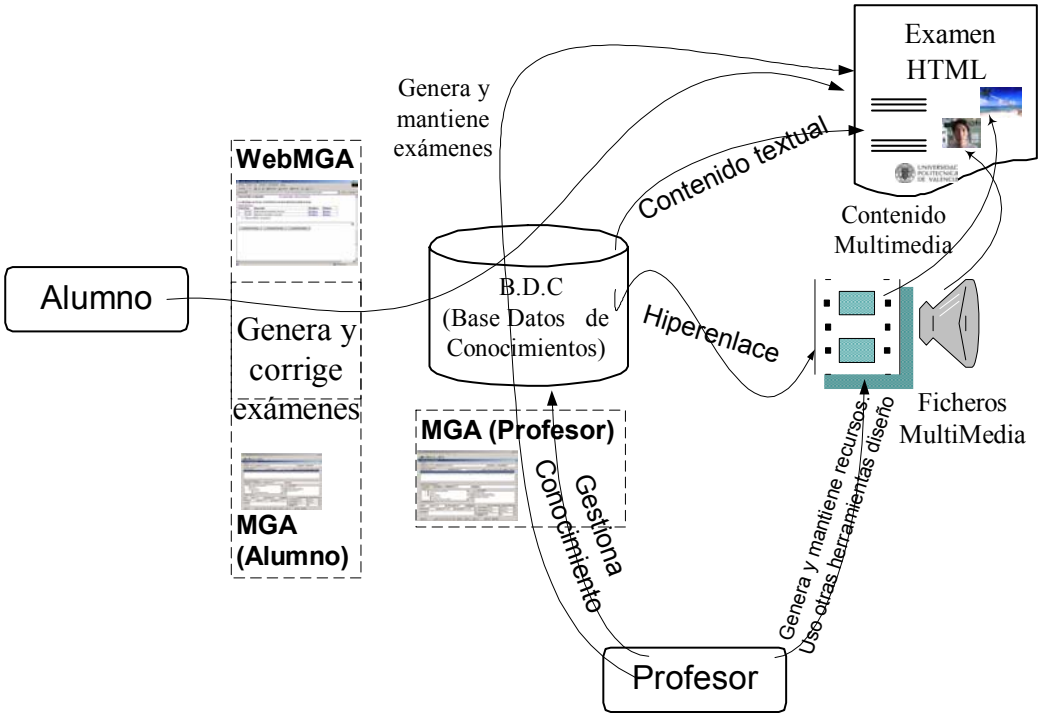


Figura 1. Esquema general de funcionamiento del Sistema de Autoevaluación Multimedia

Los tres tipos de programas acceden a la misma base de datos donde se almacena el conocimiento de la asignatura. Tanto el MGA completo como el MGA simplificado, acceden a la BD en modo local o a lo sumo, a la BD existente en un servidor (aula informática). En el caso del WebMGA, el usuario accede a través de Internet al conocimiento existente en un servidor remoto y que no es visible por parte del alumno, sino sólo por el profesor, aunque sí sus resultados (evaluaciones). La base de datos está dividida en varias áreas:

- **Conocimiento**. Se organiza por asignaturas. Cada una de ellas contiene un árbol de profundidad no acotada que organiza todos los temas y subtemas asociados.
- **Prácticas**. Se pueden gestionar las *preguntas* y las *respuestas* asociadas por separado. Cuando se asocia una pregunta con una respuesta, se crea una *cuestión*. Varias cuestiones forman un *ejercicio*. Varios ejercicios, incluso de diferentes tipos, forman una prueba objetiva o *examen*. Cada contestación al examen es una *realización* que se guarda en la base de datos para que el propio profesor o alumno pueda realizar estadísticas y un seguimiento más personalizado de la evolución.
- **Alumnos**. Contienen los datos personales y las referencias necesarias para realizar posteriormente la evaluación y el seguimiento estadístico.

Cada (sub)tema de teoría tiene asociado un recurso hipertexto que lo desarrolla, el cual a su vez puede contener otros recursos multimedia o hiperenlaces a otros temas, con independencia de que éstos, a su vez, estén o no registrados en la base de datos.

Cada pregunta o respuesta puede ser expresada en diferentes formatos que van desde el texto plano hasta imagen animada con audio incorporado. Una misma pregunta o respuesta puede estar expresada simultáneamente en diferentes formatos e incluso repitiendo diferentes contenidos (grados de dificultad) con el mismo formato. Así pues, una pregunta puede estar almacenada en formato WAV de audio o MP3, en formato AVI, texto o JPEG e incluso pueden haber varios ficheros AVI que contengan la misma pregunta. Sería el caso de una asignatura de idiomas en el que cada AVI se realiza en contextos diferentes dependiendo de la dificultad de la pregunta: en entorno silencioso, en el campo, en una calle ruidosa, con diferentes acentos, etc.

Las preguntas y respuestas están siempre asociadas a un tema de teoría. De esta forma, cuando se genera una prueba no evaluatoria, se puede incluir un hiperenlace a la teoría asociada para consultar dudas cuando no se sabe cómo responder la pregunta. El programa genera ficheros HTML con el contenido de las pruebas a realizar. Dependiendo de si se activa la consulta en línea (siempre desactivada en los exámenes reales), se puede consultar la teoría asociada a esa pregunta, recibir mensajes de ayuda, o comprobar cómo se podría resolver el ejercicio, la respuesta correcta,...

Por último, indicar que los recursos referenciados por documentos hipertexto teóricos pueden ser también referenciados desde los exámenes para generar las preguntas o respuestas; incluso entre diferentes temas o asignaturas, incrementando enormemente la productividad del sistema.

Los tres programas utilizan recursos multimedia generados mediante el uso de otros programas de edición de contenidos de imagen, vídeo o sonido y que residen en directorio locales a los servidores o disponibles en otras URL en Internet.

Se ha desarrollado la herramienta pensando en dos tipos de usuario: el alumno y el profesor, con dos perfiles claramente definidos y distintos, y en dos formas de acceso: en modo local (propio PC o intranet del centro) y modo remoto a través de Internet. Seguidamente se procede a presentar más detalladamente cada uno de estos programas que conforman en su conjunto el proyecto SAM.

2.1. M.G.A. Motor Genérico de Autoevaluación

El MGA, en su versión completamente funcional, es la versión del programa que normalmente utiliza el profesor, ya que permite un control completo de la base de datos. Se puede incluir distintas asignaturas en una misma base de datos. Cada asignatura dispone internamente de unos contenidos teóricos organizados de acuerdo con la estructura de la materia. Cada registro enlaza con los contenidos teóricos mediante hiperenlaces. Cuanto más descrito y pormenorizado sea el árbol de contenidos de la materia, de más versatilidad se dispondrá para poder realizar preguntas, desglosar contenidos, asociar a recursos externos, etc.

Asímismo, el material de práctica asociado a dichos contenidos teóricos es parametrizable en cuanto a nivel de dificultad, tipo de ejercicio, referencias teóricas, formatos multimedia permitidos y prohibidos, etc.

La utilización de este programa obliga a realizar *un único esfuerzo* de creación de contenidos en formato hipertexto susceptibles de ser gestionados por cualquier navegador, bien a través de un servidor web, bien fuera de línea desde un CD-ROM. Los materiales multimedia teóricos pueden ser reutilizados en la parte de evaluación como parte de las preguntas y respuestas, entre temas de la misma asignatura e incluso, entre diferentes asignaturas.

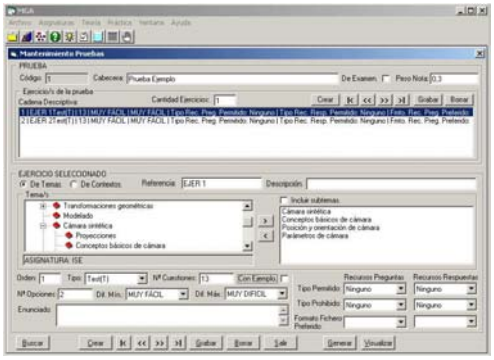


Figura 2. Detalle del MGA completo, versión de profesor

Esta es una herramienta abierta de código fuente libre, de forma que cualquier persona pueda aportar una mejora y sea toda la comunidad académica la que se responsabilice de su

mantenimiento y crecimiento. No obstante, no comparte licencia GNU ya que ha sido desarrollada sobre tecnologías propietarias Microsoft. Actualmente, se puede obtener la última versión de la herramienta en http://www.sig.upv.es/****

Su versatilidad es alta, ya que se soporta cualquier tipo de media existente actualmente y la incorporación de los nuevos que vayan apareciendo dependerá de la velocidad con la que dicho media se soporte por los navegadores de Internet habituales.

El programa es multidisciplinar, ya que no se requiere una versión de programa para cada asignatura o contenido, sino que permite la docencia de todo tipo de asignaturas, así como que dicha docencia pueda ser orientada a todo tipo de alumno (universidad, empresa, academia, educación primaria o secundaria, etc).

Es versátil, ya que permite ser configurado por el usuario, en función de niveles de dificultad, temario, asignaturas, contenidos, medias, cantidad, tipo de evaluación, etc.

El programa posibilita la formación autónoma y a distancia, sin que sea necesaria la presencia física del docente, a menos que el alumno lo requiera. El MGA permite al alumno plantearse casos reales, ejercicios y resolverlos por su cuenta, pudiendo averiguar la teoría que subyace detrás, consejos o resoluciones alternativas detalladas en formato multimedia por el profesor previamente.

Al utilizar el motor de bases de datos de MS-Access permite el acceso concurrente multiusuario, pudiendo ser utilizada tanto por profesores como por alumnos. Por ello, facilita el proceso de gestión del conocimiento, la generación de exámenes y su evaluación por parte de los profesores.

El desarrollo de esta herramienta se ha realizado atendiendo prioritariamente a la facilidad de uso por parte de personal no experto ni profesional, tanto en la elaboración de los contenidos como en el acceso a los mismos, en la utilización por parte de los alumnos, navegación dentro del conocimiento y enlace directo entre la teoría y la práctica.

El MGA debe adaptarse a personas con discapacidades generando exámenes expresados sólo con voz (ciegos o dioptrías elevadas) o sin sonido (sordos), sólo imagen estática o dinámica (niños, disminuidos psíquicos,...) o cualquier otro tipo de restricción que se considere. El objetivo final es incentivar la participación del alumno en su propia formación, implicándole de forma mucho más activa en el proceso de aprendizaje.

El programa permite la generación guiada y automática de pruebas o cuestionarios basándose en el conocimiento almacenado en la base de datos. Esta base de datos almacena información de tipo textual, así como referencias a los diferentes ficheros multimedia en los que se puede expresar un determinado conocimiento. El profesor es el que genera los contenidos multimedia, utilizando las herramientas habituales de edición de imagen, vídeo, páginas web,... o reutilizando los generados por colegas o suministrados a través de otros canales. En este sentido, esta herramienta no viene a reinventar o sustituir a ninguno de los programas habituales de generación de contenidos multimedia, sino a utilizar el resultado producido por dichos programas e integrarlos en un cuerpo coherente y uniforme de contenidos organizados.

Los exámenes generados por el MGA utilizan tecnología Internet. En realidad no son más que ficheros HTML que referencian a los ficheros multimedia anteriores y que también pueden incluir código JavaScript, Java, animaciones FLASH o cualquier otro tipo de tecnología actualmente utilizada o que en el futuro se pudiera utilizar.

Como el fichero de examen está generado por el MGA, éste puede cambiar su aspecto de forma aleatoria o de acuerdo con los criterios marcados por el usuario. De esta forma, la apariencia de los exámenes cambia cada vez, reduciéndose la posibilidad de responder por asociación o recuerdo a las preguntas actuales basándose en las anteriores.

El objetivo es mantener o incrementar la curiosidad e interés del alumno por la asignatura, generando documentos dinámicos que disminuyan la fatiga que representa tener que acceder siempre al mismo contenido y de la misma forma.

Cada prueba es una novedad, puesto que se permite cambiar tanto el orden de los ejercicios como el orden de las preguntas y respuestas dentro del mismo ejercicio. Asímismo, puede generar varias versiones del mismo examen con el mismo contenido, para disminuir la posibilidad de copia entre alumnos durante la prueba, en caso de que ésta se dedique a realizar una evaluación real. Esta facilidad impide que el alumno se aprenda las

respuestas de memoria sin que las comprenda, como ocurre en juegos culturales como el Trivial.

Por otra parte, el MGA presenta una uniformidad de interfaz que facilita su uso, entre los diferentes temas y entre diferentes asignaturas. Al separar el motor de autoevaluación del conocimiento se consigue que cada profesor pueda suministrar su propia base de datos junto con los contenidos multimedia asociados independientemente del programa, reduciendo por tanto la cantidad de información que circula a través de Internet o la cantidad de CDs que se requieren para la asignatura. La variación se encuentra en la teoría asociada a la práctica y en el contenido de la prueba. El grado de variación de las pruebas generadas dependerá de la cantidad de los conocimientos introducidos por el profesor.

La actualización de los contenidos se realiza fácilmente tan sólo con sustituir el fichero tipo *mdb* de la base de datos con la nueva versión e incorporar los nuevos recursos multimedia en los directorios correspondientes para que los hiperenlaces existentes puedan referenciarlos correctamente.

2.2. MGA simplificado

La versión simplificada del MGA consiste en un interfaz de usuario muy parecido a la versión integral, pero a la que se le ha eliminado toda la funcionalidad de edición o alteración del contenido de la base de datos. Esta es la versión que normalmente utiliza el alumno, ya que su objetivo no es alterar el contenido teórico-práctico de la asignatura, sino poder demandar del programa pruebas evaluatorias a realizar o corregidas con el fin de contrastar las respuestas dadas por el alumno respecto de las ofrecidas por el programa.

Gracias a la utilización de esta herramienta, se permite incrementar la productividad del profesor debido a la automatización de determinadas tareas docentes como son la resolución de dudas simples, el entrenamiento del alumno en problemas sencillos y la validación de conocimientos tanto en un ámbito doméstico (autoevaluación) como en el centro (evaluación).

De esta forma, el alumno realiza el estudio personalizado y adaptado y el grado de desarrollo de su capacidad cognitiva mediante la repetición de aquellas partes que menos domina, y de acuerdo a su ritmo de aprendizaje. Así mismo, se permite un seguimiento (estadístico) de su progreso tanto por parte del profesor (estadísticas en el servidor) como por él mismo (en su base de datos local).

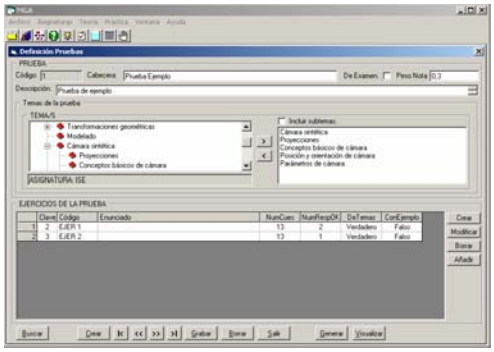


Figura 3. Detalle de la versión simplificada en la fase de generación de pruebas

La realimentación que se produce al interaccionar el alumno con el programa permite incrementar la seguridad con la que el alumno se enfrentará posteriormente a un examen, sabiendo de antemano cuál es el grado de profundidad y dominio del conocimiento del que se le va a examinar. De esta forma, el alumno reduce la ansiedad y el nerviosismo con el que se realiza el examen, mejorando la nota media. Por otro lado, se reduce el porcentaje de suspensos frente a los presentados, ya que la cantidad de presentados inconscientes de su nivel se reduce y aquellos que no se atrevían a presentarse porque dudaban de su nivel, ahora se animan a hacerlo.

Al utilizar el motor de bases de datos de MS-Access permite el acceso concurrente multiusuario en red para facilitar el acceso a las consultas o a la realización de exámenes en red. Estos exámenes pueden ser tanto impresos a la manera tradicional como en formato electrónico, con evaluación en tiempo real. Por su parte, los alumnos pueden plantear actividades, pero no alterar el contenido de la base de conocimiento.

2.3. WebMGA: La versión web del MGA

La tendencia creciente de uso de la web con fines educativos, motivó la consideración de hacer disponible para el alumnado todo el material generado con el MGA a través de web, dando

origen a la versión web del mismo, el WebMGA. Con esta herramienta, el alumno, conectado a la web, trabaja contra servidor y puede ir realizando ejercicios propuestos por el docente o generar sus propios ejercicios (basándose en la información ya incluida en la base de datos), implicándose en su propio proceso de aprendizaje.

Figura 4. Detalle del navegador durante la fase de diseño de prueba en el WebMGA

La posibilidad de que el alumno trabaje a través de la web proporciona dos importantes ventajas: la posibilidad de acceder a una información constantemente actualizada, prácticamente en tiempo real, y de establecer una comunicación fluida entre el docente y el alumno; por ejemplo, vía correo electrónico o constituyendo foros sobre la asignatura. Igualmente son importantes las opciones que ofrece el WebMGA de evaluación automática contra servidor y de seguimiento estadístico de su progreso.

Prácticamente este versión ofrece las mismas posibilidades que la versión MGA reducida pero a través de un interfaz web que ataca a un servidor remoto vía Internet. Esta opción es especialmente interesante para la realización de consultas de última hora o realización de pruebas actualizadas. Sin embargo, para alumnos que no tienen acceso a Internet y estudian en zonas remotas, que no disponen de tarifas planas o de muy escasos recursos, es más recomendable la navegación en modo local fuera de línea vía MGA. La velocidad de acceso, respuesta y economía es mayor. Este punto es especialmente importante en la formación de colectivos muy desfavorecidos o de países subdesarrollados,...

2.4. SAM: Integración MGA-WebMGA

El Sistema de Autoevaluación Multimedia (SAM) es un sistema de trabajo que fusiona los objetivos perseguidos por las herramientas MGA y WebMGA, constituyendo una nueva propuesta en la forma de desarrollar el proceso de (auto)aprendizaje. La integración de ambas herramientas permite que un docente defina los contenidos de la asignatura a impartir, distribuyéndolos en temas, haciendo referencia a páginas web de teoría de su asignatura y diseñando materiales de prácticas. Todo ello se incluye en una base de datos, la cual será accedida desde la web para posibilitar el trabajo de los alumnos.

SAM puede ser implantado en cualquier entorno y nivel educativo siempre y cuando se disponga de una Intranet, de un servidor IIS que soporte tecnología ASP y opcionalmente de una conexión a Internet. La base de datos puede estar en el mismo servidor web o, por medidas de seguridad, en un servidor de BD con acceso restringido y que sólo satisfaga servicios de acceso desde determinadas máquinas. La misma base de datos y los mismos recursos multimedia pueden ser atacados concurrentemente por diferentes clientes MGA-alumno y MGA-profesor desde un aula informática, así como por el servidor web del aula que está atendiendo a diferentes peticiones a través de Internet.

Hay que señalar que la modificación de los elementos básicos de la base de datos de conocimientos, como son temas, preguntas, respuestas o su asociación en cuestiones no es posible vía web por motivos de seguridad. Lo único que queda registrado en la base de datos cuando un alumno trabaja desde web son las selecciones que hace para generarse sus propias pruebas o las respuestas que da al realizar alguna prueba. De esta manera, sólo se permite al docente (o grupo de docentes), en modo local, incluir o modificar los contenidos básicos incluidos en la base de datos.

2.5. Situación actual

El proceso de aplicación del SAM se ha puesto en marcha en las siguientes asignaturas impartidas en la Escuela Técnica Superior de Informática Aplicada de la Universidad Politécnica de

Valencia: Inglés Técnico, Alemán I, Informática Gráfica y Estructura y Tecnología de Computadores I.

Este proceso se compone fundamentalmente de dos fases. En una primera fase el docente adapta su asignatura a la base de datos de conocimiento, distribuyendo sus contenidos teóricos en un conjunto definido de temas y subtemas, generando material de práctica y realizando las pertinentes asociaciones entre teoría y práctica. En una segunda fase, que se está llevando a cabo durante este curso, se pone en marcha la utilización por parte de los alumnos de la base de datos confeccionada vía web, teniendo en cuenta todas las posibles aportaciones e ideas que proporcionen los usuarios para incorporar las modificaciones que se crean más convenientes.

Hay que indicar que, aunque actualmente sólo sea operativo el trabajo con pruebas de tipo test, la base de datos ha sido diseñada para soportar otros tipos de ejercicios que se desarrollarán próximamente. Estos ejercicios son los de rellenado en huecos, asociación de parejas, respuesta múltiple acotada, etc.

3. Conclusiones

SAM es un sistema de autoevaluación multimedia que permite al alumno ser protagonista de su propio proceso de aprendizaje, adaptándolo a sus necesidades particulares y obteniendo continuamente información sobre su progreso, permitiéndole continuamente conocer su situación y detectar nuevas necesidades ante las que reorientar su aprendizaje.

Este tipo de evaluación implica más intensamente al alumno en su proceso de aprendizaje al hacerle corresponsable de su evolución.

El éxito en la utilización de esta tecnología por parte del alumno depende en gran medida del trabajo previo que haya realizado el docente con el MGA. El trabajo con SAM puede ser inicialmente poco gratificante y tedioso para el profesor, ya que al coste de aprendizaje de la herramienta se le une la trascripción del material tradicional a formato electrónico y su organización correspondiente. De igual modo, mientras no se haya introducido una masa crítica mínima de contenidos en la BD, ésta ofrece pocas posibilidades al alumno y éstas se agotan enseguida, disuadiendo a los alumnos de utilizarla.

Por otro lado, es importante indicar que el desarrollo de la base de datos puede ser soportada entre varios docentes, con lo que el esfuerzo se distribuye. Por otro lado, esto exige una mayor esfuerzo de coordinación entre ellos. También hay que señalar la posibilidad de reutilizar información de un año para otro, de manera que progresivamente se vaya enriqueciendo los contenidos y las opciones de práctica, pudiendo reutilizar materiales incluso entre diferentes asignaturas afines.

4. Agradecimientos

Este trabajo esta soportado por el proyecto UPV-PPI5-20000595

Referencias

[1] Odlyzko, Andrew. *The slow evolution of electronic publishing. Electronic Publishing '97: New Models and Opportunities*, A. J. Meadows and F. Rowland, eds., ICCC Press, 1997, pp. 4-18.

[2] Hawkins, Brian L. *Information Access in the Digital Era: Challenges and a call for collaboration*. Educause review, Sept/Oct 2001

[3] Pérez, Rubén; Sastre, Carlos; Cruz, Pedro. Incorporación de nuevas tecnologías en la enseñanza. JENUI 2001 ISBN: 84-7632-657-2, pág. 390-395

[4] Rebollo, M.; Carrascosa, C.; Julian, V. Botti, V. CALIOPE : Agentes para Entornos de Aprendizaje Activo. I Jornadas de innovación educativa Metodologías activas y evaluación Valencia, 24 al 27 de septiembre de 2001. ISBN 84-9705-187-4

[5] Aguaded Gómez, J. Ignacio; Cabero Almenara, Julio. *Educar en red. Internet como recurso para la educación*. Ediciones Aljibe. Málaga, 2002.

[6] Mollá, R.;Antelm, J. M. *MGA. Motor Generalista de Autoevaluación. 2000*. Proceedings Enseñanza vía Internet/Web de la Ingeniería de Sistemas y Automática *EIWISA mayo 2000*. ISBN: 84-7721-963-X

Evaluación continuada a un coste razonable

Miguel Valero-Garcia, Luis M. Díaz de Cerio
Dept. d'Arquitectura de Computadors
Universitat Politécnica de Catalunya
Avda. del Canal Olimpic, s/n
08860 Castelldefels (Barcelona)
e-mail: {miguel, ldiaz}@ac.upc.es

Resumen

Allá donde se ha intentado implantar de forma generalizada, la evaluación continuada ha acabado por provocar frustración entre el profesorado, por diferentes razones, una de las cuales es el importante incremento de carga de trabajo debida a la corrección de los ejercicios de los alumnos. En nuestra opinión, se suelen mezclar dos aspectos que es importante separar: (a) Evaluación Formativa: el suministro frecuente de información al alumno, sobre lo bien o lo mal que va en el curso (cuestión fundamental para el aprendizaje) y (b) Evaluación Sumativa: la asignación de calificaciones a las tareas, ejercicios, etc., a lo largo del curso (cuestión secundaria respecto al aprendizaje). Cuando se separan estos aspectos, es posible plantear sistemas de evaluación continuada que satisfacen el requerimiento fundamental: mantener informado al alumno, a un coste razonable para el profesor. En este trabajo se propone el uso de la autoevaluación y la co-evaluación como sistemas de evaluación formativa a coste razonable, y se analizan las implicaciones de diferentes esquemas.

1. Introducción: La promesa de la evaluación continuada (la teoría y la realidad)

Está ampliamente aceptado que un ingrediente de la docencia de calidad es un sistema de retroalimentación que permita al alumno mantenerse puntualmente informado sobre su progreso (o falta de progreso) en el plan de aprendizaje [1]. Después de resolver un ejercicio, por ejemplo, un alumno necesita saber si lo ha hecho bien o mal, y por qué. Y necesita saberlo pronto, para poder tomar rápidamente las acciones correctoras necesarias. En otras palabras, se trata de evitar que el alumno descubra el día del examen final (o peor aún, al conocer las notas del examen final) que ha entendido bien poca cosa del material del curso.

A pesar de esta obviedad, no es fácil encontrar asignaturas con un buen sistema de evaluación continuada, especialmente cuando el número de alumnos por profesor es elevado. Los alumnos se quejan con frecuencia de sentirse perdidos, sin referencias, y de verse sorprendidos constantemente en los exámenes, cuando se pone en evidencia que su rendimiento real está muy por debajo de su predicción.

La evaluación continuada ha sido un estandarte en materia de mejora de la docencia en muchos sitios. Así por ejemplo, fue uno de los ingredientes principales de la reforma de los planes de estudio en la Universidad Politécnica de Cataluña (UPC), a inicios de los 90.

En muchos casos los sistemas de evaluación continuada han ido simplificándose hasta convertirse en poco más de un examen parcial a mitad del cuatrimestre, y un último examen al final, con lo que el objetivo básico de la evaluación continuada (mantener bien informado al alumno sobre el estado de su aprendizaje) no se cumple.

La obligación de establecer un mecanismo de evaluación continuada en todas las asignaturas compensó, a los ojos de los alumnos, la pérdida de la convocatoria de septiembre, que se produjo como consecuencia de la adopción de un sistema cuatrimestral puro.

Todas las ideas presentadas en este artículo para la mejora de la evaluación continuada han sido o están siendo puestas en práctica en la asignatura de Laboratorio de Progamación. Sin embargo, creemos que una orientación teórica

sobre estas ideas es más útil, en el sentido de que puede aplicarse mejor sobre diferentes asignaturas que una determinada propuesta para una asignatura concreta.

En la siguiente sección separamos los dos conceptos clave que forman la evaluación continuada: información continuada (evaluación formativa) y calificación continuada (evaluación sumativa). En la sección 3 proponemos dos técnicas para realizar la evaluación formativa con bajo coste y por tanto disminuyendo en global el coste de la evaluación continuada. En la sección 4 veremos como corresponden las técnicas propuestas respecto a los niveles de competencia según la conocida taxonomía de Bloom. En las secciones 5 y 6 presentamos diferentes metodologías para llevar a cabo las técnicas propuestas y finalmente, en la sección 7, presentamos las conclusiones.

2. Información continuada versus calificación continuada (separemos las cosas)

En esta sección consideramos algunas cuestiones teóricas sobre la evaluación y sus características. Estas consideraciones son la base en la que se apoyan las propuestas que se realizan en este artículo.

Cuando se trata de la evaluación del nivel de aprendizaje de nuestros alumnos, solemos distinguir entre dos tipos, según el uso que se hace de la información:

- Evaluación formativa: La información se usa para guiar y mejorar los procesos de enseñanza y aprendizaje
- Evaluación sumativa: La información se usa para determinar la calificación que acredita el nivel de aprendizaje conseguido por el alumno.

Por otra parte, un sistema de evaluación (tanto sumativa como formativa) puede tener (entre otros) los siguientes atributos:

- Precisión y fiabilidad: El resultado de la evaluación es el mismo con independencia de la persona que realice la evaluación, o del momento en que se realice la evaluación (fiabilidad), y ese resultado tiene poco margen de error (preciso).
- Prontitud: El resultado de la evaluación está en manos del alumno y del profesor lo antes posible, después de haber realizado el acto a evaluar.

Es interesante ahora analizar la relación que existe entre los tipos de evaluación y los atributos de la evaluación. En particular, puede afirmarse que:

La evaluación sumativa debe ser precisa y fiable, puesto que está en juego el expediente del alumno, que es un documento oficial que puede tener mucha trascendencia a la hora de buscar trabajo, optar a becas, etc. Sin embargo, no es especialmente crítico que los resultados de la evaluación sumativa estén disponibles con prontitud (y si es crítico se debe más bien a los requisitos del sistema administrativo, y del calendario académico, que debe cerrar procesos en una fecha determinada para dar inicio al siguiente cuatrimestre).

El atributo más importante de la evaluación formativa es la prontitud. En este caso, también es deseable que el sistema sea preciso y fiable, pero estos atributos son secundarios. En otras palabras, cuando un alumno hace, por ejemplo, un ejercicio, lo importante es que el alumno sepa pronto si las decisiones importantes han sido acertadas. En cambio, no es excesivamente importante saber si el resultado merece una nota de 6.5 o una de 7 (sí que lo es, insistimos, en el caso de la evaluación sumativa).

Finalmente, es importante observar que lo que hace que un sistema de evaluación sea costoso en términos de tiempo de profesor es que el sistema sea preciso y fiable. Efectivamente, el tener que determinar una calificación, típicamente con una precisión de 0.5 puntos, implica un tiempo de análisis del trabajo realizado por cada alumno, que puede llegar a suponer una carga total de trabajo importante cuando hay muchos alumnos por profesor, y muchos actos de evaluación a lo largo del curso.

Lo importante ahora es observar que, en el caso de la evaluación formativa (la que nos interesa en el contexto de este trabajo), es aceptable sacrificar precisión y fiabilidad (y por tanto, reducir coste para el profesor), siempre y cuando se mantenga el requisito de prontitud. Este es el principio subyacente en las propuestas que se desarrollan en las secciones siguientes.

En resumen, cuando se habla de evaluación continuada, conviene distinguir entre información continuada y calificación continuada. La información continuada es imprescindible para que el proceso de enseñanza-aprendizaje tenga salud. La calificación continuada es costosa y no es en absoluto necesaria. Incluso en muchos casos, y especialmente en el caso de la organización cuatrimestral, no es razonable calificar al alumno antes de la última fase del curso, cuando ya ha tenido la oportunidad de asimilar y ejercitar los conceptos básicos. Éste es, por ejemplo, el caso de un primer curso de enseñanza de la programación, en la que todo lo que puede calificarse en la primera mitad de un cuatrimestre tiene poca relevancia comparado con la fase final del curso, que es cuando el alumno está en condiciones de resolver ejercicios de una mínima entidad.

3. Autoevaluación y co-evaluación: información con prontitud y a bajo coste

Las estrategias de autoevaluación y co-evaluación pueden usarse como base para la organización de un sistema de evaluación que proporcione información con prontitud, aunque para ello tengamos que renunciar a la precisión y a la fiabilidad. La idea básica de estas estrategias es que los alumnos pueden ser colaboradores del profesor, en este caso, en las tareas de evaluación; ya que los profesores, por lo general, no disponen de ayudantes para realizar la labor evaluadora. Si los alumnos han de colaborar en la evaluación, las únicas dos opciones de las que disponemos son: evaluarse a sí mismos (autoevaluación) o evaluar a otros compañeros (co-evaluación). En concreto, cuando se usa la estrategia de la autoevaluación, es el propio alumno el que determina en qué medida su trabajo está bien o mal siguiendo las instrucciones del profesor. En el caso de la co-evaluación, cada alumno evalúa el trabajo de uno o varios de sus compañeros, también siguiendo las instrucciones del profesor.

La autoevaluación y la co-evaluación proporcionan información con prontitud, puesto que si el profesor tiene preparadas las instrucciones con antelación, los alumnos pueden realizar la evaluación inmediatamente después de realizar el trabajo y obtener las conclusiones rápidamente. Lógicamente, la evaluación será menos precisa y fiable que si la hubiese realizado el profesor, puesto que el profesional es él, y no los alumnos.

Además de resolver el problema básico que nos concierne en este trabajo, la autoevaluación y la co-evaluación tienen otras virtudes que conviene tener bien presentes [2]. En concreto, en el caso de la autoevaluación:

- Los alumnos van interiorizando los criterios de corrección que el profesor hace explícitos a través de las instrucciones para la autoevaluación. Esto permite a los alumnos ajustar cada vez más sus respuestas a lo que el profesor espera.
- Los alumnos desarrollan el hábito de la reflexión, y la identificación de los propios errores, cuestión fundamental cuando se trata de formar personas con capacidad para aprender de forma autónoma.

En el caso de la co-evaluación, además de las virtudes anteriores, podemos mencionar también las siguientes:

- Los alumnos se esfuerzan más, impulsados por la motivación de quedar bien ante los ojos de sus compañeros (este tipo de motivación suele ser superior a la de quedar bien ante los ojos del profesor).
- Los alumnos desarrollan el hábito de criticar de forma constructiva el trabajo realizado por compañeros con los que van a tener que continuar colaborando. Ésta es también una habilidad fundamental que se echa en falta con frecuencia en el mundo profesional.

Es posible que al plantear un sistema de evaluación continuada basado en la autoevaluación y en la co-evaluación los alumnos manifiesten unas ciertas reticencias, e incluso puedan acusar al profesor de cargarles a ellos con un trabajo y una responsabilidad que no les corresponde. En estos casos, conviene admitir que, efectivamente, uno de los objetivos de estas estrategias es reducir el tiempo que el profesor dedica a la evaluación para poder dedicar ese tiempo a otras tareas igualmente importante para el aprendizaje. Pero además, conviene tener a mano todo el repertorio de virtudes de estas estrategias que, bien planteadas, proyectarán ante los alumnos la idea de que el profesor ha pensado en lo que es

bueno para sus alumnos a la hora de establecer los elementos de su programa.

4. La evaluación y los niveles de competencia

Otro elemento importante a la hora de establecer un sistema de evaluación continuada es el análisis del nivel de competencia de los objetivos formativos que se van a evaluar. La taxonomía de referencia para el estudio del nivel de competencia de los objetivos formativos es la taxonomía de Bloom [3], según la cual un objetivo formativo pertenece a uno de los siguientes niveles, en orden creciente de competencia: *conocimiento, comprensión, aplicación, análisis, síntesis o evaluación.*

En [4] se hizo un trabajo de adaptación de la taxonomía de Bloom al ámbito de la enseñanza de la ingeniería, y se plantearon algunas reflexiones sobre el equilibrio que debe existir (y el desequilibrio que muchas veces existe) entre nivel de competencia, métodos docentes y métodos de evaluación.

En el contexto de este trabajo, en el que lo que nos interesa es la evaluación, proponemos una taxonomía simplificada, basada en tres niveles de competencia. A continuación se describen cada uno de estos niveles y se da un ejemplo de ejercicio (o tipo de ejercicio) de cada nivel, perteneciente al ámbito de la enseñanza de la programación de ordenadores.

- *Conocimiento:* Requiere que el alumno recuerde datos, hechos, información que le ha sido suministrada con anterioridad.
  1. Describe la sintaxis y la semántica del bucle "for" de lenguaje C
  2. Indica cuál es el rango de representación de los enteros de tipo short, en lenguaje C
- *Comprensión*: Requiere que el alumno aplique un cierto procedimiento sistemático conocido (una "receta") en un caso particular.
  3. Determinar el valor final de una variable después de ejecutar una secuencia de sentencias de lenguaje C
  4. Escribir el código en lenguaje C para ordenar, mediante el método de la burbuja, un vector de caracteres.
- *Aplicación*: Requiere que el alumno elija, de entre las "recetas" que ha comprendido previamente, la más apropiada para resolver un determinado problema.
  5. Determinar cuál es la estructura de datos más adecuada para resolver un problema
  6. Determinar qué operaciones de un determinado programa pueden ser realizadas mediante procedimientos y funciones de librerías ya existentes.

En la taxonomía simplificada que se propone, los niveles de conocimiento y comprensión se corresponden directamente con los dos primeros niveles de la taxonomía de Bloom. Sin embargo, el nivel de aplicación de la propuesta corresponde a una fusión de los niveles de aplicación y superiores de la taxonomía de Bloom. Consideramos que esta simplificación es válida en el contexto de la enseñanza en asignaturas iniciales de primer ciclo, en las que rara vez se supera el nivel de aplicación en la taxonomía de Bloom.

El análisis del nivel de competencia de los objetivos es una herramienta muy útil para la planificación de una asignatura, y en particular, para la elección de los métodos docentes y de evaluación. Puesto que en este trabajo estamos interesados en métodos de autoevaluación y coevaluación, veamos ahora las implicaciones del nivel de competencia en los métodos de evaluación.

Cuando el ejercicio que debe realizar el alumno es de nivel de conocimiento o de comprensión, la respuesta es única o admite muy pocas variaciones. Además, en el caso de comprensión, los resultados intermedios en el proceso de aplicar la "receta" también son únicos. Por ejemplo, en el caso del ejercicio 4, no hay en realidad muchas formas distintas de codificar el algoritmo de la burbuja para ordenar un vector de caracteres. En estas condiciones, la evaluación es muy sencilla: la solución del alumno debe asemejarse a la solución "oficial", y cualquier diferencia es, en potencia, un error del alumno. Este principio será la base para los esquemas de autoevaluación que se proponen en la siguiente sección.

Cuando el ejercicio es de nivel de aplicación, diferentes alumnos pueden dar respuestas distintas, todas ellas válidas. Ahora no tiene sentido usar una solución "oficial" como base para la evaluación (aunque sí puede ser útil

Para cada uno de los ejercicios siguientes, debes preparar un informe de autoevaluación que se compone de dos partes:

a) *Solución al ejercicio.*

b) *Errores que nunca más volveré a cometer.*

Para escribir la parte (b) debes usar la solución oficial del ejercicio. Puesto que cada uno de los ejercicios tiene una solución única, cualquier diferencia entre tu solución y la solución oficial puede ser:

- Un error en la solución oficial (cosa poco probable).
- Un error tuyo.
- Una diferencia admisible entre las soluciones.

Compara tu solución con la oficial, identifica todas las diferencias y clasifica cada una de ellas según los tipos anteriores. Escribe tus conclusiones en la sección "Errores que nunca más volveré a cometer", de tu informe de autoevaluación.

Figura 1: Instrucciones para la autoevaluación

como ejemplo). Lo que se necesita es explicitar los atributos que deben tener las soluciones correctas, de manera que la evaluación consiste en identificar los atributos propuestos en las respuestas a evaluar. Por ejemplo, en el caso del ejercicio 5, uno de los atributos de una solución correcta es que la estructura de datos propuesta ocupe un espacio razonable de memoria, o en otras palabras, que no haya soluciones alternativas que ocupen mucho menos espacio de memoria. Este principio será la base para el sistema de co-evaluación que se propone en la sección 6.

5. Autoevaluación y el nivel de comprensión

En esta sección se propone un esquema de autoevaluación que puede usarse cuando los objetivos formativos a evaluar son de nivel de conocimiento o comprensión.

La idea básica es partir de una solución "oficial" al ejercicio, y basar la autoevaluación en la comparación de la propia solución con la solución "oficial". Puesto que, dada la naturaleza del ejercicio, no hay muchas variaciones válidas posibles en la respuesta, cualquier diferencia entre la respuesta del alumno y la solución "oficial" es un error potencial. Por tanto, la autoevaluación consiste en identificar las diferencias, reconocer aquellas que corresponden a errores, y justificar aquellas que son variaciones admisibles. Las instrucciones para los alumnos podrían ser las mostradas en la figura 1. Es muy importante que el alumno entienda la importancia de hacer con rigor la comparación de la propia solución con la "oficial". Al realizar la autoevaluación por primera vez, es habitual que el alumno escriba, en la sección de *Errores que nunca más volveré a cometer*, cosas como: "Mi solución es claramente distinta de la oficial, pero creo que también es correcta". En un caso así, lo adecuado es devolver la autoevaluación al alumno y exigirle que realice correctamente la identificación y clasificación de las diferencias.

Los aspectos de logística para organizar el sistema de autoevaluación también son muy importantes. Se proponen, a continuación, tres posibilidades.

1. Los alumnos no entregan las autoevaluaciones al profesor. Es un material de uso propio. Si bien esta estrategia cumple con el propósito perseguido (informar al alumno con prontitud, con una carga de trabajo baja para el profesor), no es recomendable porque:
   - El profesor no tiene ocasión de intervenir para asegurarse que los alumnos hacen bien la autoevaluación (y las primeras veces, seguro que no la harán bien).
   - El profesor no tiene información de cómo van los alumnos, y no puede realizar accio-

nes correctoras, ni a nivel individual ni colectivo.

2. El profesor pide, de vez en cuando, los informes de autoevaluación. De esta forma, puede asegurarse de que los alumnos están haciendo bien la tarea, y tiene información sobre las dificultades individuales y colectivas. Idealmente, el profesor explica a sus alumnos, al inicio del curso, que uno de sus objetivos para el curso es que aprendan a identificar sus propios errores, de manera que calificará los informes de autoevaluación (en función de si se han identificado bien los errores), y usará esa calificación para obtener parte de la nota final del alumno. Naturalmente, esta opción implica algo más de trabajo para el profesor. La opción admite dos variantes:

- De vez en cuando, el profesor recoge el informe de autoevaluación de todos los alumnos, correspondiente a un ejercicio.
- De vez en cuando, el profesor recoge todos los informes de autoevaluación de un determinado alumno o grupo de alumnos.

La primera opción tiene la ventaja de que el profesor puede tener una idea clara de las dificultades generales del grupo con un determinado tipo de ejercicios. Tiene el inconveniente de que se trabaja con muchas hojas sueltas, cosa que puede resultar incómoda. Con la segunda opción, el alumno entrega al profesor una carpeta bien organizada con sus informes de autoevaluación (y, posiblemente, otros materiales del curso). Además, al analizar esta carpeta, el profesor puede obtener más fácilmente una visión nítida de la evolución del alumno (hasta qué punto su rendimiento va mejorando). Otra ventaja adicional es que el profesor puede pedir la carpeta con más frecuencias a aquellos alumnos que necesitan más ayuda.

3. Además de calificar la calidad de la autoevaluación (como en la opción anterior) el profesor usa la calificación que se asignan los alumnos para obtener la nota final de la asignatura. En este caso, en la guía para la autoevaluación, además de la solución oficial deben aparecer los criterios precisos para asignar una calificación a la solución. La ventaja de esta opción es que el alumno tiene una mayor responsabilidad en el proceso, y tiende a tomarse más en serio la autoevaluación, al ver que hay un impacto directo mayor en la calificación final. Sin embargo, también hay inconvenientes importantes: el alumno puede centrarse más en la nota que en la identificación de errores, la elaboración de las instrucciones para la autoevaluación se complica[1], y el profesor debe, probablemente, dedicar más tiempo a analizar los informes de autoevaluación. Por otra parte, no es una alternativa apropiada cuando la organización y contenidos del curso no recomiendan el asignar calificaciones hasta la fase final en la que los alumnos están en condiciones óptimas para ser calificados. Cualquiera que sea el esquema adoptado, es muy importante tener en cuenta que los alumnos necesitan repetir el proceso con una cierta frecuencia para aprender a autoevaluarse, y a sacar provecho de la actividad. Las experiencias puntuales (por ejemplo, una autoevaluación, un día del curso) suelen generar más frustración y desorientación que otra cosa, y sólo son recomendables si se trata de probar la mecánica, de cara a una implementación generalizada en el futuro.

6. Co-evaluación y el nivel de aplicación

Cuando se trata ejercicios de nivel de aplicación, una solución "oficial" no es suficiente para realizar la evaluación, puesto que el ejercicio admite soluciones distintas y todas correctas. Lo importante es explicitar las características que debe tener una determinada solución para que pueda ser considerada correcta.

Una forma de presentar a los alumnos las características de las soluciones correctas es usar rúbricas [5] como las que se muestra en la figura 2. En este caso, se trata de evaluar el código para resolver un determinado problema de programación. En la columna de la izquierda se identifican diferentes criterios de evaluación del código. En las columnas de la derecha se identifican, para cada uno de los criterios de evalua-

[1] Muchos profesores insisten en lo complicado que puede resultar el explicitar los criterios para que el alumno pueda calcular una calificación precisa. Sin embargo, eso no es más difícil que explicitar los criterios de corrección que debe usar un grupo de profesores que se reparten la corrección de un ejercicio del examen final, cosa que hacemos de forma habitual.

Código			
	Nivel de calidad		
Criterio	**3** **Notable**	**2** **Suficiente**	**1** **Insuficiente**
Correcto	La aplicación funciona bien en todos los casos. No he encontrado ningún fallo.	Hay (como máximo) un par de situaciones en las que el programa no ha funcionado bien.	La aplicación falla constantemente.
Robusto	La aplicación resiste sin bloquearse todos los errores típicos que puede cometer un usuario "poco hábil". No he conseguido que se cuelgue.	Es razonablemente robusto. No es fácil que se queda colgado, pero en uno o dos casos se bloqueó.	La aplicación no es robusta en absoluto. Se queda colgada con frecuencia ante errores típicos del usuario al entrar datos.
Amigable	El usuario no tiene ninguna duda, en ningún momento, sobre cómo interactuar con la aplicación, qué datos debe entrar y cómo, y cómo interpretar los resultados y mensajes de la aplicación.	Los mensajes e información que da la aplicación son suficientes para trabajar bien. Sin embargo, el alguna ocasión he tenido algunas dudas sobre lo que hay que hacer o cómo hay que hacerlo.	El usuario tiene dudas constantes sobre lo que le está pidiendo la aplicación, y es difícil interpretar los resultados y mensajes en pantalla.
Comparado con el nuestro	Este código es mejor.	Este código es similar.	Este código es peor.

Figura 2: Ejemplo de rúbrica para la evaluación de un código

ción, las características que debe reunir una determinada solución para que pueda ser considerada *notable*, *suficiente* o *insuficiente* (en este ejemplo, se consideran sólo tres niveles de calidad). La tarea del evaluador es identificar en la solución que tiene que evaluar, las características señaladas en la rúbrica, y determinar el nivel de calidad para cada uno de los criterios de evaluación. El informe de evaluación se completa con los argumentos en los que el evaluador se basa para tomar la decisión, como por ejemplo:

a) El código falla en los casos de prueba 1 y 4.
b) El mensaje que avisa al usuario para que introduzca datos no es claro: tengo dudas sobre el formato que debo usar para entrar los datos.
c) El código está muy mal indentado. Tengo muchas dificultades para identificar los bloques de código.

Sin duda, la evaluación del nivel de aplicación es más difícil de objetivizar que la evaluación del nivel de conocimiento o comprensión. En el caso, por ejemplo, del argumento (b) de la lista anterior, dos personas distintas pueden tener opiniones diferentes sobre la claridad del mensaje que el programa ofrece al usuario. Además, con toda probabilidad, el autor del código opinará que su mensaje es claro. Por estas razones, la evaluación del nivel de aplicación no acaba de combinarse bien con la estrategia de autoevaluación (cuando se trata de cosas opinables, no somos buenos jueces de nosotros mismos).

En cambio, la estrategia de la co-evaluación se adapta mejor a la naturaleza de este tipo de evaluación. Un alumno puede juzgar de forma más neutral el resultado del trabajo de otros compañeros, naturalmente con la ayuda de una rúbrica bien elaborada. Además, su propia solución es un punto de referencia importante para valorar los méritos y deméritos de las soluciones de los compañeros. Esa es la razón de que, en la rúbrica que se muestra como ejemplo, la última fila haga referencia a una comparación entre la solución propia y la que se está evaluando.

Los aspectos logísticos de la co-evaluación son más complicados que los de la autoevaluación, simplemente porque los resultados de los trabajos de los alumnos deben ser recogidos y redistribuidos para ser evaluados por otros. Las soluciones a este problema dependerán mucho de las circunstancias de cada caso, por lo que poco más puede decirse aquí al respecto con carácter general.

Los comentarios que se han hecho en el apartado anterior en cuando al uso que hace el

profesor de los resultados de la autoevaluación también son aplicables en el caso de la coevaluación. En todo caso, conviene ser consciente de que los alumnos no suelen mostrarse favorables a asignar a los compañeros calificaciones que puedan afectar a la nota final. Por tanto, si se opta por esta alternativa, es importante tener a punto el argumento para convencerles de que la habilidad de emitir críticas constructivas, y juzgar el trabajo de los compañeros es importante para le ejercicio profesional.

7. Conclusión

Las técnicas que se plantean de forma teórica en este trabajo se están poniendo en práctica en la asignatura Laboratorio de Programación (LP), perteneciente a la Ingeniería Técnica en Telecomunicaciones, de la Escuela Politécnica Superior de Castelldefels (UPC). En realidad, han sido las experiencias en esta asignatura las que nos han ayudado a poner en orden nuestras ideas, en la forma presentada aquí.

La asignatura ofrece un escenario ideal. En la primera parte, los objetivos formativos son de nivel de comprensión. Los alumnos deben escribir códigos en Visual C++ que implementan algoritmos conocidos a casos particulares. En esta primera parte se utiliza la técnica de la autoevaluación, con la variante en la que la nota que se adjudican los alumnos, después de darse por válida, se usa en el cálculo de la nota final.

En la segunda parte de la asignatura los objetivos son de nivel de aplicación. Los alumnos deben tomar decisiones relativas a las estructuras de datos y algoritmos que hay que usar para resolver un determinado problema. Las actividades se realizan en modo proyecto, de forma que los alumnos, trabajando en grupo, deben resolver un problema. La técnica de la coevaluación basada en rúbricas se usa para que cada grupo evalúe el trabajo que han realizado otros.

Todas las ideas presentadas en este artículo han sido o están siendo puestas en práctica en la asignatura de LP. Sin embargo, no es objetivo de este trabajo el dar los detalles de implementación, ni analizar los resultados obtenidos para una asignatura concreta. Creemos que la orientación teórica que hemos dado al artículo es más útil, en el sentido de que puede aplicarse mejor sobre diferentes asignaturas que una determinada propuesta para una asignatura concreta. En todo caso, si pueden apuntarse algunas conclusiones, obtenidas a partir de encuestas de opinión (especialmente sobre la autoevaluación, con la que llevamos más tiempo experimentando):

- Los alumnos perciben la autoevaluación como algo positivo. En particular, creen que les permite mantenerse puntualmente informados.
- Los alumnos aprenden pronto la mecánica. El porcentaje de autoevaluaciones que deben ser corregidas es pequeño.
- El tiempo de dedicación de los profesores a la supervisión es asumible. En el caso de la asignatura LP, estamos hablando de 1 hora a la semana, para un grupo de 40 alumnos.
- El hecho de que la nota de autoevaluación cuente para la nota final hace que los alumnos pongan mucho énfasis en el mero cálculo de la nota, y menos en la identificación de los errores cometidos. Éste es, sin duda, el aspecto en el que debemos poner más énfasis de cara a mejorar el funcionamiento.

Referencias

[1] A. W. Chickering y Z. F. Gamson, Seven Principles for Good Practice in Undergraduate Education, http://www.hcc.hawaii.edu/intranet/committees/FacDevCom/guidebk/teachtip/7princip.htm

[2] A.W. Bangert, Peer Assessment: A Win-Win Instructional Strategy for Both Students and Teachers, *J. Cooperation & Collaboration in College Teaching*, Vol. 10, No. 2, p. 77.

[3] B.S. Bloom et al, Taxonomy of Educational Objectives: Handbook I, Cognitive Domain. Nueva York: David McKay, 1956.

[4] M. Valero-García y J.J. Navarro, Niveles de competencia de los objetivos formativos en las ingenierías, *VII Jornadas sobre la Enseñanza Universitaria de la Informática JENUI 2001*, p. 149

[5] Ideas and Rubric, Instructional Intranet, Chicago Public Schools. http://intranet.cps.k12.il.us/Assessments/Ideas_and_Rubrics/ideas_and_rubrics.html

Una apuesta por la motivación al alumnado en las asignaturas de programación: el sistema de evaluación continuada

Marisa Durán
Dpto. de Informática
Universidad de Extremadura
10071 Cáceres
e-mail: mlduran@unex.es

Andrés Caro
Dpto. de Informática
Universidad de Extremadura
10071 Cáceres
e-mail: andresc@unex.es

Pablo G. Rodríguez
Dpto. de Informática
Universidad de Extremadura
10071 Cáceres
e-mail: pablogr@unex.es

Resumen

En el presente trabajo se exponen las dificultades ante las que se encuentran la mayoría profesores que imparten asignaturas de programación en la Universidad española. La excesiva masificación en las aulas, la convivencia de diferentes planes de estudios, la limitación de aulas y de espacio en las mismas, etc... provocan una desmotivación en el alumnado que conduce al fracaso. En este artículo se propone como una posible solución la realización de un sistema de evaluación continuado que implique mayor motivación en el alumnado. Además, ante un foro de discusión como el presente, se comparten las experiencias que, en los últimos cuatro años, han ido adquiriendo los autores del trabajo.

1. Introducción

Las asignaturas programación suponen un gran número de créditos en las titulaciones de informática que se imparten en la actualidad en la Universidad española. Los créditos prácticos conllevan la resolución de un supuesto de una cierta envergadura. Uno de los objetivos de este tipo de asignaturas es conseguir que el alumno aprenda a afrontar problemas importantes y pueda llevar a cabo su resolución utilizando las estructuras de datos y las técnicas algorítmicas adecuadas. Sin embargo, la masificación en las aulas en este tipo de titulaciones es un problema muy común, que viene aumentado con el hecho de que, además, existe la posibilidad de que convivan diferentes planes de estudios simultáneamente. Además, la limitación de equipos, aulas, y espacios en las mismas, inevitablemente, lleva aparejada una cierta desmotivación del alumnado, lo que provoca un alto índice de fracaso en estas asignaturas.

En la sección 2 de este artículo se expone la problemática de las asignaturas en los últimos años en nuestra Universidad. En la sección 3 se comenta el material de apoyo elaborado por los profesores de la asignatura. Ya en la sección 4 se expone la propuesta de evaluación continua puesta en marcha hace tres años. En la sección 5 se discute acerca de los resultados obtenidos, exponiendo la opinión de los alumnos en la sección 6. Para terminar, en la sección 7 se exponen las principales conclusiones obtenidas.

2. Las asignaturas de programación en nuestra Universidad

En la Universidad de Extremadura, para las tres titulaciones de informática (I.I., I.T.I.S. e I.T.I.G.), se imparten dos asignaturas de programación troncales en primer curso de carrera: “Elementos de Programación” (anual, de 9 créditos) y “Laboratorio de Programación I” (2º cuatrimestre, 6 créditos). En segundo de carrera se imparte una asignatura troncal, “Estructuras de Datos y Algoritmos” (anual, 9 créditos) y otra obligatoria, “Laboratorio de Programación II” (2º cuatrimestre, 6 créditos).

Las dos asignaturas de *laboratorio* suponen la realización de prácticas de una cierta magnitud (según el curso) que complementen los conceptos teóricos expuestos en las correspondientes asignaturas anuales. El éxito en este tipo de asignatura influiría, irremediablemente, en las

asignaturas anuales. Sin embargo, son asignaturas que han venido sufriendo un alto índice de abandono en los últimos años, debido, en gran medida, a que el alumno pospone la realización de la práctica hasta el final del curso, encontrándose en ese momento con falta de tiempo y de fluidez. Por tanto, la desazón y la falta de motivación del alumnado están a la orden del día.

En un intento por promover la realización de los supuestos prácticos en los tiempos establecidos, hace tres cursos los autores de este trabajo decidieron ofrecer al alumno la posibilidad de someterse a una evaluación continuada en la asignatura de "Laboratorio de Programación II", asignatura en la que imparten docencia fundamentalmente y en la que se centrará el estudio de ahora en adelante. Para aprobar esta asignatura se exige que los alumnos desarrollen *individualmente* una práctica en C++, utilizando técnicas de P.O.O., con las especificaciones expuestas en un enunciado que se les entrega al principio del curso.

La evaluación de esta asignatura se basa en los siguientes criterios:

- Realización de una práctica desarrollada de forma individual, en lenguaje C++. La práctica debe entregarse debidamente resuelta y documentada.
- Superación, si se estima necesario, de un examen práctico que consistirá en realizar una modificación de la práctica.
- Entrega de una documentación que consistirá en: Manual del Usuario y Manual del Programador (que incluirá el código fuente), así como de discos con los ficheros fuente de la práctica y modificación.

Dicha materia consta de 6 créditos, 4.5 prácticos y 1.5 teóricos, y se imparte durante el segundo cuatrimestre del segundo curso de la carrera.

De lo expuesto anteriormente se concluye que el método de evaluar a los alumnos es el de que cada alumno, de forma individual, realice la práctica durante el cuatrimestre, y sea convocado a un examen en el que realicen, en un periodo de dos horas, una modificación a la práctica. Este tipo de exámenes suele consistir en añadir nuevos métodos en algunas de las clases, en añadir una nueva clase derivada a alguna de las jerarquías de herencia que conforman la práctica o en desarrollar algoritmos que solucionen los problemas propuestos.

El objetivo perseguido con este tipo de examen es la discriminación entre alumnos que han desarrollado la práctica por sí mismos con respecto a aquellos que acuden a otros medios para obtener la resolución final de la misma. El alumno es evaluado en función de la capacidad demostrada para modificar su ejercicio. Así, se premia a las prácticas bien estructuradas y organizadas en cuanto a modularización. El alumno entrega al profesor el código fuente desarrollado, el de su modificación y una documentación que incluye un manual del usuario y un manual del programador.

Durante los últimos años la práctica planteada a los alumnos consiste, a grandes rasgos, en utilizar estructuras de almacenamiento lineales, tales como colas y listas, sobre las que aplican polimorfimo y genericidad. Siempre han de llevar a cabo el diseño de una jerarquía de clases con métodos propios y métodos heredados, a fin de aplicar el polimorfismo, y entender el mecanismo de la herencia. Además trabajan con una estructura de almacenamiento jerárquico, que suele ser un árbol, manejan a su vez entrada y salida sobre ficheros de texto e implementan estructuras de tipo grafo.

Se ha de considerar el hecho de que desde el curso 1998/99 se introduce un nuevo plan de estudios, y pierden su derecho a docencia aquellos alumnos que no se acogen a este nuevo plan. A esta situación se ha de añadir que en los últimos años se venía arrastrando un alto nivel de fracaso, lo que originaba desmotivación previa en los alumnos y, por otro lado, una sensación generalizada de "cierto temor" y abandono de la asignatura desde los primeros días del curso.

3. Material de apoyo a la docencia

Los autores de este trabajo han confeccionado un guión del curso teniendo en cuenta las semanas lectivas de que consta el cuatrimestre, elaborando una colección de supuestos prácticos totalmente resueltos, que sirven, en algunos casos, como ejercicios que se desarrollan y se discuten en clase, y en otros casos como material complementario que se facilita a los alumnos, para motivar su autoformación.

Los ejercicios propuestos, aunque diferentes en su planteamiento, mantienen una relación directa con la resolución del supuesto práctico que deben realizar los alumnos, de modo que se proponen para que sean los propios estudiantes los que, durante las sesiones prácticas desarrolladas en las salas de ordenadores, van confeccionando su propia biblioteca de ejercicios propuestos y resueltos por ellos mismos. Estos ejercicios suponen ejemplos claros y evidentes de los conceptos fundamentales de la asignatura, tales como encapsulación, herencia, polimorfismo, sobrecarga, genericidad y uso de estructuras de datos.
La bibliografía básica utilizada para el desarrollo de esta asignatura se muestra al final del trabajo.

4. Evaluación continua

En el curso 2000-01 el número de alumnos ascendía aproximadamente a 550 (400 del plan nuevo y el resto del plan antiguo). Ante esta situación parecía difícil introducir innovaciones docentes en la asignatura. Sin embargo, tras analizar la situación de los años anteriores, los autores consideraron necesario establecer mecanismos que contribuyesen a incrementar el grado de motivación de los alumnos y su implicación en la asignatura durante el curso. La experiencia demostraba que el alumno se enfrentaba a la materia con total "desconfianza" sobre sus capacidades y, en muchas ocasiones, no resolvía la práctica por sí mismo, sino que buscaba ayuda en otros medios, tales como academias, alumnos de años anteriores u otras soluciones. A fin de evitar esta situación, se apostó por llevar a cabo un sistema de *evaluación continua*. Dicho sistema consistía en exigir a los alumnos que, a lo largo del curso, fuesen entregando módulos de la práctica. No era suficiente la entrega de dichos módulos, sino que, además, se iría haciendo un pequeño examen por cada una de las entregas. De esta forma, se fijaba una fecha para la conclusión de los módulos y además, el alumno recibía información acerca de los errores que iba cometiendo, permitiéndose así la posibilidad de subsanarlos, en lugar de arrastrarlos hasta el fin de la práctica como podía ocurrir con el método de evaluación anterior. La idea de que "a programar se aprende programando" no sólo se transmite al alumno, sino que, además, se fomenta y potencia mediante el trabajo diario que supone la realización por partes de la práctica.

El alumno es informado de las fechas en las que se van a llevar a cabo estos controles en el mismo momento en que se les entrega el enunciado de la práctica. Después de tres cursos realizando este tipo de evaluación, el esquema se fundamenta en cinco entregas: las tres primeras suponen la resolución de forma independiente de determinados aspectos de la práctica, tal y como se comenta a continuación, siendo la cuarta la que implica la integración de estas tres entregas en un único proyecto.

- 1ª Entrega: finales de marzo. Se exigen las estructuras de datos lineales y la implementación de algunas clases en herencia, así como el uso de polimorfismo.
- 2ª Entrega: finales de abril. Se exige el Árbol Binario Ordenado y especificación de las clases que se almacenan en el mismo.
- 3ª Entrega: mediados de mayo. Se pretende que se implemente la estructura de tipo grafo y las clases relacionadas con el mismo, así como el uso de programación genérica.
- 4ª Entrega, finales de mayo. Integración final de los módulos. Entrega de código fuente.
- 5ª Entrega, en junio. Documentación (manual del usuario y del programador).

Hay que tener en cuenta que el alumno no pierde nunca la posibilidad de ser evaluado del mismo modo en que se venía haciendo en anteriores cursos. No obstante, con esta propuesta, los alumnos que han ido superando un número mínimo de entregas, no tendrán la necesidad de presentarse al examen final que seguirá consistiendo en realizar una modificación de la práctica tal y como se ha expuesto en la sección 2.

Así mismo, la evaluación continua y los controles periódicos que ésta supone ha conllevado un espectacular aumento en las consultas realizadas por los alumnos en los horarios de tutorías. Si bien antes los alumnos apenas asistían a consultar sus dudas, y lo hacían a escasos días de la finalización del curso, en la actualidad los alumnos utilizan este recurso docente con bastante frecuencia, lo cual propicia una mayor interacción alumno-profesor, mejores relaciones entre ambos y más confianza mutua.

5. Discusión

Según lo expuesto, se ve bastante clara la necesidad de mejorar la *actitud participativa* de los alumnos en la asignatura. El cambio de planes de estudio, la nota mínima de acceso tan asequible para estas titulaciones y las condiciones socioeconómicas de la región, son factores que influyen en el alto número de matriculados con bajo interés en la carrera. La mejora expuesta en este trabajo hizo pensar a los autores del trabajo en la posibilidad de que los alumnos aún se desentendiesen más ante el esfuerzo que supone la evaluación continua. Sin embargo, con la perspectiva que suponen los tres cursos académicos que se lleva realizando este sistema de evaluación, los resultados no pueden haber sido mejores. La gran mayoría de alumnos matriculados se acoge al sistema de evaluación continua, intentando, al menos, seguir el ritmo de realización de la práctica. En la figura 1 se observa cómo el porcentaje de aprobados en los últimos años en las asignaturas de programación de segundo curso han evolucionado de una forma bastante similar. Desde el curso 1999/00 se ha incrementado el porcentaje de aprobados en las asignaturas, de forma constante. Sin duda, la evaluación continua y la posibilidad para los alumnos de aprobar la asignatura sin presentarse al examen final, hecho que, por experiencia, puede decirse que les motiva bastante, propicia un estudio y trabajo diario que implica que la asignatura sea superada casi sin esfuerzo. Y este estudio se ve reflejado en las dos asignaturas de programación del curso.

6. Opinión de los alumnos

Es muy destacable la buena acogida que el sistema de evaluación continua ha tenido en estos años por parte de los alumnos. Los autores han realizado encuestas anónimas a los alumnos al final de cada curso, siendo unánime la buena acogida que el sistema de evaluación continua ha tenido entre lo mismos.

A continuación se exponen algunas de las opiniones más repetidas expresadas por los alumnos:

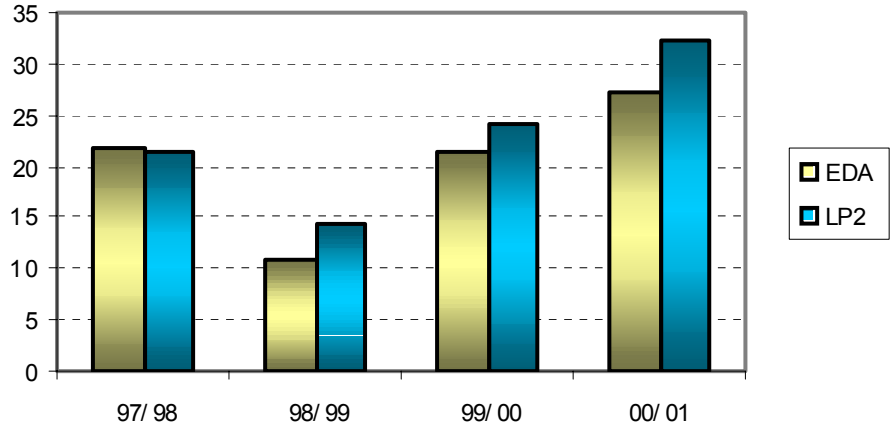


Figura 1. Porcentaje de aprobados en los últimos años en las asignaturas de programación de segundo curso

- El método de aprendizaje es muy bueno. Si te tomas en serio la evaluación continua, aprendes. Es duro, pero es necesario y eficiente.
- Lo mejor de la asignatura es la evaluación continua: el seguimiento a los alumnos es bueno.
- La "obligatoriedad" de ir entregando por partes la práctica para la evaluación continua hace que nos planteemos la asignatura día a día y no la dejemos para el final, que es lo que suele ocurrir casi siempre con las prácticas de este tipo.
- El método nos obliga a trabajar a diario y nos hacemos una idea de cómo se implementa un proyecto poco a poco, ya que de otra manera no sabemos ni como empezar.
- El método de evaluación es más exigente, pero creo que así se aprende más y se te evalúa de forma más justa.
- El método de evaluación es muy bueno. Se obliga al alumno y al profesor a un avance progresivo. El alumno aprende poco a poco y el profesor corrige poco a poco.
- La asignatura es más fácil de llevar, porque hacerlo todo al final y jugarse la asignatura en un examen es bastante más difícil.
- Con el trabajo diario he tenido mayor motivación y he prestado más atención a la asignatura.
- En mi caso, el tener una fecha tope para entregar algo es una buena manera de obligarme a trabajar.

7. Conclusión

La evaluación continuada de las prácticas de programación han resultado muy satisfactorias en la enseñanza de las asignaturas de programación, ya que han permitido la motivación de los alumnos, el estudio cotidiano de los conceptos teóricos y el desarrollo diario de ejercicios, programas y de la propia práctica de la asignatura. El estímulo que supone para el alumno la posibilidad de "aprobar sin presentarse al examen", así como el temor al examen final, propician la elevada motivación que los autores han observado en el alumnado. La elevada aceptación de este sistema por parte del alumnado, así como los buenos resultados obtenidos en los últimos años hacen que los autores del trabajo recomienden este sistema de evaluación para las asignaturas de programación.

Bibliografía

[1] Bell, D., *The essence of programming using C++* . Prentice-Hall, 1997.

[2] Charte, F. *Programación Orientada a Objetos con Borland C++*. Anaya, 1993.

[3] Deitel, H.M. y Deitel, P.J.: *Cómo programar en* C/C++. 2ª edición, Prentice-Hall, 1998.

[4] Mansfield, K.C. *An introduction to programming in C++*. Prentice-Hall International, 1997.

[5] Joyanes, L. *Programación en C++: algoritmos, estructuras de datos y objetos*. McGraw-Hill, 2000.

[6] Heileman, G. *Estructuras de Datos, Algoritmos y Programación Orientada a Objetos*. McGraw-Hill, 2000.

[7] Schildt, H. *C/C++: Manual de referencia*. McGraw-Hill, 1999

[8] Schildt, H. *C: Guía para usuarios expertos*. Osborne / Mc Graw-Hill, 1991.

[9] Stroupstrup, B. *El lenguaje de programación C++*. Addison-Wesley, 2002.

Informática en otras carreras

Propuesta para la enseñanza de Informática en titulaciones de Ingeniería Química

Ana Belén Moreno Díaz
Dept. de Informática, Estadística y Telemática
Universidad Rey Juan Carlos
28933 Móstoles (Madrid)
e-mail: a.b.moreno@escet.urjc.es

Juan José Pantrigo
Dept. de Informática, Estadística y Telemática
Universidad Rey Juan Carlos
28933 Móstoles (Madrid)
e-mail: j.j.pantrigo@escet.urjc.es

Rosalía Peña
Dept. de Informática, Estadística y Telemática
Universidad Rey Juan Carlos
28933 Móstoles (Madrid)
y Dept.CC. Computación.
Universidad de Alcalá de Henares
28871 Alcalá de Henares (Madrid)
e-mail: rpr@uah.es

Resumen

Este trabajo describe el enfoque y desarrollo de los cursos de introducción a la informática impartidos en las titulaciones Técnica y Superior de Ingeniería Química en la Universidad Rey Juan Carlos de Madrid. Desde hace varios años, hemos venido realizando cambios en las asignaturas para adecuar su contenido a las necesidades de los ingenieros químicos, aumentar el grado de consecución de los objetivos y fomentar el interés y aprovechamiento de los alumnos. En este artículo presentamos nuestra experiencia. Se analizan los aspectos que se deben considerar al elaborar los temarios. Se evalúa la adecuación de *Fortran90* como lenguaje soporte para un primer acercamiento a la programación y se justifica su elección. Se estudia qué tipo de prácticas contribuye a aumentar la motivación de los alumnos y favorece el aprendizaje de la metodología de la programación.

1. Introducción

En el diseño de una asignatura hay que considerar diversos factores: los descriptores oficiales, las recomendaciones de las asociaciones profesionales, la evolución y el estado actual de la materia, el plan de estudios, la formación previa del alumnado, el número de créditos teóricos y prácticos y la experiencia de los profesores.

En este trabajo se expone y justifica la propuesta de las asignaturas denominadas "Informática" y "Fundamentos de Informática", de las titulaciones *Ingeniería Química* (Superior) e *Ingeniería Técnica Industrial en Química Industrial* (Técnica), respectivamente. Ambas son obligatorias, y se imparten en el primer cuatrimestre del primer curso (15 semanas), con cinco horas lectivas por semana, dos de teoría y tres de práctica.

Al elaborar los temarios hemos tenido en cuenta, entre otros factores, la opinión de profesores del área de *Ingeniería Química* de esta universidad.

Al tratarse de una introducción a la informática de corta duración, el lenguaje de programación debe ser imperativo, sencillo, expresivo y fácil de asimilar, para disponer de más tiempo para el trabajo práctico.

Para motivar a los alumnos se proponen prácticas cercanas a la ingeniería, que en numerosas ocasiones, requieren conocimientos sobre química, matemáticas o física, haciendo énfasis en la metodología de la programación estructurada y del diseño descendente y en la adquisición de buenos hábitos. A la elaboración de este tipo de prácticas ha contribuido que los profesores (y autores del trabajo), pertenecientes al área de Lenguajes y Sistemas de Información e Ingeniería del Software, son titulados superiores en Química o en Física.

La estructura del artículo es la siguiente: la sección 2 presenta los conocimientos de informática más relevantes en el ámbito de la Ingeniería Química. La sección 3 expone los objetivos docentes. La sección 4 resume el contenido teórico y el material bibliográfico. La sección 5 justifica la elección del software. La sección 6 describe la organización y el enfoque de

los contenidos prácticos, mostrando algunos ejemplos. La sección 7 concreta la metodología empleada para el seguimiento de la asignatura y los criterios de evaluación. En la sección 8 se valora esta oferta docente, proponiendo futuras mejoras y en la sección 9 se exponen las conclusiones.

2. Informática en Ingenierías Químicas

Los titulados en Ingeniería Química en el mercado laboral en España van a necesitar para su ejercicio profesional conocimientos de informática, por lo que se recomienda su inclusión en los planes de estudio. Los conocimientos que requerirán con mayor probabilidad [1][2][3][4][5] son:

- *Ofimática e Internet.*
- *Diseño asistido por computador (CAD),* mediante paquetes como *AutoCAD*, *AutoSketch* o *CADHEY,* que permiten definir objetos (por ejemplo, componentes de plantas químicas) y manipularlos gráficamente.
- *Simuladores de procesos químicos:* además de las funciones programadas, suelen proporcionar flexibilidad, admitiendo módulos externos (programados muy frecuentemente en *Fortran90* o *Visual Basic*).
- *Paquetes de cálculo matemático,* como *MATLAB* o *Maple*, que disponen de librerías predefinidas para la realización de cálculos y visualización de gráficos.
- *Programación:* las herramientas comerciales no resuelven todos los problemas, a veces es necesario realizar desarrollos propios. Además, en los casos en que el entorno de trabajo cuente con equipos multidisciplinarios, el conocimiento de conceptos de programación favorece la comunicación entre químicos e informáticos.

2.1. Elementos curriculares

Los contenidos de nuestras asignaturas siguen las directrices de las asociaciones de profesionales de Informática y Química sobre formación en informática. El Documento de Política de Educación Científica [2] de la *American Chemical Society* recomienda el uso habitual de ordenadores modernos y el acceso a Internet en las titulaciones de Química. También recomienda que la simulación por ordenador complemente, pero no sustituya, al laboratorio de química.

En el *Computing Curricula 2001* [6] (Volumen *Computer Science* [7]) se dan recomendaciones para cubrir la docencia de informática en titulaciones no informáticas. La definición de cursos de informática para estas titulaciones ha de seguir las fases:

- *Especificación:* los objetivos del curso se deben consensuar entre profesores informáticos y no informáticos (en este caso, químicos).
- *Diseño:* los objetivos educativos se tienen que concretar, especificando las destrezas y conceptos que deben alcanzar los alumnos.
- *Implementación*: para la estructuración del curso se deben tener en cuenta los conocimientos previos, los métodos de trabajo, las motivaciones de los alumnos y los recursos disponibles.
- *Evaluación:* debe ser activa y redundar en una constante actualización.

2.2. Formación previa de los alumnos

A estas asignaturas se incorporan todos los años 75 alumnos nuevos, y entre veinticinco y treinta repetidores; en total, hay cien alumnos de media en cada grupo. La mayoría de los alumnos de la Superior, provienen de selectividad. Sin embargo, la procedencia de los alumnos de la Técnica es de Módulos de Formación Profesional, Bachillerato y de Titulaciones de Ingeniería Superior que a veces no han superado. La calificación necesaria para acceder a esta titulación es menor, por lo que su formación previa es inferior. En cuanto a sus conocimientos de informática, la situación es también muy variada: algunos conocen un lenguaje de programación (normalmente Pascal, C o Basic) desde una perspectiva muy práctica, pero deficiente desde el punto de vista conceptual y metodológico, lo que dificulta nuestra tarea. Otros solamente conocen el sistema operativo *Windows* y el manejo de algunos programas, y para un 30% de los alumnos este es su primer contacto con el ordenador.

2.3. Contexto académico

En cuanto a las materias relacionadas con la que nos ocupa conviene señalar que en ambas titulaciones hay una asignatura de Diseño Asistido por Ordenador, en la que se aprende el manejo de *AutoCAD*. En algunas asignaturas de los últimos cursos los alumnos trabajan con simuladores de procesos químicos y para cursarlas conviene tener conocimientos de programación. En las asignaturas de Matemáticas realizan prácticas de laboratorio utilizando *MAPLE*. También existe una asignatura de libre elección sobre Internet. En el tercer curso de la titilación Superior se imparte una optativa de ofimática y *Visual Basic*.

3. Objetivos de la asignatura

Debido a que gran parte de nuestros alumnos no ha tenido contacto con un ordenador anteriormente, esta asignatura tiene un marcado carácter introductorio. Por otro lado, ésta es la única asignatura obligatoria en la que se estudian conceptos de informática, por lo que debe cubrir las necesidades profesionales del ingeniero. Por estos motivos, el objetivo central es poner a disposición del alumno *los conocimientos necesarios para que consiga identificar y comprender la utilidad de los conceptos fundamentales de la informática, haciendo énfasis en el algoritmo y en la estructura física y lógica de los ordenadores actuales.* Concretamente, se pretende que, finalizado el proceso formativo, el alumno:

- conozca de modo sucinto los conceptos básicos referentes a la evolución de la informática hasta nuestros días,
- comprenda cómo funciona un ordenador y conozca sus componentes básicos,
- conozca las funciones del sistema operativo, y maneje el entorno de uno de ellos,
- conozca los elementos básicos de un lenguaje de alto nivel y su sintaxis en *Fortran90*, más concretamente se presta atención a: (a) tipos de datos simples y compuestos; (b) estructuras secuenciales, selectivas, repetitivas y subprogramación, (c) normas de estilo de programación referentes a la construcción y documentación de programas;
- conozca algoritmos clásicos (recorrido, búsqueda, ordenación, entre otros),
- sea capaz de diseñar algoritmos y de crear programas sencillos utilizando la metodología de la programación estructurada y el diseño descendente,
- conozca la utilidad de algún paquete de software científico para la Ingeniería Química, y maneje *MATLAB* (este objetivo sólo se introduce en la titulación Superior).

4. Contenido teórico del curso

Los contenidos teóricos de la asignatura *Informática* (Superior) están divididos en tres partes. La primera ocupa un cuarto del curso y trata los conceptos básicos de la informática. En la segunda se lleva a cabo una introducción a la programación estructurada con ayuda de *Fortran90*, y ocupa la mayor parte del cuatrimestre. La tercera, y menos extensa, se dedica al estudio de los paquetes de software matemáticos, utilizando *MATLAB*. A continuación, se muestra un esquema de los temas especificando las horas teóricas y prácticas:

Parte I: Introducción a la Informática
- **Tema 1:** Conceptos básicos (3T+1P)
- **Tema 2:** Arquitectura de ordenadores (3T+1P)
- **Tema 3:** Sistemas operativos (3T+4P)

Parte II: Programación en lenguaje *Fortran90*
- **Tema 4:** Los primeros pasos en la programación en *Fortran90* (2T)
- **Tema 5:** Tipos de datos, constantes, variables y expresiones. Sentencias de asignación y entrada/salida (4T+7P)
- **Tema 6:** Sentencias de selección (2T+5P)
- **Tema 7:** Sentencias de repetición (2T+7P)
- **Tema 8:** Metodología del diseño descendente. Subprogramas (3T+5P)
- **Tema 9:** Arrays (3T+7P)
- **Tema 10:** Ficheros (2T+4P)

Parte III. Paquetes de software científico
- **Tema 11.** Software de cálculo científico para la Ingeniería Química (3T+4P)

En la titulación Técnica se omiten los apartados *registros* del tema 5, *módulos* del tema 8 y los *paquetes de software científico*.

Al principio de cada tema se presentan los objetivos que los alumnos deben alcanzar. Las

clases teóricas se imparten con la ayuda de un ordenador portátil y un cañón de proyección. Se recomienda al alumno el uso de bibliografía.

No hemos encontrado un único libro que abarque la asignatura en su totalidad, siendo los libros más utilizados:

- J. Glenn Brookshear, *Introducción a las ciencias de la computación*, 4ª Edición, Addison-Wesley Iberoamericana, 1995.
- F. García Merayo, *Lenguaje de Programación Fortran90*, Ed. Paraninfo, 1999.
- T.M.R. Ellis, Ivor R. Philips and T.M. Lahey, *Fortran90 programming*, Addison.Wesley 1994.
- C. Pareja, A. Andeyro y M. Ojeda, *Introducción a la Informática: I. Aspectos Generales*, Ed. Complutense, 1994.
- [Etter97] Delores M. Etter, *Solución de problemas de Ingeniería con MATLAB*, 2ª edición, Prentice Hall, 1997.

5. Otros aspectos considerados

A continuación se justifica la elección del sistema operativo, el lenguaje, el entorno de programación y el paquete de software científico.

5.1. Elección del sistema operativo

Es fundamental entender el contexto en el que se ejecutan los programas en el ordenador, y por tanto, el alumno debe comprender los conceptos fundamentales del sistema operativo. En entornos personales, el usuario debe ser autónomo, al menos en el manejo cotidiano de sus ficheros. Sin embargo, en grandes sistemas, se suele disponer de personal especializado para administración y mantenimiento, que dará soporte al especialista Químico que se acerca al sistema con unos fines concretos.

Adicionalmente, se desea exista la mayor disponibilidad de puestos para los alumnos en horas de trabajo personal, todo ello nos conduce a elegir *Windows* como sistema operativo.

5.2. Valor pedagógico de *Fortran90*

Varios autores [8-9] han estudiado las características que deben poseer los lenguajes utilizados para el primer acercamiento a la programación imperativa, coincidiendo en las siguientes:

1 un sistema de tipos fuerte
2 simplicidad
3 entrada/salida sencillas, para poder presentarlas desde el inicio del curso
4 homogeneidad (sistemática o coherencia en su sintaxis)
5 controles que eviten errores conceptuales
6 todos los mecanismos necesarios para facilitar la programación modular y estructurada
7 características que permitan fomentar buenos hábitos de programación (comentarios, sangrado sistemático de los bloques de ejecución, por ejemplo), y
8 tipos de datos derivados.

A continuación, se examina cada una de estas características en *Fortran90*, comentando la forma en que las implementamos en nuestra docencia.

1 Se fuerza la declaración explícita de tipos mediante la instrucción `IMPLICIT NONE`. *Fortran90* dispone de funciones intrínsecas de conversión entre tipos, permitiendo no presentar las reglas de compatibilidad y forzar la conversión explícita, impidiendo expresiones de tipos mixtos.
2 *Fortran90* es un lenguaje muy rico, versátil y potente, que no se puede calificar de simple; sin embargo, en nuestra aproximación docente, hemos optado por introducir, exclusivamente, las estructuras que posibilitan la programación modular y estructurada y que detallamos en los puntos siguientes. Fomentar el empleo de las estructuras más generales del lenguaje conlleva la ventaja de que las habilidades adquiridas por nuestros estudiantes son directamente transportables a otros lenguajes imperativos, con sólo disponer de una tabla de conversión de la sintaxis.
3 Del mismo modo, optamos por restringirnos a la entrada/salida sin formato que es sencilla e intuitiva:

```
READ*, variable[,variable]n
PRINT*,{expresión/literal}
[,{expresión/literal}]n
```

Sintaxis 1. Entrada/salida

4 Las estructuras que presentamos a los alumnos disponen sistemáticamente de cabecera y fin de estructura. El hecho de que el fin sea

específico para cada una (IF/END IF, FUNCTION/END FUNCTION, DO/END DO, etc.) aumenta la legibilidad frente al uso del mismo terminador para todas ellas. La sintaxis es homogénea, de modo que los modificadores o parámetros de cada estructura se ubican de forma sistemática (en su versatilidad, *Fortran90* posibilita formas abreviadas, o sintaxis alternativas, pero manejamos solamente la forma más estándar). Cada instrucción por defecto es una línea, pero puede especificarse su continuidad en los casos excepcionales en que se necesite (& carácter de continuación).

5 *Fortran90* dispone de controles que impiden realizar manipulaciones inadecuadas, por ejemplo: (a) modificar la variable de control de un bucle con contador dentro de su ámbito; (b) la dirección del flujo de los argumentos de subprogramas se declara de forma explícita IN|OUT|INOUT; (c) por defecto, todas las variables declaradas, tanto en el programa principal como en subprogramas son locales, aunque existe la posibilidad de otorgar visibilidad, expresándolo de forma explicita.

6 Las estructuras selectivas que manejan nuestros alumnos son IF y CASE (Sintaxis 2). Las tres estructuras repetitivas clásicas son (Sintaxis 3): bucle controlado por contador y bucles controlados por expresión lógica pre y post-probados. Aunque la sintaxis de esta última pueda generar, en principio, alguna reticencia, su funcionalidad es exactamente la misma que el REPEAT/UNTIL de Pascal, saliendo a la instrucción inmediatamente posterior a END DO si se satisface la condición, e iniciando una nueva iteración en caso contrario, de modo que no hay motivo para omitirla.

```
IF (exp_logica) THEN
   conjunto_instrucciones
[ELSE conjunto_instrucciones]
END IF

SELECT CASE (exp_E|C|L)
[CASE (valor:valor)
      conjunto_instrucciones]1-n
[CASE DEFAULT
      conjunto_instrucciones]
END SELECT
```

Sintaxis 2. Estructuras de selección

```
DO var_Entera=valInicial,valFinal
   [,incremento]
conjunto_instrucciones
END DO

DO WHILE (expresión lógica)
conjunto_instrucciones
END DO

DO conjunto_instrucciones
IF (expresión_lógica) EXIT
END DO
```

Sintaxis 3. Estructuras de repetición

Como ya se ha comentado, se declara explícitamente la dirección del flujo de los argumentos de los subprogramas, conforme a la Sintaxis 4:

```
SUBROUTINE nombreSubrutina [(arg
[,arg]n)]
[tipo, INTENT(IN|OUT|INOUT)::
arg [,arg]n]
END SUBROUTINE nombreSubrutina

tipo FUNCTION nombreFuncion
([argumento][,argumento]n)
[tipo,INTENT(IN)::arg[,arg]n]
END  FUNCTION nombreFuncion
```

Sintaxis 4. Estructuras de selección

7 *Fortran90* dispone de tipos de datos derivados, y las variables de éstos pueden ser empleadas en cualquier lugar en que se pudiera usar una variable de tipo simple, incluido el valor devuelto por las funciones.

8 Se pueden incluir espacios en blanco, tabuladores, líneas en blanco y comentarios, en cualquier punto del programa, lo que permite imponer estilos de sangrado, y documentación adecuados. Adicionalmente, *Fortran90* no es sensible al modo mayúscula/minúscula de los caracteres, y los identificadores pueden tener hasta 31 caracteres, características que facilitan la diferenciación entre los componentes del lenguaje y la elección adecuada de identificadores. Hemos optado por la escritura de las palabras reservadas del lenguaje en mayúsculas y los identificadores en minúsculas, a excepción de la primera letra de cada palabra en los nombres compuestos. Entendemos que establecer desde el primer momento del curso estas

pautas facilita la legibilidad del código y la conveniencia de adoptar una metodología.

Estas características permiten concluir que *Fortran90* es pedagógicamente adecuado para la ejercitación de alumnos noveles.

5.3. Elección del lenguaje

El hecho ser uno de los lenguajes más empleados en la programación de simuladores en química nos condujo a evaluarlo para su utilización en estas asignaturas.

Además, *Fortran90* es muy potente para cálculos matemáticos, concretamente, para el manejo de *arrays*, tan frecuentemente utilizados en Química (por ejemplo, para el estudio de energías de enlace o simetrías moleculares); en particular, es posible realizar cálculos entre *arrays* n-dimensionales con la misma sintaxis empleada para variables simples (Sintaxis 5).

```
INTEGER,DIMENSION[N,M]::a,b,c
c=2*b+a-c
PRINT*,c
```

Sintaxis 5. Declaración y uso de arrays

En [10] se describen más características de este lenguaje.

Por todo lo anterior y por su adecuación pedagógica, llegamos a la conclusión de que *Fortran90* es una elección correcta para nuestra asignatura.

5.4. El entorno de programación

Para Lahey [8] las características pedagógicas del entorno de programación son tan importantes como las del lenguaje en el entrenamiento de alumnos principiantes y propone el entorno *Elf90*, desarrollado específicamente para fines docentes.

Las sesiones prácticas de programación se están desarrollando sobre *Plato* (de *Salford*). Es un entorno que permite tener abiertos simultáneamente varios archivos, copiar información de unos a otros y compilar y ejecutar sin salir de la edición. Sin embargo, los mensajes del compilador son muy escuetos, con frecuencia imprecisos y no dispone de utilidades para trazar y depurar. La mejora de estos aspectos haría más confortable y eficiente el aprendizaje. Con frecuencia, en nuestras universidades resulta más difícil adquirir software, que incurrir en cualquier otro tipo de gastos, este es el motivo de mantener el actual entorno.

5.5. Elección del paquete de software científico

Es aconsejable que en estas titulaciones se aprenda algún lenguaje interactivo, con herramientas para la producción de gráficos y construcción de prototipos, como *Mathematica*, *MATLAB*, *Maple* o *MATHCAD*, siendo todos ellos potentes herramientas de cálculo matemático. Nuestros alumnos trabajan con *Maple* en asignaturas de matemáticas. Con objeto de que se aproximen a otro paquete de software científico, muy extendido en las empresas del sector, se ha escogido *MATLAB*, que posee una amplia gama de librerías específicas para distintas especialidades de la ingeniería; por su sencillez de manejo y por la semejanza de su sintaxis a la de *Fortran90*, resulta adecuado para esta asignatura [11], ya que una vez conocido *Fortran90*, es rápido, sencillo, intuitivo y casi inmediato de aprender.

6. Prácticas del curso

Los 4,5 créditos prácticos se invierten en la realización de ejemplos, ejercicios y prácticas de programación. Semanalmente se dedica una hora a resolver y comentar ejercicios en el aula y dos en el laboratorio.

Durante el curso se proponen alrededor de 60 ejercicios en hojas de problemas, de los que gran parte se resuelven posteriormente en clase y del resto se proporcionan las soluciones comentadas. Los ejercicios propuestos cubren preguntas cortas sobre el contenido de la parte I del curso, construcción de programas sencillos en *Fortran90* sobre cada uno de los temas y ejercicios de *MATLAB*.

Las prácticas de *MS-DOS*, *Windows* y *MATLAB* son tutoriales que contienen conceptos teóricos intercalados con ejercicios prácticos. Las prácticas de programación tienen carácter obligatorio y deben ser entregadas al profesor. Cada una de ellas consiste en la realización de un programa que ayuda a sedimentar los conceptos teóricos correspondientes al capítulo más reciente, ponerlos en práctica y descubrir su utilidad para

resolver problemas. Como muestra, se describen algunas de estas prácticas:

- ***Integral definida***
  Objetivo: Practicar el diseño y la codificación de algoritmos con estructuras de repetición controladas por contador.
  Práctica: Se propone la construcción de un programa para calcular una integral definida de una función conocida, mediante el método de los trapecios. Este método aproxima el valor de la integral de una función en un intervalo *[a, b]* de la recta real, a la suma de las áreas de los trapecios bajo la curva, que resultan al efectuar *N* divisiones en el intervalo.

- ***Combinatoria***
  Objetivo: Practicar el diseño y la codificación de algoritmos que usen estructuras de selección y repetición controladas por contador, por expresión lógica y subprogramación.
  Práctica: Se propone la construcción de un programa que ofrezca al usuario un menú en el que puede optar entre calcular combinaciones *(Cnk = n!/[(k!)(n-k)!])*, permutaciones *(Pnk = n!/k!)*, o salir. Posteriormente se pedirán los datos *n* y *k*. Puesto que el factorial de un número se requiere en 5 puntos del programa, el alumno descubre la utilidad de codificar una función para calcularlo. El algoritmo iterativo para el cálculo del factorial es conocido por el alumno.

- ***Química del Carbono***
  Objetivo: Practicar el diseño y la codificación de algoritmos que usan cadenas de caracteres, estructuras de selección y repetición controladas por expresión lógica y subprogramación.
  Práctica: Se propone la construcción de un programa que analice una fórmula de la química del carbono proporcionada por el usuario en una cadena de caracteres. Este programa debe llamar a un subprograma que valide la entrada (caracteres permitidos: '`C`', '`H`', '`O`', '`N`', '`2`', '`3`', '`4`', '`5`' y '`6`') y si es correcta, calcula el número de átomos de cada tipo que contiene la fórmula y su peso molecular.

En todo momento se respetan unas normas de estilo acordadas, que unifican la notación empleada, haciendo más legible el código. Estas normas hacen referencia, entre otras cuestiones, a la elección de identificadores significativos, uso de constantes no literales, tabulado de instrucciones según su bloque de ejecución, inclusión de comentarios (para especificar objetivos, valores de entrada y salida, pre y post-condiciones y documentar segmentos no obvios), separación de bloques de programa, evitar operaciones mixtas e impresión de literales explicando al usuario el dato pedido o mostrado.

7. Metodología y evaluación de alumnos

Las prácticas de ordenador son obligatorias. El alumno las entrega durante la semana siguiente a la que fueron propuestas, de manera que tiene un tiempo razonable para realizarlas. Se devuelven comentadas semanalmente, de forma que el alumno conoce la calidad de su trabajo y puede corregir sus errores.

Considerando el número de alumnos proponemos un método tradicional de evaluación, se realiza un examen final escrito, dividido en dos partes: teoría y práctica. Han de superar ambas para aprobar la asignatura. El alumno que suspende las prácticas en febrero, debe realizar otras distintas para septiembre. Cuando todas las prácticas están aprobadas, sus puntuaciones se promedian y normalizan entre 0 y 1 y se suman a la nota del examen final si este está aprobado. De esta forma se fomenta el seguimiento semanal de las prácticas.

La realización de hojas de problemas es voluntaria, pero muy recomendable para incrementar las habilidades de programación.

8. Evaluación de la asignatura

Algunos puntos de las encuestas docentes relacionados con el interés de la asignatura han mejorado al acercar los contenidos prácticos a problemas relacionados con su ámbito de conocimiento. A los alumnos, en general, les gusta la programación, aunque la encuentran

difícil. Valoran positivamente que se les entreguen comentadas sus prácticas de forma personalizada.

Reclaman disponer de un libro en español que cubra los conceptos de la metodología de la programación estructurada y el diseño descendente sobre *Fortran90* y que presente ejemplos y ejercicios.

Los resultados obtenidos en las dos titulaciones son diferentes. Un mayor número de alumnos de la titulación Técnica tiene dificultades para alcanzar los objetivos. Es posible que esto sea debido a la diferente formación previa.

Consideramos adecuado reducir la primera parte del temario para dedicar más tiempo a la programación.

Dado que la informática está cada vez más extendida, posiblemente el manejo del entorno del sistema operativo se podrá realizar de manera optativa, en horarios no lectivos, en los próximos cursos.

9. Conclusiones

Los estudiantes de Ingeniería Química necesitan conocimientos de: ofimática, *CAD*, paquetes matemáticos y desarrollo de habilidades algorítmicas, que les doten de cierta autonomía y les faciliten la comunicación en equipos de trabajo multidisciplinarios. La responsabilidad de buena parte de dicha formación recae en estas asignaturas.

El contenido de la asignatura ha tenido que ser ajustado en cada una de las titulaciones, en función de las características del colectivo.

Fortran90 muestra ser un lenguaje pedagógicamente adecuado como soporte en un curso de iniciación a la programación.

Fortran90 y *MATLAB* son una buena elección para alumnos de ingenierías químicas.

El acercamiento de las prácticas a su contexto contribuye a que los alumnos encuentren mayor sentido a la presencia de esta asignatura en su plan de estudios, mejorando su interés y rendimiento.

En un futuro próximo, se tenderá a trasladar el aprendizaje del entorno de *Windows* a seminarios fuera de horas lectivas y a mejorar el entorno de programación.

Referencias

[1] L. E. Romero Zúñiga, *Los titulados en ingeniería química ante el mercado laboral*, Ingeniería Química, Julio-Agosto 2000, pp. 201-206.

[2] *Education Policies for Sustainable Reform,* Society Committee on Education SOCED of the American Chemical Society ACS, Science, 2001.

[3] American Institute of Chemical Engineers, http://www.aiche.org/education/

[4] Asociación Nacional de Químicos de España, http://www.anque.es/index.htm

[5] European Federation of Chemical Engineering, http://www.efce.info.

[6] The Joint Task Force on Computing Curricula IEEE-CS/ACM. *Computing Curricula 2001*. http://www.computer.org/education/cc2001/

[7] The Joint Task Force on Computing Curricula IEEE-CS/ACM. *Computing Curricula 2001,* Computer Science Volume, December 2001, http://www.computer.org/education/cc2001/final/.

[8] Lahey T. M., Walker T., *Elf90-A first Programming Language*, Proceedings ASEE Annual Conference & Exposition, Washington, I, 1996.

[9] Kölling, M. et al. *Requirements for a first year OO teaching language,* SIGCSE 1995, 173-77.

[10] Lahey T. M., *A look at Fortran90*, http://www.lahey.com/lookat90.html.

[11] Delores M. Etter, *Solución de problemas de Ingeniería con MATLAB*, 2ª edición, Prentice Hall, 1997.

Experiencia del ISPJAE en la formación Informática de los Ingenieros Industriales

Msc. Ing. Mavis Lis Stuart Cárdenas
Msc. Diana Aguilera Reina
Msc. Miguel Angel Díaz Martínez
Ing. Yadary Ortega González

Resumen

El presente trabajo se desarrolló en la Facultad de Ingeniería Industrial del Instituto Superior Politécnico José Antonio Echeverría (ISPJAE) de Cuba. En el mismo se aborda la experiencia de enseñar la disciplina Informática en la carrera de Ingeniería Industrial.

El Ingeniero Industrial cubano, con su función integradora como diseñador y gestor de los procesos que se desarrollan en una organización, puede convertirse en un catalizador para que las organizaciones comprendan con profundidad y apliquen los últimos adelantos de las Tecnologías de la Información y las Comunicaciones (TIC) en función de sus objetivos de trabajo, vayan delante de la competencia y sobrevivan en este nuevo mundo informatizado.

Sobre la base de estos planteamientos y tomando en consideración además, los principales causantes de rechazo hacia la Informática por parte de los estudiantes de Ingeniería Industrial, se trabaja en la enseñanza de las asignaturas de Informática para los Ingenieros Industriales cubanos.

1. Introducción

El Instituto Superior Politécnico José Antonio Echeverría, situado en la Ciudad Universitaria del mismo nombre, es el centro rector para el estudio de las Ciencias Técnicas en Cuba. El mismo dirige los programas de diferentes carreras de Ingeniería y Arquitectura, a través de sus seis facultades: Arquitectura, Ingeniería Civil, Ingeniería Eléctrica, Ingeniería Industrial, Ingeniería Mecánica e Ingeniería Química.

En la Facultad de Ingeniería Industrial se estudian las carreras de Ingeniería Informática e Ingeniería Industrial, siendo la misión de esta última: *formar un profesional integral de alta calidad, comprometido con la patria, que satisfaga los requerimientos de la producción y los servicios en los inicios del siglo XXI, en los campos de la proyección, ejecución y dirección de los sistemas que garantizan la planificación, organización, regulación, control y calidad de los procesos de cualquier organización empresarial, estatal o social, con soluciones creativas, autóctonas, eficaces y eficientes. Contribuir de forma significativa al desarrollo sostenido y sustentable de la sociedad cubana y ser competitivo internacionalmente en el campo de la ingeniería industrial para lo cual hace suyas las aspiraciones más legítimas de trabajadores y estudiantes* [1].

La carrera de Ingeniería Industrial tiene dos especialidades: Industrial y Organización de Empresas, todas las asignaturas son comunes en los tres primeros años de la carrera y a partir del cuarto año se seleccionan los estudiantes que pasan al grupo de Organización de Empresas y el resto continúa en la especialidad de Industrial. Ambas especialidades se gradúan con el título de Ingeniero Industrial.

El actual Plan de Estudios de esta carrera se modificó en el año 1998 valorando los factores y condiciones cubanas, teniendo en cuenta los cambios ocurridos como consecuencia de la desaparición del campo Socialista, los procesos acelerados en los cambios tecnológicos como consecuencia de la 3ra. Revolución Industrial y los ajustes y formas de operar nuestra economía ante los fenómenos de la Globalización, la necesaria operación interrelacionada con una

economía de mercado y las condiciones que los marcos financieros en que actualmente el mundo opera, nos imponen.

Dicho Plan de Estudio es un diseño según las condiciones cubanas pero no ignora los aportes que en el sistema mundial de conocimientos aportan las Carreras de Ingeniería Industrial norteamericana y las versiones desarrolladas en diferentes países bajo el título de Ingenieros de Producción, Ingenieros de Sistemas, Ingenieros Económicos y otros enfoques utilizados en América y Europa.

La época actual obliga a velar porque la formación del Ingeniero Industrial tenga una elevada formación informática, que permita a este profesional hacer uso de las tecnologías de la informática y las comunicaciones, en su función integradora como diseñador y gestor de los procesos que se desarrollan en una organización, y que les permita además elevar su nivel de competitividad.

De ahí que la formación Informática del Ingeniero industrial de hoy tenga que ser muy elevada y con una perspectiva que le permita hacer un uso de ella, lo más eficiente y eficaz posible. No sólo debe garantizar que este opere como un "usuario avanzado" la computadora, sino que debe garantizar que este reconozca en las nuevas tecnologías las herramientas necesarias e imprescindibles para el éxito rotundo de la organización.

Por estas razones, se le ha prestado especial interés, al diseño de la disciplina Informática en la concepción del actual Plan de Estudios, con vistas a crear una base amplia y sólida en la utilización de estas tecnologías en este perfil profesional.

Es nuestro objetivo, mostrar con este trabajo, nuestra experiencia en la formación informática de los Ingenieros Industriales en sus dos especialidades, así como conocer sobre experiencias similares en otras universidades y dejar abierto un espacio para el debate sobre estas temáticas y su importancia en la sociedad actual.

2. Caracterización de la Disciplina Informática para el Ingeniero Industrial

La Disciplina de Informática en la formación de los Ingenieros Industriales cubanos está presente casi desde el comienzo de esta especialidad en nuestro país. La misma se ha ido perfeccionando a lo largo de los diferentes planes de estudio, pasando de 56 horas dedicadas a estas materias, en un inicio, a 314 horas en la actualidad.

Respecto a los planes anteriores, en la actual disciplina se realizaron cambios sustanciales tales como: emigrar hacia la plataforma Windows, incluir el tema de Redes y de Internet, así como exigir por parte de las asignaturas de perfiles terminales, el uso de la Informática. La disciplina aporta los elementos básicos para el trabajo en el campo de la computación: sistemas de operación, modelación algorítmica de problemas, lenguajes estructurados de alto nivel, bases de datos y sistemas de información.

Actualmente está formada por seis asignaturas distribuidas en los cuatro primeros años de la carrera, para un total de 314 horas. Esta disciplina dota a los estudiantes de una herramienta de trabajo como la computación, que le permite abordar tareas donde se procese gran volumen de información o que se realicen cálculos científicos complejos.

En general las actividades de esta disciplina requieren de actividades prácticas sistemáticas frente a la computadora. No obstante, los temas tratados y sistemas utilizados, poseen una teoría y fundamentos que deben ser explicados en conferencias. Además se requieren actividades prácticas en aula para la elaboración de modelos, algoritmos, análisis y diseño de sistemas, etc.

Respecto a la evaluación, ninguna de las asignaturas de la disciplina es evaluada a través de un examen final, por lo que se enfatiza en su evaluación sistemática y a través de trabajos extraclases. Dichos trabajos se relacionan con otras disciplinas de la carrera, principalmente con las que se estén impartiendo en el mismo año académico.

La bibliografía de la asignatura se soporta fundamentalmente en materiales electrónicos que se sitúan en la Red de la Facultad. Cada asignatura cuenta obligatoriamente con una carpeta en la Red, con carpetas independientes para los materiales de cada tema y en las cuales el estudiante puede encontrar artículos sobre el tema, ejercicios, indicaciones para las actividades de laboratorios, etc.

A pesar de la importancia de la materia y de la renovación de contenidos en la actual disciplina, es frecuente la falta de motivación de los estudiantes hacia las asignaturas de Informática. Según nuestros análisis, esto se debe fundamentalmente a que el estudiante no posee convencimiento de la posterior utilidad de los conocimientos que se le imparten, debido esto a que las temáticas enseñadas por Informática son muy poco utilizadas en el resto de las asignaturas. Por esta razón en la disciplina se está trabajado fuertemente en el principio de la *interdisciplinaridad*, que al decir de algunos autores, es importante, no porque la deseemos, sino porque es el hombre el que para comprender el mundo y transmitir sus enseñanzas lo ha subdivido y ha creado cuerpos de conocimientos para las diversas ramas del saber, pero en la vida se manifiestan como un todo interrelacionadas unas con otras [3].

Otro aspecto influyente en la falta de motivación de los estudiantes hacia las asignaturas de la disciplina y relacionado con el anterior es el contenido. Muchas veces las asignaturas tenían un contenido eminentemente técnico olvidándose del auditorio destino y por lo cual solían aburrir a los estudiantes. En este sentido se ha realizado un trabajo intenso, en lo que llamamos nosotros "darle la vuelta industrial al contenido informático", es decir que en todo momento los ejemplos que se utilicen sean de dicha especialidad.

Por otro lado también se trabaja en actualizar los materiales de enseñanza utilizados en las asignaturas, de forma que en el propio proceso de enseñanza el estudiante reconozca en la Informática una poderosa herramienta que lo apoya en su aprendizaje.

3. Caracterización de las asignaturas de la disciplina

Como mencionamos anteriormente la disciplina está formada por seis asignaturas:

- Fundamentos de Computación
- Programación I
- Programación II
- Elementos de Sistemas Informativos
- Introducción a los Sistemas de Información para la Dirección
- Sistemas de Información

A continuación describiremos brevemente cada una, y el trabajo que se ha realizado en ellas para lograr motivación en los estudiantes.

3.1. Fundamentos de Computación

Los objetivos principales de esta asignatura están enfocados, en primer lugar, hacia que los estudiantes adquieran una cultura sobre la computación de modo que sean capaces de mantenerse actualizados en este campo; y, en segundo lugar, a desarrollar las habilidades en los alumnos que les permitan utilizar la computadora como herramienta en la solución de problemas profesionales.

La asignatura cuenta con 70 horas distribuidas en 7 quincenas. Cada quincena se comporta de la siguiente forma:

Conferencia – se introduce el tema a abordar en la quincena. Las clases deben orientarse a mostrar qué es lo que se puede hacer desde cada una de las aplicaciones y no cómo se hace. Por ejemplo, en la clase de Word: Mostar una página Web hecha en Word, Mostrar una aplicación de una macro en Word.
Laboratorio – se trabaja con el Profesor de la asignatura en la resolución de problemas complicados
Laboratorio – se trabaja de modo independiente los ejercicios que el profesor orienta sobre el tema de la quincena.
Laboratorio – se evalúan las habilidades que el estudiante deberá haber alcanzado sobre el tema de la quincena.

Seminario – dos equipos expondrán sus trabajos referativos. Se evalúan tanto a los miembros del equipo como a los alumnos que intervengan en el debate.

Los temas que se abordan en la asignatura son: Arquitectura de una PC. Sistemas operativos. Tipos. Características. Ordenes de un sistema operativo de alto nivel. Redes. Tipos. Características. Formas de trabajo en red. Medios de comunicación. Tipos. Características técnicas. Modos de comunicación. Utilitarios para el trabajo con disco (antivirus, compactadores, etc.). Graficadores. Suit de Oficina. Internet. Características. Servicios que oferta. Creación de una pagina de WWW. Correo electrónico.

El objetivo del trabajo referativo, que realizan en cada quincena los estudiantes, es que los mismos ejecuten una pequeña investigación, a partir de la consulta de diferentes fuentes bibliográficas y que elaboren un informe final, dónde referencien las bibliografías consultadas. Los temas que se distribuyen entre los estudiantes, son temas novedosos en el campo de la Informática, relacionados con los contenidos de la asignatura, y se les busca además, su vinculación con la Ingeniería Industrial.

Esta es una de las asignaturas donde se ha trabajado con más fuerza el tema de la interdisciplinaridad. Uno de sus principales resultados está en la creación de un sitio Web con fines didácticos que contiene soluciones a algunos problemas de otras disciplinas de la Ingeniería Industrial, haciendo uso de las funcionalidades más avanzadas del Microsoft Excel. *ExcelWeb*, está siendo utilizado exitosamente por la asignatura Fundamentos de Ingeniería Industrial (FII) y por la propia asignatura Fundamentos de Computación.

Otro aspecto importante y que tributa en alto grado a la interdisciplinaridad es un sitio Web que se realizó también con la asignatura FII y que constituye el sitio oficial de la misma. Desde el mismo el estudiante accede a múltiples enlaces como Clases, Historia, Herramientas, etc. El sitio refuerza de forma indirecta tanto el aprendizaje de la Informática como el uso de la informática en el proceso de aprendizaje.

3.2. Programación I

Los objetivos de esta asignatura están enfocados fundamentalmente a que el estudiante sea capaz de algoritmizar un problema no muy complejo de la vida real y posteriormente implementarlo en un lenguaje de programación.

La asignatura cuenta con 70 horas distribuidas en 7 quincenas. Cada quincena se comporta de la siguiente forma:

Conferencia – se introduce el tema a abordar en la quincena.
Clase Práctica – se trabaja en la resolución de ejercicios de conjunto con el Profesor
Clase Práctica – se trabaja en la resolución de ejercicios de conjunto con el Profesor
Laboratorio – se trabaja con el Profesor de la asignatura en la implementación de los ejercicios resueltos en la clase práctica
Laboratorio – se trabaja de modo independiente en la resolución de ejercicios que el profesor orienta

Los temas que se abordan en la asignatura son: Análisis y diseño de algoritmos. Elementos de programación. Programación Visual. Ambiente de programación Visual.

La asignatura cuenta con un amplio banco de problemas. Además el profesor selecciona un ejercicio modelo que se comienza su resolución con el primer tema de la asignatura y se culmina con el último tema. El mismo se pone a disposición de los estudiantes y sirve de base para la resolución de la tarea práctica de la asignatura.

La evaluación de la asignatura constituye el elemento utilizado para la interdisciplinaridad. Los estudiantes resuelven problemas de otras asignaturas como Probabilidades y que dan solución a situaciones existentes en la vida real.

3.3. Programación II

Los objetivos de esta asignatura están enfocados a que el estudiante sea capaz de modelar un problema de la vida real a través de bases de datos, así como implementarlo en un sistema de gestión.

La asignatura cuenta con 60 horas distribuidas en 8 quincenas. Cada quincena se comporta de la siguiente forma:

Conferencia – se introduce el tema a abordar en la quincena.
Clase Práctica – se trabaja en la resolución de ejercicios de conjunto con el Profesor
Clase Práctica – se trabaja en la resolución de ejercicios de conjunto con el Profesor
Laboratorio – se trabaja con el Profesor de la asignatura en la implementación de los ejercicios resueltos en la clase práctica

Los contenidos de la asignatura son: Principios de SQL. Conceptos fundamentales relacionados con la gestión de grandes volúmenes de datos. Sistema de desarrollo de aplicaciones para la creación de aplicaciones front-end basadas en el modelo relacional.

La evaluación constituye el punto de contacto con la especialidad, ya que el estudiante resuelve problemáticas de diseño de bases de datos a partir de problemas que encuentra en las empresas dónde realiza sus prácticas de FII-2.

3.4. Elementos de Sistemas Informativos (ESI)

Esta asignatura tiene como objetivo preparar al futuro Ingeniero Industrial, posible directivo de una organización, para que participe en el proceso de producción de los softwares que necesita su entidad, desde el rol que le compete y pueda tomar decisiones acertadas.

La asignatura cuenta con 54 horas distribuidas en 8 quincenas. Cada quincena se comporta de la siguiente forma:

Conferencia – se introduce el tema a abordar en la quincena.
Laboratorio – se trabaja de forma independiente en la búsqueda de información, fundamentalmente en Internet, para la preparación de los seminarios.
Seminario– actividad evaluativa donde por equipo se exponen trabajos orientados y relacionados con la temática de la quincena.

Esta asignatura la primera vez que se impartió fue muy rechazada por los estudiantes, debido fundamentalmente a una mala concepción de la misma. A partir de esa experiencia se modificaron algunos de los contenidos y los objetivos que se debían alcanzar y los resultados fueron favorables.

En la siguiente tabla se muestra los resultados de la valoración de los estudiantes en algunos aspectos, en ambos cursos.

Aspecto	1er Curso	2do Curso
El sistema de evaluación	Negativo (Prueba intrasemestral)	Positivo (Seminarios)
Vinculación con la carrera	Negativo	Positivo
Conocimientos informáticos	Interesante	Positivo
Práctica	Poca	Poca
Búsquedas en Internet	Positivo	Positivo
Utilidad futura	Poca	Mucha

Tabla 1. Resultados de la encuesta aplicada a los estudiantes

Los contenidos abordados son: Los Sistemas Informativos (SI) y la Organización. La producción de software. Ingeniería del Software. Introducción a las metodologías de Análisis y Diseño. CASE's, La planificación de los procesos del software. Estimaciones para la planificación. Métricas. Estimaciones para la planificación. Métricas. Calidad del Software. Estándares de Calidad. Métricas de Calidad. Seguridad en el desarrollo de SI. Protección de la información. Seguridad en las Bases de datos

La asignatura utiliza una página Web desde donde el estudiante obtiene toda la documentación que necesita la asignatura. Además se incluyeron en la práctica profesional de ese año académico tareas concretas relacionadas con los conocimientos adquiridos y que permiten la vinculación de ESI con la especialidad.

3.5. Introducción a los Sistemas de Información para la Dirección (ISI) y Sistemas de Información (SI)

Estas son las asignaturas que en materia de Informática diferencian a los estudiantes de Organización de Empresas, ya que sólo se les imparte a los estudiantes que cursan esa especialidad a partir del 4to año de la carrera. Las mismas llevan dos cursos impartiéndose y en ellas se ha trabajado muy fuertemente en la concepción industrial y en la vinculación con otras asignaturas de la especialidad.

Estas asignaturas permiten al estudiante en primer lugar integrar los temas informáticos recibidos hasta el momento y en segundo lugar los provee de concepciones, métodos, etc sobre el uso de la informática y las comunicaciones en la gestión de la organización. Ambas tienen como objetivo que el estudiante se enfrente con los diferentes tipos de sistemas informativos que pueden coexistir en una organización. En ISI se aborda fundamentalmente las bases técnicas para la implementación de estos sistemas y en SI se abordan los sistemas informativos en sí: los diferentes tipos que pueden existir en una organización y el rol que desempeña cada uno de ellos en la misma.

ISI tiene 28 horas distribuidas en tres bloques de:

Conferencia - se introduce el tema y se orienta el seminario
Laboratorios - se trabaja de forma independiente en la búsqueda de información en Internet, profundizando en la temática abordada en la conferencia y obteniendo información para los seminarios
Seminario - se discuten las preguntas previamente orientadas y se evalúa el tema

La asignatura reserva dos actividades para realizar *visitas a empresas*. Esto fue algo novedoso, puesto en práctica por primera vez en esta asignatura, y resulta una forma muy efectiva de vincular la disciplina con la especialidad del estudiante, ya que el mismo puede comprobar en la práctica la relación de la informática con la gestión empresarial. Estas visitas, realizadas siempre a empresas líderes en el uso de la informática, tienen el objetivo de mostrar al estudiante, cómo la informática es usada para gestionar la empresa y cómo se vincula con los procesos claves de la misma.

Las temáticas abordadas en ISI son: Información como recurso estratégico. Concepto de negocio de los Sistemas de Información (SI). Hardware de los SI. Software de los SI. Administración de recursos de Información. Telecomunicaciones. La nueva arquitectura de la Información.

SI tiene 32 horas distribuidas en cuatro bloques de:

Conferencia - se introduce un tema y se orienta el seminario
Clase práctica - se resuelven casos de estudio sobre el tema
Laboratorio - se trabaja de forma independiente en la búsqueda de información en Internet, profundizando en la temática abordada en la conferencia y obteniendo información para los seminarios
Seminario - se discuten las preguntas previamente orientadas y se evalúa el tema

Los contenidos de la asignatura son: Conceptos básicos sobre Información. Información como recurso estratégico. Sistemas de Información. Tipos de Sistemas de Información. Enfoques contemporáneos sobre sistemas de información. Retos de los sistemas de Información. Sistemas de Información integrados. Características. Tendencias. Sistemas de información para el control de gestión. Sistemas de Información Inter organizacionales.

La asignatura toca temas que son muy utilizados por otras asignaturas de la especialidad como los sistemas ERP y por tanto facilita la interdisciplinariedad. Entre las acciones que se han realizado para apoyar este principio, están las conferencias de conjunto entre SI y otras asignaturas, donde el ponente es algún especialista externo a la Universidad que aborda una temática de interés para ambas asignaturas.

En ambas asignaturas, el seminario es la principal forma de evaluación. El mismo tiene por objetivos, primero que el estudiante profundice en

el tema, segundo que se entrene en la búsqueda de información en Internet y por último, que se entrene en la elaboración de informes. Todos los seminarios tienen un grupo de preguntas generales, dirigidas a cumplimentar el primero de los objetivos, y un grupo de preguntas a desarrollar por equipos, que permiten el trabajo en grupo, entre los estudiantes.

Ambas asignaturas utilizan un sitio Web que permite a los estudiantes:

- Acceder a toda la documentación que necesitan: clases, seminarios, presentaciones power point de las clases, etc.
- Acceder de forma fácil y rápida a sus evaluaciones en los seminarios.
- Acceder a una biblioteca con material complementario: artículos, noticias, libros digitales, etc.

El sitio cuenta con una herramienta administrativa que permite a los profesores:

- Publicar noticias docentes sobre la asignatura
- Montar toda la asignatura: clases, seminarios, material complementario, etc.
- Publicar las evaluaciones de los estudiantes y realizar un pequeño seguimiento de los mismos.

Figura 1. Interfaz del sitio Web para las asignaturas

El acceso a dicho sitio se realiza a través de la Intranet de la Facultad y está disponible las 24 horas, todos los días.

Los resultados obtenidos en ambas asignaturas han sido muy favorables. Se ha encuestado a los 40 estudiantes, que entre los dos cursos, han recibido la asignatura.

En el caso específico de la asignatura ISI, los principales aspectos positivos fueron:

- total vinculación de los temas tratados con su especialidad (100 % de los estudiantes)
- total satisfacción con las visitas a empresas (100 % de los estudiantes)
- total satisfacción con los seminarios, como forma de evaluación

El principal señalamiento realizado estuvo en la ausencia de un texto bibliográfico, disponible para los estudiantes.

En el caso de la asignatura SI los principales resultados han sido:

En primer lugar resultó muy positivo que el 100 % de los estudiantes consideran importante o muy importante la asignatura dentro de su plan de estudio; el 64 % considera que la misma contribuye muy positivamente en su futuro desempeño profesional y que el 84 % se siente capaz de participar medianamente en soluciones de sistemas de información en sus organizaciones.

Los principales aspectos positivos resultaron:

- Actualidad, novedad y vinculación con la especialidad, de los temas tratados
- El sistema evaluativo, tanto los seminarios como la mesa redonda
- La conferencia con el especialista de la empresa de software
- La calidad de las clases
- Los accesos a Internet

Como aspecto negativo principal, aunque se mantiene el señalamiento respecto a la bibliografía, se señala que no se ven en la práctica ningún tipo de sistema informativo. Y este es precisamente uno de los aspectos donde todavía se debe trabajar con profundidad.

4. Recursos Humanos en la Disciplina Informática

Un aspecto importantísimo en toda la concepción planteada son los profesores que imparten las asignaturas y en nuestro caso esto es una fortaleza.

Existe un colectivo dedicado a impartir las asignaturas de Informática para la Ingeniería Industrial, así como a realizar investigaciones en ese perfil. La jefa del colectivo tiene doble ingeniería: informática e industrial. Esto ha permitido que en todas las asignaturas se puedan utilizar ejemplos prácticos de ingeniería industrial, además que se ha trabajado muy fuertemente en que la concepción de cada asignatura sea para los Industriales. El resto del colectivo está compuesto por profesionales todos de diversas especialidades: informáticos, industriales, estadísticos, automáticos y matemáticos. Esta característica multidisciplinaria, hace de los análisis, discusiones muy provechosas.

Estratégicamente se ha trabajado por fortalecer el *trabajo en grupo* dentro de la disciplina y se exige que cada concepción sea previamente discutida en el colectivo sin dejar espacios a la espontaneidad en este sentido. Además se trabaja porque todos los miembros se *capaciten* en temas nuevos que sean afines.

5. Conclusiones

El Ingeniero Industrial actual requiere una formación informática elevada y en el ISPJAE se ha trabajado fuertemente en este sentido. Se han tomado en consideración tres aspectos esenciales para lograrlo: la vinculación de las asignaturas informáticas con la especialidad de Ingeniería Industrial, el trabajo en grupo y la atención al recurso humano.

Aunque no se tienen resultados formales de encuestas, ni ninguna otra herramienta de retroalimentación a nivel de disciplina, a nivel de las asignaturas si se han realizado y los resultados demuestran una amplia satisfacción por parte del estudiante hacia estas asignaturas. Además hemos podido realizar comparaciones discretas, de nuestras experiencias con otras similares en universidades latinas y hemos comprobado la validez de nuestro programa y de nuestras acciones.

Entre las recomendaciones para el trabajo futuro se encuentran, en primer lugar seguir trabajando en el principio de interdisciplinaridad en la disciplina, estimulando a los profesores del resto de las especialidades, que realicen acciones de este tipo en sus asignaturas, que tributen a la formación Informática de los Ingenieros Industriales. En segundo lugar, establecer mecanismos formales de retroalimentación sobre la formación Informática de estos especialistas, que permitan realizar una valoración más efectiva sobre los cambios que se han introducido. Y en tercer lugar seguir profundizando en los estudios sobre la temática de los Sistemas Informativos, por considerarla de gran importancia para que el Ingeniero Industrial pueda hacer un uso más eficiente y eficaz, de las tecnologías de la Informática y las Comunicaciones en la empresa.

Referencias

[1] MES. Ministerio de Educación Superior. Plan de Estudio del Ingeniero Industrial. Versión C´. 1998, Cuba.

[2] Fiallo, Jorge P. ¨ La interdisciplinariedad en la escuela: de la utopía a la realidad ¨. Curso del evento Internacional Pedagogía 2001, Cuba.

[3] Gil Pechuán, Ignacio. Sistemas y Tecnologías de la Información para la gestión, Mc Graw-Hill, 1997

[4] Stuart Cárdenas, Mavis Lis. Diagnóstico de la utilización de las NTIC para la Ingeniería Industrial. Diseño de una concepción de utilización. Trabajo de Diploma, Departamento Ingeniería Industrial, ISPJAE, 2001, Cuba

Informática para profesionales de la Geología: docencia, aprendizaje y práctica

María Vaquero Domínguez
Dpto. de Informática
Universidad de Salamanca
37008 Salamanca
e-mail: mvaquero@usal.es

Roberto Therón Sánchez
Dpto. de Informática
Universidad de Salamanca
37008 Salamanca
e-mail: therón@usal.es

Resumen

En este documento se aborda la situación de la docencia de la Informática en los estudios de Licenciatura e Ingeniería Geológica. En concreto, se analizan los planes de estudio de los centros que imparten estas titulaciones a lo largo de la geografía española, para observar la disparidad entre los que imparten y los que no ofrecen contenidos informáticos dentro de dichas carreras.

Dichos conocimientos son escasos y se imparten a través de asignaturas cuatrimestrales y de carácter optativo, no ajustándose a la demanda de conocimientos informáticos por parte de las empresas a los profesionales de la Geología.

La Universidad es un elemento de la sociedad cambiante actual, y como tal, debe atender a sus necesidades y demandas, tanto educativas como profesionales.

Así, se concluye que la formación informática que reciben los alumnos de estas titulaciones, es insuficiente, dadas sus necesidades debidas al carácter técnico de la titulación y la complejidad de la instrumentación geológica.

1. Introducción

Siendo la Informática la ciencia que más rápidamente ha adoptado la sociedad, los miembros de ésta necesitan recibir una formación adecuada para no ser superados por la tecnología. A la vista de tal situación, la Informática como asignatura universitaria está presente en la práctica totalidad de las titulaciones. También es cierto que en gran parte de los casos, tal presencia sólo es optativa, y generalmente reducida a una Informática Básica.

Sin embargo, hay algunas áreas técnicas que requieren de sus licenciados un manejo de herramientas y entornos informáticos que van más allá del manido "conocimiento a nivel de usuario". Este es el caso de los Licenciados en Geología e Ingenieros Geólogos.

El resto del artículo se organiza como sigue: en la segunda sección se analiza la situación de la Informática en el plan de estudios de Geología.

En el tercer apartado se analiza como caso de estudio una situación de trabajo real a la que se puede enfrentar un recién licenciado. Encontrándose por una parte con numerosas dificultades a la hora de elaborar un mapa topográfico del océano, y por otra la inexperiencia total en el diseño y manipulación de una base de datos oceanográfica.

En el cuarto apartado se propone un programa docente que podría solventar la situación actual.

En la quinta sección se ilustra la relación entre Informática y Geología con una herramienta informática desarrollada para la docencia e investigación en la titulación de Geología [7].

Finalmente se exponen las principales conclusiones a las que ha llegado este estudio.

2. Informática como materia optativa

Tras un análisis de los planes de estudio de los centros que imparten estas titulaciones a lo largo de la geografía española [9], se observa que la Informática como asignatura específica dentro del plan de estudios de Licenciado en Geología sigue sin formar parte de las materias optativas que pueda cursar un alumno. Aunque existen algunos centros donde sí se imparte, llama la atención el

carácter optativo de la asignatura. Otro factor a tener en cuenta es la poca importancia que recibe la parte práctica, el número de créditos prácticos como máximo es de 2,5.

En la Tabla 1 se observan los centros en los que se imparte una formación informática a través de una serie de asignaturas y el número de créditos tanto teóricos como prácticos. Todas las universidades que imparten la titulación "Ingeniería Geológica" tienen en sus planes de estudio asignaturas optativas u obligatorias relacionadas con la Informática (Tabla 2).

Análisis de la información (Tabla 1 y 2):

- De los datos presentados cabe destacar que sólo en el 25% de los centros que imparten la Licenciatura en Geología incluyen materias optativas relacionadas con la Informática, mientras que dichas materias se incluyen en el 100% de los centros que imparten la Ingeniería[1]
- Es de destacar que en La Universidad Complutense de Madrid la asignatura "Aplicaciones Informáticas en Geología" no tenga créditos prácticos, cuando su programa se centra sobre todo en la realización de prácticas.
- Otro dato de interés es el carácter obligatorio de la asignatura "Aplicaciones Informáticas en Geología" dentro de la titulación de Ingeniería Geológica, Universidad de Salamanca. Constituyendo un buen referente a la hora de valorar la importancia de la formación informática.
- Es significativo que en la Universidad de Salamanca el número de créditos de la asignatura "Aplicaciones Informáticas y Programación Básica" sea 6, superando en 1,5 al resto de asignaturas impartidas a lo largo de la geografía española.
- La mayoría de las materias de Informática se imparten en el segundo ciclo, lo cual retrasa el aprendizaje de los alumnos e impide una formación continua.
- Analizando los programas de las asignaturas, llegamos a la conclusión de que en los estudios de Geología, al igual que ocurre en otras titulaciones como Diplomado en Turismo [6], la docencia de Informática se reduce a una informática básica, que no se ajusta a las necesidades de los alumnos. Los conocimientos que se adquieren se basan en Ofimática y están limitados a la plataforma Windows. Dado el carácter técnico de los trabajos en Geología y el gran volumen de datos que se maneja, este sistema operativo resulta insuficiente, pues muchas de las aplicaciones se desarrollan para trabajar en estaciones de trabajo con sistemas de la familia UNIX.

En contraste con la pobre formación, existe en el mercado una gran cantidad de herramientas informáticas desarrolladas para el ámbito geológico. Entre las más destacadas están las de cálculo de volúmenes, tratamiento de imágenes, cálculo de gradientes, cartografía, modelado y simulación de sistemas geológicos, etc. En los programas actuales no se recoge el uso de estas herramientas y tampoco se proporciona a los alumnos los conocimientos necesarios para elaborar sus propios programas.

Universidad Complutense de Madrid	Descripción de la Asignatura	Aplicaciones Informáticas en Geología
	Curso	Quinto
	Clase	Optativo
	Carácter	Cuatrimestral
	Créditos	4,5
	Práctica	0
Universidad de Salamanca	Descripción de la Asignatura	Aplicaciones Informáticas y Programación Básica
	Curso	Segundo
	Clase	Optativa
	Carácter	Cuatrimestral
	Créditos	6
	Práctica	1,5

Tabla 1. Licenciatura en Geología

[1] No se considera la Universidad Politécnica de Valencia, puesto que no se puede cursar la titulación completa.

Universidad de Alicante	Descripción de la Asignatura	Diseño de Ingeniería por Ordenador
	Curso	Sin curso
	Clase	Optativo
	Carácter	Cuatrimestral
	Créditos	4,5
	Práctica	1,5
Universidad de Barcelona	Descripción de la Asignatura	Programación Gráfica y Técnicas Visuales
	Curso	Quinto
	Clase	Optativa
	Carácter	Cuatrimestral
	Créditos	4,5
	Práctica	2,5
Universidad Complutense de Madrid	Descripción de la Asignatura	Aplicaciones Informáticas en Geología
	Curso	Quinto
	Clase	Optativa
	Carácter	Cuatrimestral
	Créditos	4,5
	Práctica	0
Universidad Politécnica de Cataluña	Descripción de la Asignatura	Programación Gráfica y Técnicas Visuales
	Curso	3 ó 4
	Clase	Optativa
	Carácter	Cuatrimestral
	Créditos	4,5
	Práctica	2,5
Universidad de Salamanca	Descripción de la Asignatura	Aplicaciones Informáticas en Geología
	Curso	3
	Clase	Obligatoria
	Carácter	Cuatrimestral
	Créditos	4,5
	Práctica	1,5

Tabla 2. Ingeniería Geológica

Si bien, es verdad que en ciertas asignaturas troncales u optativas como "Teledetección" o "Sistemas de Información Geográfica", se imparte una base informática necesaria para comprender la temática; sirva como ejemplo que dentro del plan de estudios de la Ingeniería Geológica impartida en la Universidad de Alicante se encuentra la asignatura "Técnicas de cartografía avanzada" donde se incluyen temas como Sistemas de Información Geográfica, Modelos Digitales del terreno, etc, utilizando un soporte informático. En contraposición a este tipo de asignaturas se encuentran la mayoría de las impartidas en la titulación, en las que el aprendizaje se basa en clases magistrales sin inclusión de nuevas tecnologías.

La Universidad es un elemento de la sociedad actual, y como tal, debe atender a sus necesidades y demandas, tanto educativas como profesionales. No debe limitarse a una formación meramente académica en la que primen los conocimientos intelectuales. Uno de sus objetivos es ir más allá y lograr una educación permanente que favorezca el autoaprendizaje y facilite la incorporación en el mundo profesional. Para ello el alumno debe adquirir conocimientos y capacidades más acordes con la situación cambiante que atraviesa nuestra sociedad; en este sentido, las nuevas tecnologías han producido numerosos cambios. Dicha alteración está patente en muchos aspectos, por ejemplo, en la demanda de conocimientos informáticos por parte de las empresas a los profesionales de la Geología.

El uso de nuevas tecnologías y una sólida formación informática ayudan al alumno en su proceso de aprendizaje. Por esta razón es evidente la necesidad de diseñar una titulación con unos planes de estudio flexibles, acorde con el nuevo perfil, y cuyo objetivo sea una formación continua [4].

Es precisamente esta falta de conocimientos y la necesidad del uso de nuevas tecnologías lo que hace proponer una formación más acorde con las necesidades de los alumnos de Geología, que disponga de nuevos métodos de aprendizaje, basados en tecnologías de la información y la comunicación Esta misma situación se ha detectado en otras titulaciones y, similarmente, se han realizado análisis con conclusiones de parecida índole, entre otras: nueva programación

aplicada a la Licenciatura de Matemáticas [5] o una mejora en general de la enseñanza informática en diferentes titulaciones [1].

3. Caso de estudio

Como ejemplo que ilustra la necesidad de una mayor formación informática, se muestra a través de un caso práctico, las dificultades e inconvenientes con los que se encuentran los alumnos, a la hora de elaborar un mapa batimétrico: cartografía de profundidades oceánicas para determinar la topografía del fondo marino [8]. Para ello, se parte de datos almacenados en bases de datos oceanográficas, con información de diferentes aspectos geológicos (Figura 1). Es patente la inexperiencia total de los alumnos al manejar dicha información y, entre otros, se enfrentan a los siguientes problemas:

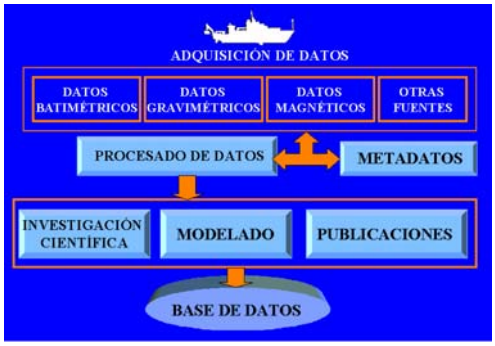


Figura 1. Generación de una base de datos oceanográfica

Figura 2. Contenido de la base de datos

1. Los equipos de adquisición de datos utilizan como medio de almacenamiento plataformas que requieren gran potencia y seguridad, como plataformas UNIX. Los alumnos no están familiarizados con ellas y no conocen su funcionamiento.
2. La gran cantidad de datos recopilados (datos brutos) necesitan volcarse en dispositivos de almacenamiento secundario como cintas o discos mágneticos. Los alumnos no conocen los comandos básicos para almacenar los datos y posteriormente recuperarlos.
3. Es necesario extraer la información correspondientes a la zona a representar, para ello, se debe manejar herramientas gráficas que limitan los datos al área estudiada. Los alumnos carecen de experiencia en el manejo de dichas herramientas.
4. Dada la cantidad de información recopilada, existen numerosos datos no válidos, debidos a errores de adquisición, interferencias con aparatos electrónicos, adversidades climáticas, etc. Estos datos deben ser corregidos y filtrados. Los alumnos desconocen las herramientas que les pueden facilitar su corrección y tampoco son capaces de crear sus propios filtros con lenguajes de programación como Matlab.
5. Una vez procesados los datos es interesante ver una representación gráfica de ellos. Para la elaboración de productos como modelos digitales del terreno, y cartografía temática.

6. Finalmente los datos filtrados y corregidos, deben ser exportados a una base de datos donde se almacenan para futuras consultas. Como se observa en la Figura 2, engloban información de la topografía del terreno (datos recogidos con instrumentación específica), las campañas oceanográficas realizadas, y los resultados obtenidos (publicaciones científicas). Los alumnos se encuentran con la dificultad de que carecen de conocimiento sobre sistemas de bases de datos. Por lo tanto, no pueden crear sus propias bases de datos, ni tampoco tienen la posibilidad de introducir, actualizar o recuperar información de las existentes.

4. Propuesta de un programa docente

En este apartado se propone un programa de una asignatura para dotar a los alumnos, no sólo de conocimientos teóricos de la Informática, sino conseguir que adquieran unas capacidad del manejo de equipos informáticos, que les puedan ser útiles para su formación geológica y les faciliten la integración en el mundo laboral o en posteriores estudios o investigaciones.

Los objetivos iniciales de este programa, que se imparte en la Licenciatura o Ingeniería Geológica son:

- Proporcionar al alumno unos conocimientos básicos, en el ámbito de usuario, de los paquetes desarrollados específicamente para trabajos en Geología.
- Facilitarle la compresión de la terminología usada habitualmente en los ámbitos informáticos.
- Capacitar para poder responder con herramientas propias a problemas particulares.

La asignatura se centraría principalmente en la utilización de la plataforma UNIX para el procesado , análisis, modelado y visualización de datos geofisicos. Conjuntamente se recibiría una formación de los comandos básicos de UNIX para el manejo de ficheros. La asignatura incluiría:

- Introducción al manejo de bases de datos públicas con información de batimetría, terremotos, propiedades físicas del océano, y otros datos geofísicos.
- Introducción al procesamiento y visualización de datos geológicos y oceanográficos. Se podría utilizar el paquete GMT gratuito.
- Introducción a aspectos computacionales para el procesado de datos sísmicos utilizando paquetes de programas estándar y públicos.
- Introducción a la programación de alto nivel para manipulación de datos geológicos y modelos geofísicos. Matlab, C, Visal Basic, etc.

Es interesante destacar en este último punto, que actualmente Matlab es quizás uno de los entornos informáticos líderes en aplicaciones científicas. Por ello sería adecuado y muy fructífero que los alumnos aprendieran a utilizar Matlab a través de ejercicios específicos relacionados con la Geología, como por ejemplo:

- Matrices de datos geológicos.
- Límites, series, series de Fourier y de predicción de terremotos.
- Diferenciación parcial y flujo del calor
- Múltiple integración y desintegración de hidratos.
- Integración y refracción sísmica.
- Número complejos y geofísica electromagnética.
- Ecuaciones diferenciales y velocidad sísmica.

Tanto los contenidos como objetivos de la propuesta planteada son similares a los correspondientes a asignaturas relacionadas con la Informática en la Licenciatura de Geología impartidas en otros países [2].

Hay que ser consciente de que adquirir estas capacidades conlleva enfrentarse a las siguientes dificultades:

- Los alumnos deben tener una sólida formación matemática, que se adquiere en el primer año de carrera.
- Existe una limitación de horarios patente en todas los centros y normalmente nos encontramos con la problemática del límite de equipos por alumno y de disponibilidad de licencias.
- Dada la extensión de conocimientos se debería impartir en una asignatura anual o en dos cuatrimestrales.

5. Herramienta informática para la docencia

Para finalizar y para que ilustre la íntima relación entre Informática y Geología, se muestra un ejemplo del uso de una herramienta (PaleoPlot, http://carpe.usal.es/~paleotools) desarrollada explícitamente para el manejo por expertos geólogos y alumnos de la Universidad de Salamanca

Estudio del contexto:

La variabilidad climática a gran escala ha estado regulada por cambios periódicos en la configuración astronómica de la geometría de tierra-sol. Durante los últimos cientos de miles de años, los parámetros astronómicos han determinado el clima del planeta que ha oscilado del estado glacial al interglacial y viceversa en función de la configuración astronómica existente en cada momento [3].

Los registros geológicos marinos son una de las pocas evidencias con las que cuentan los geólogos para reconstruir de manera completa estos cambios climáticos.

Los alumnos de la Licenciatura o Ingeniería Geológica estudian estos cambios en diversas asignaturas como "Paleoceanografía", "Prospección y explotación oceánica", etc, y cómo afectan a las especies de nanofósiles características de cada edad. Normalmente tienen que familiarizarse con gráficas en las que se reflejan las variaciones en el placton marino en relación a las variaciones cíclicas astronómicas.

Para facilitar el aprendizaje de estas asignaturas es interesante que los alumnos manejen la herramienta PaleoPlot (Figura 3), diseñada para monitorizar, integrar y analizar series paleoclimáticas. La herramienta actualmente se utiliza con datos de microfósiles pero podría aplicarse a cualquier series de datos y otros campos como la Estatigrafía.

Conceptos del diseño:

Se pensó en diseñar esta herramienta e incluir a los alumnos en el proyecto teniendo en cuenta unos puntos de partida:

- Es independiente del sistema operativo, puesto que no todos los alumnos están familiarizados con entornos como UNIX.
- Dispone de una arquitectura fácilmente escalable, proporciona una forma de introducir una alimentación de los usuarios y de incorporar nuevas técnicas.
- La interfaz gráfica es sencilla de utilizar e intuitiva, los alumnos de Geología no suelen tener mucha práctica en el manejo de ordenadores y tienen dificultades con los distintos formatos de ficheros y su creación.
- Se basa en un entorno completamente personalizado, para evitar la necesidad de programas adicionales que realicen por ejemplo el cambio de formato de los números o los colores de los gráficos.
- Posee un eficiente manejo de información como corrección de datos incongruentes y representación de datos efectivas .

Todas las opciones son accesibles desde la barra de menús, en la parte superior de la ventana principal. El resto de la herramienta, está formada por multitud de ventanas y de cuadros de diálogo que dependen del estado de la operación y del análisis realizado (Figura 3 y 4).

Objetivos alcanzados:

- Los alumnos asimilan que el clima cambia como resultado de la interacción de la variabilidad astronómica que regula los grandes ciclos climáticos y los cambios bruscos de tipo milenario.
- Facilita el trabajo de los paleoceanógrafos ayudándoles durante todo el proceso: partiendo de la integración y correlación de diferentes bases de datos, continuando con análisis interactivo y concluyendo con resultados e impresiones.
- El uso de esta herramienta permite que aprendan de un forma deductiva, relacionando conocimientos.
- La interfaz gráfica de la herramienta ayuda a tener una representación que amplia la visión de la información, ayudando al alumno a asimilar conocimientos (Figura 4).

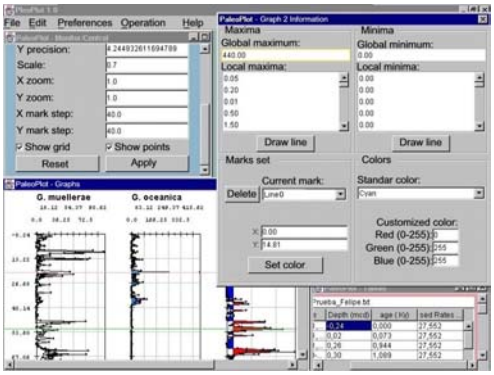


Figura 3. Interfaz gráfica de la herramienta PaleoPlot

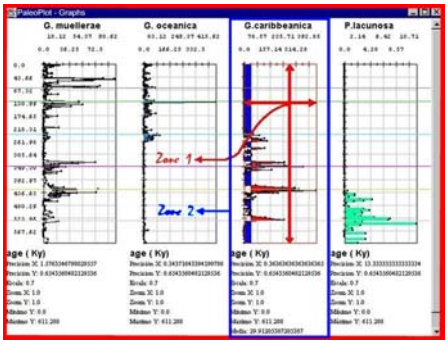


Figura 4. Ventana de análisis de gráficas correspondientes a series temporales

6. Conclusiones

Es importante resaltar que a universidad no debe limitarse a impartir aquellos contenidos básicos de usuario en el campo de la Informática. Es necesario ir más allá e impartir contenidos más específicos relacionados con Geología para el desarrollo de su futura profesión, así como manejar herramientas específicas para facilitar su trabajo y la asimilación de conocimientos . En este sentido hay que reseñar que incorporar nuevas tecnologías facilita el poder seguir un enfoque constructivista del aprendizaje a través de una comprensión activa.

Referencias

[1] Alonso, P., García, F., y Mollá, R. *Mejoras en el aprendizaje de la informática en otras escuelas universitarias*. JENUI'99. V Jornadas sobre la Enseñanza Universitaria de la Informática, Zaragoza, octubre, 1999.

[2] Charman, D., y Elmes, A. *Computer based assessment (Volume 2) Case studies in Science & Computing*. SEED Publication. University of Plymouth, 1998.

[3] Imbrie, J., Hays, J.D., Martinson, D.G., McIntire, A., Mix, A.C., Morley, J.J., Pisias, N.C., Prell, W.L., y Shackleton, N.J. *The orbital theory of Pleistocene Climate. Support from a revised chronology of marine $\delta^{18}O$record*. Milankovitch and Climate, Reidel, Dordrecht:269-305. 1984.

[4] García, J. *Las directrices Propias de las Titulaciones de Informática. La urgente necesidad de cambio.* JENUI'2002. VIII Jornadas sobre la Enseñanza Universitaria de la Informática, Cáceres, julio 2002.

[5] Llorens, F., Molina, R., Rizo, R. y Satorre, R. *La Programación en la Licenciatura en Matemáticas*. JENUI'99. V Jornadas sobre la Enseñanza Universitaria de la Informática, Zaragoza, octubre 1999.

[6] Ministral , M., y Majó, J. *La informática en los estudios de turismo.* TURITEC. Turismo y tecnologías de la información y las comunicaciones. Málaga, 1999.

[7] Therón, R., Flores, J.A., Sierro, F.J., Vaquero, y M., Barbero., F. *PaleoPlot: A tool for the Análisis, Integration and Manipulation of Time-series Paleorecords.* Proceedings of IEEE International Geoscience and Remote Sensing Symposium. Canadá 2002.

[8] Vaquero, M., Acosta., J., Muñoz, A., Palomo, C. Ecosondas multihaz de alta resolución. Aplicaciones a la hidrografía-oceanografía litoral. Geotemas. Volumen 1(4), 2000.

[9] Web. Páginas Web de las distintas universidades españolas.

El comercio electrónico en los estudios empresariales

Pedro L. Pérez Serrano
QUERCUS Software Engineering Group
Dpto. de Informática
Universidad de Extremadura
10071 Cáceres
e-mail: plperez@unex.es

Lourdes Moreno Liso
Dpto. de Derecho Privado
Universidad de Extremadura
06071 Badajoz
e-mail: lmoreno@unex.es

Resumen

Se ha demostrado desde distintos ámbitos de estudio que el comercio electrónico [1] forma parte de la vida empresarial. Su introducción en las empresas se está produciendo paulatinamente, sin prisas pero sin pausas, ofreciendo a quien se acoge a este tipo de negocio un mayor campo de oferta y una mejor atención a la demanda, es decir, a sus clientes.

Este artículo analiza la experiencia docente realizada en el segundo cuatrimestre del curso 2001/2002, en la Facultad de Ciencias Económicas y Empresariales de Badajoz, protagonizado por los departamentos de Derecho Privado e Informática de la Universidad de Extremadura. Una primera reflexión nos lleva a evaluar la oportunidad de incluir en los Planes de Estudio de las titulaciones de estudios empresariales una asignatura específica, multidisciplinar, denominada Comercio Electrónico, en la que se expliquen contenidos informáticos, jurídicos y económicos y cuyo objetivo final sea ofrecer una noción completa sobre la materia objeto de estudio.

1. Presentación

Poco a poco las empresas están implantando en Internet sus propias tiendas virtuales[1], en las cuales se reflejan sus actividades, como una manera adicional de obtener beneficios al *mínimo coste* sin renunciar al comercio tradicional. Su presencia en Internet es cada vez más frecuente, lanza una mejor imagen empresarial y ofrece numerosas ventajas y servicios al cliente. Pero tener una tienda virtual no sólo consiste en publicar una serie de páginas web; el empresario debe conocer y afrontar los costes que supone y la responsabilidad que implica trabajar en un entorno virtual.

Por ello, creemos que las titulaciones de empresariales deben adaptarse a los cambios que experimentan las empresas y actualizar sus conocimientos, siempre desde un punto de vista interdisciplinar. La necesidad de establecer relaciones con áreas de conocimiento de otros departamentos implicados en estas titulaciones, se ha revelado necesaria e imprescindible si queremos explicar la asignatura de forma completa o plena. El fomento o las relaciones interdisciplinares es en muchos casos necesario cuando se trabaja en una misma titulación, sobre todo cuando se tratan temas comunes, para de esta forma complementar y dar una visión mucho mas amplia del tema, y de este modo no abarcar uno el campo del otro.

Los continuos y rápidos cambios que se producen [2] dentro del campo de la informática, junto con su implicación directa en el mundo empresarial, nos obligan a realizar de forma constante revisiones de los planteamientos docentes de las asignaturas de informática, al igual que de las asignaturas en las que se trate algún tema relacionado con la informática de forma directa o indirecta. No debemos olvidar el entorno social que nos rodea para la formación de nuestros alumnos, así como las necesidades laborales del momento. Tampoco hay que dejar al margen factores como el nivel de conocimiento con el que llegan los alumnos, la velocidad a la que evolucionan las nuevas tecnologías (a las que casi siempre la

[1] Hablar de tiendas virtuales es hablar de comercio electrónico.

Universidad tarda tiempo en adaptarse) y sus implicaciones, los medios con los que contamos para impartir la docencia, así como las exigencias actuales del mercado laboral.

Se expondrán a continuación los objetivos, la metodología utilizada para la realización de las clases, así como el temario explicado en las mismas, los resultados y las conclusiones.

2.Objetivos

Los objetivos que se fijaron a la hora de realizar esta experiencia eran bastante precisos. En líneas generales, resultaba innovador el planteamiento de impartir una clase mixta derecho-informática, ofreciendo un punto de vista completo de la materia. En el resto de universidades el comercio electrónico se estudia a través de cursos de doctorado, cursos de especialización, etc, pero no de forma interdisciplinar ni como asignatura propiamente dicha[2].

Por otra parte se pretendía analizar el grado de interés mostrado por los alumnos sobre el reto que tiene las empresas de abrirse a nuevos mercados (en este caso virtual) utilizando las nuevas tecnologías.

Por lo tanto, los objetivos planteados eran varios:

- Introducir al alumno, como futuro empresario (dependiendo de la titulación), en el mundo del Comercio Electrónico y todo lo relacionado con él.

- Motivar la utilización de Internet como una forma alternativa de realizar actividades cotidianas tales como reservas de entradas, compras electrónicas, consultas, reclamaciones, una vez que se les ha explicado todos los conceptos de forma clara.

- Orientarles, como consumidores y usuarios, sobre la forma más correcta y segura de realizar compras a través de la red, en especial cómo y cuándo deben de facilitar sus datos personales, incluido su número de tarjeta de crédito, al vendedor virtual, con la garantía de que sean tratados de forma confidencial.

- Explicar de forma sencilla los conceptos jurídicos e informáticos que deben tener en cuenta a la hora de crear su propia tienda virtual. Dependiendo del tipo de negocio que se desee implantar, se indica al alumno la posibilidad de que sea él mismo quien cree su propia empresa en la red, con las herramientas informáticas que tiene a su disposición, sin necesidad de contratar a un informático.

Una vez impartidas estas clases, el alumno ha adquirido los conocimientos jurídicos e informáticos necesarios para entrar en este nuevo mundo que es el de las tiendas virtuales.

Además de los objetivos puramente académicos, nos interesaba evaluar el interés despertado por parte de los alumnos ante el Comercio Electrónico.

3.Metodología docente

La lección sobre comercio electrónico se desarrolló en las titulaciones de Diplomatura en Ciencias Empresariales, Licenciatura en Administración y Dirección de Empresas y Diplomatura en Relaciones Laborales. Las asignaturas implicadas fueron las de *Derecho*

[2] Dentro de las universidades a las que hemos tenido acceso vía web, existen algunas cono la UNED en la cual se imparte una asignatura, Derecho Informático I y II, pero la hacen a través de programas de enseñanza abierta o programas de formación de profesorado, pero no incluidas en los Planes de Estudio como una asignatura propia. Sí hemos de mencionar que en la Universidad de Girona, por ejemplo, en la Facultad de EE. y Empresariales, dentro de la asignatura de *Informática Aplicada a la Gestión de la Empresa*, de la Diplomatura en Empresariales, el tema 5 se dedica al Comercio Electrónico, haciendo referencia, en uno de los apartados, a los aspectos legales.

Mercantil, 2º curso, de cada una de las titulaciones mencionadas, así como la de *Informática Aplicada a la Gestión de la Empresa* en el 3º curso de la Diplomatura en Ciencias Empresariales.

La metodología utilizada para la realización de estas clases fue la lección magistral activa para las clases teóricas combinada con una parte práctica desarrollada a través de diversos ejercicios en la computadora de actividades relacionadas con el comercio electrónico. Las clases teóricas (cuatro sesiones, dos de derecho y dos de informática) tuvieron una duración de 1,5 horas cada una de las clases de la parte de derecho y 2 horas para cada una de las clases de informática. En primer lugar se expusieron los aspectos jurídicos, para en la segunda parte, impartir los conocimientos informáticos relacionados con los conceptos dados anteriormente.

Para la explicación, tanto en la parte jurídica como en la informática, se utilizaron métodos tradicionales, pizarra, transparencias, así como nuevos recursos para la docencia, tales como el cañón de vídeo, presentaciones de ordenador o la exposición de casos reales. También se accedió a la Red en tiempo real durante la exposición teórica para explicar con ejemplos concretos algunos conceptos. Posteriormente, en otra sesión planificada, los alumnos tuvieron una clase práctica delante de la computadora, de una duración de dos horas.

4.Temario

El temario impartido en esta clase interdisciplinar fue el que se muestra a continuación. Hay que mencionar que aquellos apartados o definiciones comunes a ambas disciplinas no se repitieron oralmente, si bien sí ofrecimos las distintas interpretaciones que, para un mismo concepto, tienen el Derecho y la Informática.

A. *Parte de Derecho Mercantil*.

1.Consideraciones generales.

2.Concepto y clases de Comercio Electrónico.

3.Regulación jurídica.
- 3.1.Directiva sobre Comercio Electrónico.
- 3.2.Ley de Servicios de la Sociedad de la Información.
- 3.3.Otras normas que inciden en el Comercio electrónico.

4.El contrato de compra venta electrónico.
- 4.1.Obligaciones del prestador de servicios.
- 4.2.Perfección del contrato.
- 4.3.Lugar de celebración del contrato.
- 4.4.El valor de la firma electrónica.

B. *Parte de Informática*.

1.Introducción al Comercio Electrónico.
- 1.1.Definición de Comercio Electrónico.
- 1.2.Ventajas del Comercio Electrónico.
- 1.3.Tipos de Comercio Electrónico.
- 1.4.Fases en las que mejora Internet la productividad de la empresa.
- 1.5.Que debe de ofrecer un buen software de Comercio Electrónico.
- 1.6.Comparadores.

2.Seguridad en el Comercio Electrónico.
- 2.1.Criptografía.
- 2.2.Encriptación de clave privada.
- 2.3.Encriptación de clave pública.
- 2.4.La Firma digital.
- 2.5.Autoridades de Certificación.
- 2.6.El Certificado Digital.
- 2.7.Servidor Seguro.
  - 2.7.1.Conceptos.
  - 2.7.2.Identificación de un Servidor Seguro.

3.Formas de pago en el Comercio Electrónico.
- 3.1.Introducción..
- 3.2.Protocolo SET.
- 3.3.Qué es un TPV virtual.
- 3.4.Cómo pagar con TPV virtual.
- 3.5.Quienes intervienen en una compra virtual.

4.Cómo comprar de forma correcta en Internet.

5.Cómo montar nuestra propia tienda virtual.

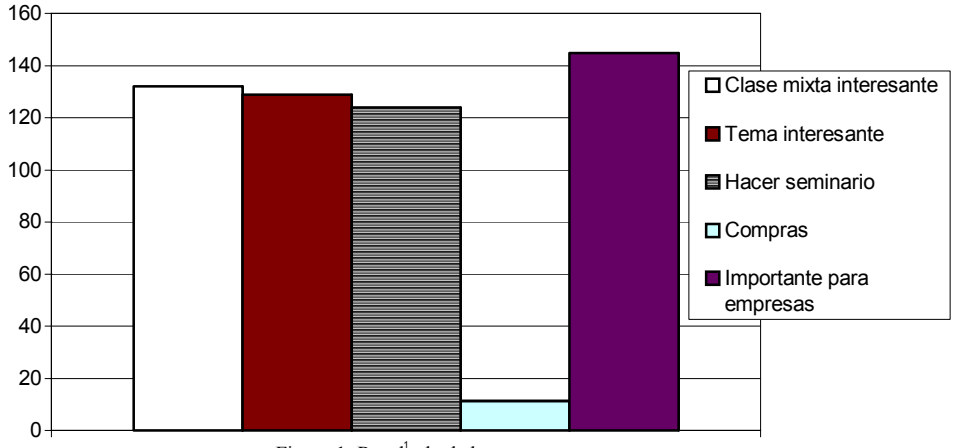


Figura 1. Resultado de la encuesta

A pesar de la aparente extensión del tema, se tratan todos los conceptos de forma clara y concisa, para que el alumno no caiga en la distracción ni el aburrimiento.

Para analizar el resultado de la exposición del tema, se realizó una encuesta, a modo de evaluación general sobre las clases recibidas, a rellenar de forma individual y confidencial, por parte de los alumnos. El cuestionario fue de tipo cerrado, en el cual los alumnos debían de contestar con un simple SI o NO.

Las preguntas de la encuesta fueron las siguientes:

A. ¿Te ha parecido interesante la exposición desde el punto de vista mixto, informático-jurídico?

B. ¿Te ha despertado interés el tema de Comercio Electrónico?

C. ¿Te gustaría realizar un Seminario más específico sobre Comercio Electrónico?

D. ¿Has realizado alguna compra por Internet previa a estudiar este tema?

E. ¿Crees importante que una empresa de servicios tenga una tienda virtual en Internet en la actualidad?

5.Resultados

Lo primero que hay que tener en cuenta es que cuando se tratan temas relacionados con Internet, el interés de los alumnos por las clases es mucho más notable, así como la asistencia, sobre todo si se realizan prácticas con ordenador. A pesar de que Internet ya se encuentra muy expandido, hasta el momento sólo un número relativamente bajo de los alumnos tienen fácil acceso a Internet, tal y como veremos en los resultados de la encuesta realizada. De esta forma, observamos que la utilización de Internet (aunque poco a poco se esta incrementando considerablemente en los domicilios particulares) se limita bien a los ciber-cafés, bien a las aulas de ordenadores de prácticas de las facultades, lo cual resulta bastante escaso en muchas de ellas. En nuestro caso, en la Facultad de CC. EE. y Empresariales disponemos de dos aulas de informática, con 25 equipos cada una.

Siempre que se habla de asignaturas de informática surge el gran problema del déficit de horas prácticas, de las que se beneficiaría el alumnado, y la nada adecuada distribución de

créditos prácticos por asignatura. En relación a éstas también hay que tener presente la distribución de créditos de las asignaturas, en las cuales los créditos prácticos son escasos. Con el agravante, además, de la siempre necesaria duplicación o triplicación del número de grupos de prácticas, con el objetivo de proporcionar una buena calidad docente y no tener un número demasiado elevado de alumnos por grupo. Así, en nuestro caso, tenemos un alumno por ordenador, distribuidos los grupos en las dos aulas de informática.

Otro aspecto que conviene recordar es que cuando se explica informática en titulaciones que no son las propias de informáticos, los alumnos suelen llegar con escasos o nulos conocimientos informáticos [1]. Por este motivo hay que diseñar los temarios de informática de tal forma que trate todos los aspectos básicos, en nuestro caso de Comercio Electrónico, sin llegar a profundizar en exceso. Es decir, debemos ofrecer la informática como una herramienta de trabajo, un medio, no un fin en sí misma considerada. De esta manera se evita el aburrimiento del alumno y se le enseña las utilidades necesarias para que en un futuro laboral sea capaz de desenvolverse con facilidad y ampliar conocimientos.

Para conseguirlo es imprescindible provocar una alta participación en clase, atraer al alumno con el continuo lanzamiento de preguntas, captar el interés y, por supuesto, la asistencia a clase. Somos conscientes de que un alto porcentaje de nuestros alumnos no son vocacionales, sino que acceden a nuestra Facultad bien por no haber alcanzado la nota requerida para acceder a otra titulación o universidad, bien por no tener una idea clara sobre sus intenciones laborales. Somos, pues, los profesores quienes tenemos la difícil tarea de intentar que esa apatía inicial, falta de interés o desmotivación se reduzca, incluso, en el mejor de los casos desaparezca.

La evaluación de la experiencia desarrollada por profesores de estos dos departamentos, dos mundos que cada vez están más interrelacionados [3], se realizó sobre un total de 161 alumnos de las tres titulaciones, distribuidos en 36 de 2º de la Licenciatura en Administración y Dirección de Empresas, 38 de 2º de la Diplomatura en Relaciones Laborales y 87 de 2º de la Diplomatura en Ciencias Empresariales. La encuesta arroja unos resultados concretos (como se pueden apreciar en la figura 1) que pueden resumirse en los siguientes puntos:

Pregunta A. ¿Te ha parecido interesante la exposición desde el punto de vista mixto, informático-jurídico? Los alumnos saben del continuo auge del comercio electrónico en las empresas por lo que valoran como interesante que se incluya en el temario su explicación. Un 82% de los encuestados respondieron favorablemente a su impartición interdisciplinar, lo que demuestra que fueron capaces de captar las interconexiones jurídicas e informáticas.

Pregunta B. ¿Te ha despertado interés el tema de Comercio Electrónico? Un 80% de los encuestados respondieron afirmativamente.

Pregunta C. ¿Te gustaría realizar un Seminario más específico sobre Comercio Electrónico? Un 77% manifestó su opinión favorable sobre la celebración de un seminario, en el cual siempre se puede profundizar en más aspectos, e incluso permitir que empresas reales muestren, mediante conferencias o charlas, sus experiencias y formas de trabajar en el comercio electrónico. El hecho de ver en la realizad cómo se aplican los conceptos estudiados, siempre les resulta interesante.

Pregunta D. ¿Has realizado alguna compra por Internet previa a estudiar este tema? Tan sólo un 7% había realizado alguna compra por Internet antes de estudiar este tema. Prácticamente ninguno conocía aspectos tan importantes como el de la seguridad en los pagos electrónicos, lo cual le identifica con el resto de usuarios españoles, que siguen siendo bastantes reacios a dar su número de tarjeta a través de la red, por miedo a estafas o engaños por parte de las empresas.

Pregunta E. ¿Crees importante que una empresa de servicios tenga una tienda virtual en Internet en la actualidad? A la mayoría, un 90%, le parece importante que las empresas tengan una tienda virtual, ya que ofrecer una buena imagen

de la empresa es necesaria dentro del mercado competitivo en el que hoy vivimos.

6. Conclusiones

Universidad es sinónimo de universal. Por esto creemos que al alumno se le deben transmitir los estudios de una forma global, completa. Para ello es imprescindible la coordinación entre departamentos y áreas, complementando unas asignaturas con otras. El alumno debe percibir que cada asignatura no es un departamento estanco que debe ir aprobando, sino que ha de observar la titulación en su conjunto interrelacionando conceptos de distintas materias. Sólo así podrá comprobar la utilidad de su paso por la universidad.

Las conclusiones que, por tanto, podemos destacar son:

1. Los buenos resultados obtenidos con este planteamiento se constataron con un alto grado de asistencia a estas clases, lo cual nos animó a incluir de forma permanente este tema en la asignatura de Derecho Mercantil de las titulaciones de L.A.D.E.[3], L.ECO[4], D.RR.LL[5]. y D.CC.EE.[6], así como introducirlo en la asignatura troncal de tercero de D.CC.EE. denominada *Informática Aplicada a la Gestión de la Empresa*. El hecho de tratar temas relacionados con Internet hace que el interés y asistencia por parte del alumnado se incremente, sobre todo si parte de la clase es práctica y el alumno tiene la oportunidad de experimentar por sí mismo delante del ordenador todo lo que se le ha explicado.

2. Consideramos imprescindible para la formación de futuros empresarios o administradores de empresas la realización de alguna práctica sobre Internet, así como la introducción de los aspectos más básicos relacionados con Internet y el Comercio Electrónico.

3. El carácter de *modernidad* y actualidad de los contenidos expuestos incentiva el acercamiento a los intereses académicos del alumno.

4. La buena aceptación de la clase nos motivó a la realización de un completo, más amplio y detallado Seminario sobre el Comercio Electrónico, donde además de los aspectos informáticos-jurídicos, se incluyeron temas económicos relativos a Internet. Aprovechando la situación geográfica de Extremadura se enfocó desde una perspectiva hispano-lusa[7]. En este seminario, realizado en octubre de 2002, se cubrió el 100% de las matriculas.

5. Creemos que en una titulación, incluso en un curso, deben establecerse y coordinarse las asignaturas y programas de los distintos departamentos y áreas que imparten docencia en ella, para, de esta forma, complementar unas asignaturas con otras y ofrecer una formación más completa al alumno. Este vería una relación entre asignaturas de distintos departamentos, comprendería la carrera como un todo, y el profesor no tendría que limitarse a exponer conceptos generales y repetidos en otras áreas de la misma titulación.

7. Bibliografía utilizada para el temario

- Aguila, A.R. del. *Comercio Electrónico y Estrategia Empresarial: Hacia la economía digital*. Ed. Ra-Ma. 2001.
- Aguila, A. R. del y PADILLA, A. *E-business y Comercio electrónico. Un enfoque estratégico*. Ed. RA-MA, 2001.
- Calvo Orra, A. y otros. *Comercio Electrónico en Internet*. Ed. International Thonson Editores Spain Paraninfo, S.A.

[3] Licenciatura en Administración y Dirección de Empresas

[4] Licenciatura en Economía

[5] Diplomatura en Relaciones Laborales

[6] Diplomatura en Ciencias Empresariales

[7] Seminario "*El comercio electrónico: una visión hispano-lusa*", celebrado los días 17 y 18 de octubre de 2002 en la Facultad de Ciencias Económicas y Empresariales de Badajoz.

- Cremades, Fernández-Ordóñez, Illiescas Ortiz. *Régimen jurídico de Internet*. Editorial La Ley 2002.
- *Directiva 200/31/CE del Parlamento Europeo y del Consejo*, de 8 de junio de 2000, relativa a determinados aspectos jurídicos de los servicios de la sociedad de la información, en particular el comercio electrónico en el mercado interior. DOCE nº L 178, de 17/07/2000.
- *El comercio electrónico en España: Posicionamiento de las empresas españolas y previsiones de futuro*. Congreso AUI.
- Algunas web acerca del comercio electrónico, formas de pago y conceptos sobre Servidores Seguros, desde una perspectiva informática:
  http://www.sets1.mcyt.es/novedad/
  http://www.senyal.com/centreshop
  http://www.grupomaj.com/atrasados

  http://www.iec.csic.es/criptonomicon
- Illiescas Ortiz, R, Ramos Herranz, I. *Derecho del comercio electrónico. Biblioteca de Derecho de los Negocios*. Ed. La Ley. Madrid 2001.
- Illescas Ortiz, R. *Derecho del comercio electrónico*. Ed. La Ley-Actualidad.
- Illescas Ortiz, R. *La firma electrónica y el Real Decreto 14/1999, de 17 de septiembre*. Rev. Derecho de los Negocios. Octubre 1999, pp. 1 a 14.
- *Ley de comercio minorista* Ley 7/1996 de 15 de enero BOE 17 enero 1996 Real Decreto-Ley 14/1999, de 17 de septiembre , sobre firma electrónica. Anteproyecto de Ley de servicios de la Sociedad de la información y de comercio electrónico.
- Martínez Nadal, A. *Comercio electrónico, firma digital y autoridades de certificación*. Ed. Cívitas Ediciones S.L.
- Ramos Herranz, I. *Registro bajo .es en Internet*. Rev. Derecho de los Negocios Julio-Agosto 2000. Pp. 13 a 26
- Ramos Suárez, Fernando. *Problemas jurídicos del comercio electrónico*. Revista electrónica de Derecho Informático (R.E.D.I.), edita vlex.com ISSN 1576-7124.
- Vargas Gómez-Urritia, Marina. *Comercio Internacional electrónico y conflicto de leyes y de jurisprudencia en el cyberespacio*. Rev. Derecho de los Negocios, abril 2000, pags. 1 a 26.

Referencias

[1] Moreno Liso, Lourdes. *El Comercio Electrónico: Una visión hispano-lusa.* ISBN:84-688-0912-8. Pag. 71-84. Marzo, 2003.

[2] Lapeña Marcos, María Jesús. *La enseñanza de la informática en estudios empresariales*. VII JENUI 2001 (Madrid).

[3] Ull Port, Eugenio. *Derecho privado y derecho público de la informática*. UNED, Madrid 2000.

La informática en los estudios de Gestión y Administración Pública

Francisco Araque
Facultad de Ciencias Económicas y Empresariales
Universidad de Granada
España
faraque@ugr.es

Juan José Gaitán
IP Learning e-ducativa
España
jjgaitan@e-ducativa.com

Vlasta Hlavickova
Fac. Rel. Internacionales
Universidad de Económicas de Praga
República Checa
hlavicko@vse.cz

Ernesto Zianni
Facultad de Ciencias Económicas y Empresariales
Universidad Nacional del Litoral – Argentina
ezianni@fce.unl.edu.ar

Resumen

En este trabajo presentamos nuestra propuesta de contenidos académicos, tanto teóricos como prácticos, para la asignatura de informática en la titulación de Gestión y Administración Pública. Se describe también la experiencia docente, la tecnología utilizada y los métodos de trabajo empleados en asignaturas de informática impartidas en Facultades de Ciencias Económicas y Empresariales (C.C. E.E. y E.E.) y que pueden ser aplicados en las asignaturas de informática de Gestión y Administración Pública.

1. Introducción

La diplomatura en Gestión y Administración Pública (GAP), perteneciente al área de Ciencias Sociales y Jurídicas, es una titulación de primer ciclo centrada en el estudio de la Administración Pública y está pensada para crear cuadros intermedios expertos en gestión administrativa y financiera [17][18]. El Real Decreto 1426/1990, de 26 de octubre, estableció el título universitario oficial de Diplomado en Gestión y Administración Pública y las directrices generales propias de los planes de estudios conducentes a la obtención de aquél. (BOE 20-11-90). Tanto en Argentina como en la República Checa existen titulaciones similares adecuadas a las necesidades específicas de cada país.

El contenido de la carrera se centra fundamentalmente en el estudio de la gestión administrativa y financiera, aunque también incluye una amplia gama de materias diferentes que se relacionan con la informática, el derecho, la estadística, la economía, la contabilidad, los recursos humanos, dirección y planificación de organizaciones, la documentación o la sociología, todas ellas aplicadas al ámbito de las Administraciones Públicas.
De esta forma un titulado logra un perfil multidisciplinar con un conocimiento profundo del entorno legal, económico y social siendo personas adecuadas para trabajar en cualquier sector de la Administración Pública, aunque su formación integral les hace totalmente adecuados para su incorporación en el Sector Privado.

En este trabajo presentamos nuestra propuesta de contenidos teóricos y prácticos para impartir la asignatura de informática en la titulación de GAP.

Creemos que, dado que los alumnos que realicen dicha titulación acabarán trabajando en la administración pública (que también es una gran empresa) o en el sector privado, necesitarán una visión de la informática desde el punto de vista empresarial, no solo desde el punto de vista académico como tradicionalmente se ha venido haciendo. Cuestiones como e-business, data warehouse, modelos de negocio, e-govermment, seguridad en internet, etc. deben incluirse en el programa de la asignatura, y no solo cuestiones como software, hardware, bases de datos, etc.

En el entorno de la Unión Europea cada vez más las relaciones del sector público y privado son de carácter internacional (entre países de la UE o entre algún país miembro de la UE y otro país). Pensando en esto, y teniendo en cuenta la experiencia positiva anterior [3][4][8], creemos

que es importante dar una visión práctica y de carácter internacional de la asignatura. Esto se consigue realizando las prácticas entre diferentes alumnos de diferentes países. Estas actuaciones conjuntas se realizan de forma asíncrona dado la dificultad que supondría realizarlo de otra forma.

Además, todo el programa teórico debe tener su correspondencia en el programa práctico. Es en este punto donde aportamos nuestra experiencia en el ámbito de asignaturas de informática impartidas en Facultades de C.C. E.E. y E.E. [4] y proponemos un nuevo marco para realizar prácticas de e-business. La Universidad de Granada se encuentra en fase de implantación de la titulación de GAP en la Facultad de C.C. E.E. y E.E. de Granada. Creemos que sería de utilidad para el desarrollo de la misma la experiencia y propuesta que a continuación planteamos.

Dado que lo que se pretende es que interactúen alumnos de diferentes universidades en diversos países, se delinearon algunas propuestas conducentes al aprovechamiento de Internet y sus servicios básicos como vehículo para la realización de experiencias pedagógicas. En este trabajo pretendemos presentar la experiencia adquirida en el desarrollo de las prácticas conjuntas entre la Universidad de Granada (UGR), la Universiadd Nacional del Litoral) UNL y la Universidad de Económicas de Praga (UEP).

A continuación se comentan los contenidos académicos de las asignaturas de informática y se introduce nuestra propuesta de contenidos teóricos y prácticos. En el punto tercero se expone la experiencia adquirida en otras asignaturas y proyectos realizados. En la sección cuarta se explica la arquitectura funcional propuesta y se finaliza con las conclusiones.

2. La informática en la titulación de gestión y administración pública.

Actualmente la titulación de GAP se imparte en 25 centros de España. Los perfiles profesiona-les del titulado son entre otros [17]:

... técnico con gran capacidad de comprensión y adaptación a la normativa cambiante y a las nuevas técnicas de gestión; Enlace entre la dirección y el núcleo organizativo; Enlace entre la Administración y el sector privado; Personal con iniciativa y con criterio a la hora de evaluar los problemas públicos; Personal con capacidad de argumentar e introducir innovaciones en la gestión de los servicios públicos y en la organización administrativa; Profesional con conocimientos, como hemos visto anteriormente, jurídicos, económicos, politológicos, de gestión de personal, contables, informáticos, estadísticos, de gestión de servicios públicos etc.; Gerente público y privado; Profesional que conecta al ciudadano con la Administración ....

Y algunas de las competencias del diplomado en GAP son [informe]:

... dirección, planificación y control de la puesta en funcionamiento de las técnicas de gestión pública; Dirigir y planificar el funcionamiento de los servicios que prestan las empresas privadas por encargo de la Administración; Asesorar a las empresas privadas, asociaciones, fundaciones sobre su relación con la Administración; Ser técnicos en la gestión administrativa y económica de las Administraciones; Dirección de secciones y negociados relativos a Recursos Humanos, Personal, intervención ...

En general, las asignaturas relacionadas con Informática impartidas por las diferentes Universidades en la Diplomatura de Gestión y Administración Pública suelen tener las siguientes características:

Contenido académico similar en la mayoría de los casos:

- La parte teórica: Introducción (Concep-tos, Sistemas de Información), Funda-mentos tecnológicos (software, hardware, bases de datos, redes).
- Y la parte práctica: Conceptos básicos (Windows, e-mail), Paquetes Integrados (Procesador de textos, Hoja de cálculo, Bases de Datos) y Comunicaciones (Na-vegación, Diseño de Páginas Web).
- Suele haber una troncal y una o varias obligatorias. Además de las posibles optativas. Varían entre 4,5 y 6 créditos
- En la mayoría de los casos las imparten departamentos de informática. En algunas universidades lo hacen departamentos de biblioteconomía y documentación.

En muy pocos casos incorporan en la parte teórica temas relacionados con Tecnologías avanzadas (DSS, EIS, EIP, DW, E-Business, XML, seguridad, etc.) [2][5][6][9][13] menos aún en la

parte práctica. Otra parte importante a incluir es la descripción de los proyectos de tecnologías de la información y las comunicaciones que desde la propia administración pública se impulsan, y que podemos encontrar detalladamente en [20], [21].

Es en el tema de tecnologías avanzadas donde nosotros creemos que es necesario hacer un esfuerzo a la hora de impartir la asignatura de Informática. Todo lo que se aplica en el ámbito empresarial es aplicable en el ámbito del sector público. Cada vez más se tiende a optimizar los recursos públicos y es necesario cambiar el enfoque hasta ahora empleado. Podemos ver a las distintas administraciones (local, regional, estatal) como empresas de diferente tamaño. Las empresas y administraciones públicas están implantando soluciones basadas en E-business para adquirir ventaja competitiva respecto a sus competidores.

El E-commerce es el intercambio de bienes y servicios realizado a través de las Tecnologías de la Información y las Comunicaciones [2].

El E-business, en cambio, es la transformación de los procesos clave del negocio, mediante el uso de las tecnologías de Internet [2]. Una organización basada en e-business, conecta directamente los sistemas críticos del negocio, con sus componentes (clientes, empleados, vendedores, proveedores, partners, etc.) a través de intranets, extranets y la World Wide Web. El e-business es la nueva forma de comercio en Internet, sin fronteras, con un nuevo planteamiento de los modelos empresariales, clientes globales, nuevos sistemas de pago y estrategias innovadoras. En este sentido los titulados en GAP deben de ser capaces de entender todo lo relacionado con los procesos de negocio desde el punto de vista de las tecnologías de la información y las comunicaciones.

E-Business está compuesto, al menos y dependiendo de los autores, de las siguientes tecnologías:

- Modelos de negocio: B2B, B2C, B2A, C2A.
- Customer relationship management (CRM).
- Supply Chain Management (SCM), Supply Chain Planning (SCP).
- Business Intelligence (BI).
- Knowledge Management (KM).

Dentro del e-business, las materias que deberían tener cabida en la asignatura de informática serían:

- Modelos de negocios: E-Goverment (Formas de relación entre los ciudadanos y las Administraciones Públicas, y entre estas últimas, realizadas mediante tecnologías de la información y de las telecomunicaciones. Ejemplos: la declaración de impuestos a través de Internet, o los servicios de información y tramitación ofrecidos a través de los sitios web de las Administraciones Públicas). Engloba B2A y C2A [20][21].
- Customer relationship management (CRM): una nueva filosofía empresarial, que orienta las estrategias de las organizaciones a reforzar y mantener las relaciones de la empresa con los clientes, y más en concreto, a mantener y reforzar las relaciones con los clientes (ciudadanos) [10].
- Seguridad en el e-business: certificados y huellas digitales, firmas electrónicas, comunicaciones seguras, etc.

Otra área temática que se debe incluir en los planes de estudio y que en muy pocos está contemplada es la relacionada con los Almacenes de Datos (Data Warehouse, DW). Este tema se podría incluir en el de Bases de Datos o bien en el de Tecnologías Avanzadas. Un almacén de datos [11][12] es una base de datos que contiene una copia de datos de los sistemas operacionales (fuentes de datos) con una estructura especialmente adecuada para realizar consultas y análisis (ofrece una visión histórica de los datos de los sistemas operacionales). Los DW son utilizados en labores de análisis y son de gran utilidad a la hora de tomar decisiones por parte de los directivos de las empresas, públicas o privadas [1].

3. Experiencia en prácticas de e-business

Parece claro que bajo este denominador común, y después de ponerse de acuerdo en varias cuestiones organizativas, no debería de haber problema en la realización de prácticas de manera conjunta entre grupos de alumnos de diferentes Universidades. Esta experiencia ya se ha realizado en cursos anteriores (ver [4][8]).

Con esta idea, y también con el objetivo de fomentar la creatividad, el trabajo en equipo y las relaciones internacionales, fue como surgió la idea de montar unas prácticas conjuntas entre dos Universidades de diferentes países. Por una parte

la Universidad de Granada (UGR, España), la Universidad Nacional del Litoral (UNL, Argentina) y con la idea de fomentar el uso del inglés la Universidad de Económicas de Praga (República Checa).

En una primera fase ya realizada y superada, las prácticas integradas implicaban un intercambio de mensajes entre los participantes junto con el manejo de la base de datos asociada. Se analizaba la operatoria comercial de cuatro empresas –ficticias- de un determinado ámbito comercial que se dedican a la fabricación y comercialización de distintos productos finales e intermedios (lámparas, faros, ventiladores, etc.) y los materiales y repuestos necesarios (cables, enchufes, etc.). Cada una de estas empresas fabricaba el producto final empleando para ello materiales de elaboración propia, y materiales que adquiere a una o más de las otras empresas. Los alumnos diseñaban la base de datos y la utilizaban en la gestión administrativa de su "empresa virtual".

En la segunda fase, la idea era permitir que los alumnos interactúen académicamente en un entorno de negocios virtuales a través de internet. En este caso, se trataba de trabajar con las entidades que intervienen en el Comercio Electrónico: el que compra (cliente), el que vende (tienda), el(los) bancos encargados de realizar las transacciones y la pasarela de pago [12]. Para las dos primeras entidades la operación de compra se realiza de forma transparente: un cliente se conecta a una tienda, compra unos productos y los paga automáticamente sin ser conscientes de que existe una pasarela de pago virtual que está realizando la transacción. La pasarela de pago se encarga de realizar las labores propias de un notario que permite realizar operaciones de transferencia de capital entre las entidades participantes. Se dispone también de una herramienta para la creación de tiendas virtuales de una forma sencilla. Como era previsible, para completar el entorno necesario, se dispone de un banco virtual ficticio en la que cada cliente podrá realizar operaciones bancarias sobre sus cuentas corrientes y tarjetas de crédito necesarias para la compra de productos a través de Internet. Cada grupo de prácticas comenzaba las prácticas con un saldo mínimos en el banco virtual.

Los pasos a seguir, una vez organizados los gru-pos de prácticas (3 o 4 personas) eran:

- Pensar algún tipo de empresa y delinear un plan estratégico básico.
- Dar de alta la tienda utilizando el generador de tiendas virtuales y crear una cuenta en el banco.
- Realizar una mailing informando de la creación de la empresa al resto de alumnos.
- Realizar transacciones comerciales (compras y ventas) con otros grupos de alumnos.
- Pagar las compras utilizando sus cuentas corrientes y tarjetas de crédito ficticias.
- Realizar informes comerciales de sus ingresos y ventas.

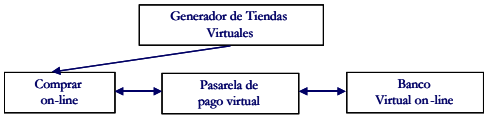


Figura 1. Arquitectura de la 2ª aproximación.

Vistas las carencias de las dos experiencias anteriores, lo siguiente que nos planteamos es intentar subsanarlas en la medida de lo posible. Por una parte, es necesario descargar al usuario de la operativa innecesaria y tediosa, y por otra, es conveniente que pueda utilizar una base de datos diseñada por los alumnos y hacerlo de una manera transparente y fácil para el usuario. Lo que nos planteamos fue la utilización de un Portal [7] en el que se integren varias cuestiones:

- Punto de encuentro entre docentes para intercambio de experiencias, aportación de conocimientos, etc.
- Punto de encuentro para alumnos. Foros de consulta, intercambio de apuntes, dudas, etc.
- Incorporar al Portal, sistema de Juegos de Empresa (en desarrollo), donde las distintas empresas virtuales compitan por el logro de ciertos objetivos comerciales y empresariales, en un entorno globalizado [16].
- Punto de encuentro para la realización de las prácticas.

4. Nueva arquitectura de prácticas integradas

Como continuación de lo comentado en el apartado anterior, y para la realización de las prácticas integradas se ha pensado la implantación de un e-hub [15]. Un e-hub se define como un intermediario neutral basado en Internet y especializado en industrias verticales o en procesos de negocio específicos. Utiliza mecanismos basados en el

mercado para mediar en cualquier transacción que se realice entre las empresas. Los e-hub crean valor añadido reuniendo a compradores y vendedores, creando liquidez y reduciendo costos. La idea de un e-hub es que una empresa, conectándose a un solo lugar, pueda acceder a muchos vendedores y suministradores. Para entender los e-hub centrados en B2B es necesario entender qué y cómo compra una empresa.

A grandes rasgos, los qué compra una empresa se puede dividir en dos grandes bloques: compras para la fabricación y compras para el funcionamiento. Las primeras van directamente a la cadena de producción de la empresa, son productos específicos para cada empresa y suministrados por empresas especializadas. Los segundos son necesarios para el trabajo diario de la empresa y se denominan MRO (Maintance, Repair and Operating). En este último caso se incluyen billetes de avión, faxes, ordenadores, fotocopiadoras, etc.

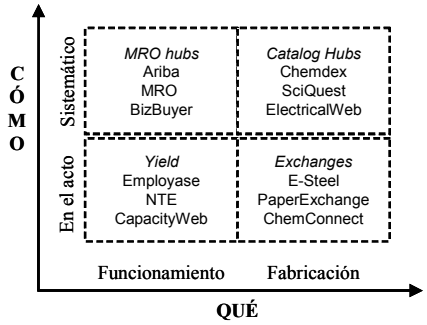


Figura 2. Clasificación de los E-Hub

Cómo compran las empresas se puede dividir en: aprovisionamiento sistemático y aprovisionamiento en el acto o en tiempo-real. En el primer caso los compradores negocian unos pre-contratos con vendedores especializados en base a unas condiciones. Suelen ser una relación duradera en cuanto al tiempo entre los participantes. En el segundo caso los compradores no saben a quien están comprando, lo hacen a vendedores "anónimos". Este tipo de relación es orientada a la transacción y en muy pocas ocasiones la relación entre el comprador y el vendedor es duradera. Una clasificación la podemos ver en la figura 2.

En esa primera fase (centrándonos en la nueva arquitectura) nos hemos centrado en los e-hub llamados Exchanges. Más adelante, está previsto que se incorporen nuevos tipos de e-hub. En el caso de los Exchanges, y siguiendo los pasos de la primera fase, las empresas compran productos necesarios para su cadena de producción. Los compran a vendedores anónimos (también son los alumnos) y la relación entre el comprador y el vendedor no suele ir más lejos de la propia transacción. El e-hub se encargaría de:

- Administración de contratos entre participantes: administración de las condiciones, etc.
- Administración de pedidos: realización, confirmación, historia, etc. de pedidos, personalización en base al cliente.
- Información técnica: facilita la administración de las operaciones en los terminales, enruta los mensajes XML/EDI [5], proporciona estadísticas detalladas de todas las operaciones y conexiones (útil para el CRM).
- Atención al cliente: seguimiento, modificación de pedidos y de facturas, personalización del interface, acceso a la información necesaria, etc.
- Integración de los sistemas empresariales la empresa: integración de los ERP existentes.

Con la idea de, por una parte integrar las prácticas de bases de datos y las de e-business (realizadas con el e-hub), y por otra subsanar las carencias de experiencias anteriores, está previsto que los alumnos diseñen una base de datos y que ésta se integre (en base a consultas SQL realizadas por los alumnos, grabadas en un fichero plano de texto y posteriormente integradas de forma transparente para el usuario) en el e-hub. Habría otra parte, común a todas las empresas creadas, que incluiría la operatoria habitual (pedidos, contabilidad básica, stock, etc.) y que ya estaría incluida en el e-hub. Con esto se consigue que el alumno tenga que dedicar parte de su esfuerzo al diseño de una base de datos y a la utilización de, en nuestro caso Microsoft Access. La base de datos haría el papel del sistema existente en la empresa y que tienes que ser integrada en el e-hub.

Los pasos a seguir por el grupo de alumnos sería (sujeta a modificaciones ya que está en proceso de desarrollo) serian los mismos comentados en el punto 3. Se prevé también que se realicen nuevas tareas por parte de los alumnos que actualmente sólo se llevan a cabo de forma teórica pero que en un futuro próximo se llevarán a cabo también de forma práctica, tales como:

- Crear Empresas que sean parte de la admin. Pública (dptos, negociados, etc.).
- Diseñar, a pequeña escala y basándose en la teoría de otras asignaturas, formatos de intercambio de datos. Por ejemplo, diseñar un negociado (como una pequeña empresa) para rellenar formularios relativos a una determinada actividad (pago de impuestos municipales). En este caso la BD habría que diseñarla con el objetivo de almacenar los datos necesarios. Esto se podría pasar a XML e intercambiarlo con otros grupos de prácticas, por ejemplo de otra Universidad o país.
- Diseño de un sistema normalizado de intercambio de información entre las empresas privadas y la Administración Central.
- Creación de un sistema de registro de las operaciones de las empresas, generado a partir de la presentación de informes o rendiciones periódicas.
- Desarrollo de un registro de proveedores, para lo cual se espera que las empresas (grupos de alumnos) puedan articular el cumplimiento de normas específicas de certificación (similares a ISO9000), con el fin de cumplimentar los requerimientos para inscribirse como proveedores.
- Establecer parámetros, servicios de información, seguimiento de expedientes y documentaciones, de manera que todos estos trámites permitan consolidar la idea y la importancia de establecer y cumplir normas específicas que permitan viabilizar el funcionamiento de un sistema integrado de información.

Esta fase, así como la segunda lo fue, está siendo desarrollada por alumnos de proyectos de 5 de la Universidad de Granada [7][12].

En la figura 5 podemos ver la arquitectura funcional de nuestra propuesta. Por una parte tenemos el e-hub, del que se cargan datos en el Almacén de Datos y en el Banco Virtual. Por otra parte está la plataforma de E-learning. En nuestro caso los clientes podrían ser los ciudadamos y las empresas el sector público (administración).

Las empresas, integradas en el e-hub, son creadas por los alumnos (con ayuda de un asistente). Los clientes de las empresas son los propios alumnos. Las estadísticas generadas en el e-hub (de acceso, de traza de compra de productos, de actualización de stock, de bajada de archivos, de envío de formularios, etc.) son almacenadas en el propio e-hub y transferidas a la herramienta de e-learning.

Esta información, es utilizada por los alumnos para analizar sus ventas (productos más vendidos, páginas más visitadas, etc.) y poder tomar decisiones estratégicas para mejorar la competitividad de su empresa y adquirir ventaja competitiva; y es utilizada por los profesores para ver la evolución de las empresas de los alumnos en base a las decisiones que estos van tomando [3], [8].

Así mismo el banco virtual envía información a la plataforma de e-learning sobre estado de las cuentas, evolución de los saldos, etc. Toda esta información (la del e-hub y la del banco) se presenta en forma texto y gráfica para ayudar a la toma de decisiones tanto al alumno como al docente. La parte de data warehouse que se muestra en la figura 5 está en proceso de diseño y servirá para poder recopilar información tanto del e-hub como del banco virtual para incorporarla al data warehouse y poder utilizarla posteriormente para sistemas de Customer Relationship Management (CRM), Executive Information System (EIS) y Business Intelligence (BI) [14].

Figura 3. Sistema de apoyo a la docencia.

Hasta ahora, en lugar de una plataforma de E-learning comercial, hemos utilizado como complemento a la docencia y para intercambiar archivos, apuntes, documentos, trabajos, etc. y para asignar prácticas y trabajos teóricos una herramienta llamada SpiGa [7] (Figura 4).

En un futuro próximo, tenemos pensado incorporar una herramienta de e-learning comercial (figura 6). Tenemos instalada en periodo de pruebas en la Universidad de Granada y utilizándola en algunas asignaturas la plataforma de E-ducativa [19].

La incorporación de la Plataforma contribuirá, entre otras cuestiones, a la creación de la "comunidad", lo que se facilita en gran medida con las herramientas de comunicación de eventos y novedades, tales como el Calendario común, o las noticias y newsletters, a las que los alumnos se suscriben para estar enterados y actualizados de todo lo que sucede.

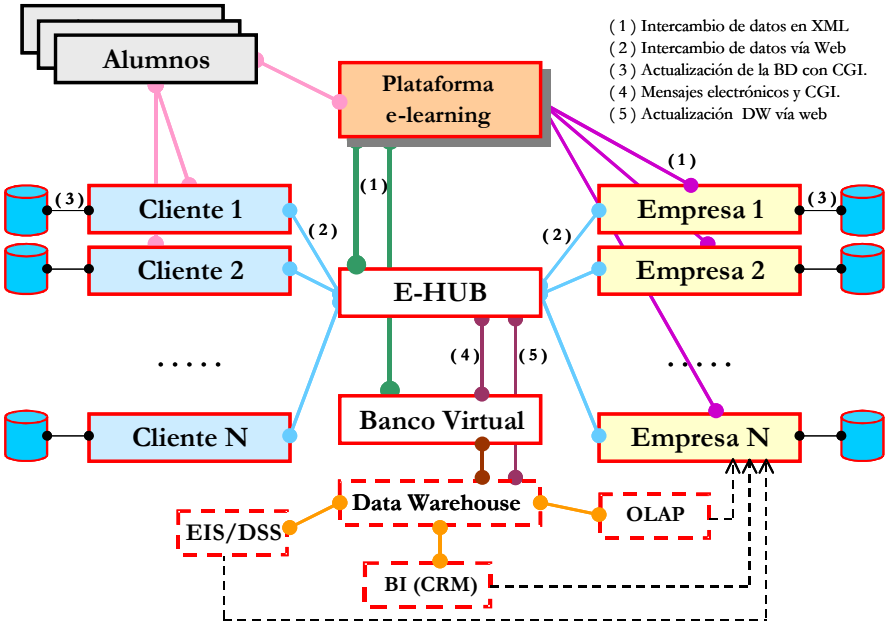


Figura 4. Arquitectura funcional

La comunicación asincrónica es fundamental, y si bien en experiencias anteriores el e-mail tradicional permitía este tipo de tráfico de mensajes, nada puede compararse con la posibilidad de tener un entorno único e integrado para contener todo este espacio pedagógico. La mensajería interna y los foros fomentan la participación.

Figura 5. Plataforma de e-learning

La comunicación sincrónica, a base de una sala de chat y la videoconferencia, permite optimizar algunas situaciones mediante la comunicación on-line. La posibilidad de contener y compartir un repositorio de archivos y recursos es de gran utilidad. Lo mismo sucedió con los sitios de internet, espacio que se conforma y enriquece con el aporte de cada uno de los alumnos, en una participación responsable y controlada.

De este modo, todo lo que es comunicación interpersonal, de información académica y relativa a los ejercicios, se realiza a través de la Plataforma, mientras que el resto de las operaciones son realizadas vía e-hub. La plataforma se comporta como el espacio de comunicación natural de todos los actores de esta experiencia (alumnos y docentes), pero además puede utilizarse como verdadera plataforma, ya que desde la sección SITIOS, es posible enlazar con los sitios web de cada una de las empresas.

5. Conclusiones

En el presente trabajo hemos presentado como podría enfocarse el desarrollo de las asignaturas de informática impartidas en la titulación de Gestión y Administración Pública. También se ha comentado nuestra experiencia docente adquirida en proyectos anteriores y como puede mejorarse y ampliarse para adaptarse a las nuevas tecnologías y necesidades.

Como trabajo actual y futuro nos planteamos:

- Completar la implementación del marco propuesto (una primera fase está implementada y probada [4]).
- Avanzar en la puesta en marcha de prácticas conjuntas, con grupos de prácticas formados por personas de diferentes Universidades.
- Ampliar la plataforma de e-learning con los módulos necesarios para que se pueda realizar la comunicación entre ésta y el resto de los componentes de la arquitectura presentada.

Este trabajo ha sido parcialmente financiado por el proyecto de investigación de la CICYT con código TIC2000-1723-C02-02.

Referencias

[1] Abelló, F. Araque, J. Samos, F. Saltor. *Bases de Datos Federadas, Almacenes de Datos y Análisis Multidimensional*. Taller Almacenes de datos y tecnología OLAP dentro de las VI Jornadas de Ingeniería del Software y Bases de Datos (JISBD´2001), 21-23 noviembre 2001, Almagro (Ciudad Real).

[2] Ana Rosa del Águila, Antonio Padilla. *E-business y Comercio Electrónico: un enfoque estratégico*. Rama, 2001.

[3] Araque, F., Gaitán, J.J. *Elección e implantación de una plataforma de e-learning en una universidad argentina en un contexto de crisis.* 2º Congreso Internacional "Docencia Universitaria e Innovación (CIDU´02). Tarragona, 1, 2, 3 de Julio del 2002.

[4] Araque, F., Gaitán, J.J., Vlasta Hlavickova. *Prácticas internacionales integradas de e-business*. VIII Jornadas de Enseñanza Universitaria de la Informática (JENUI'02). Cáceres, 10-12 de Julio, 2002.

[5] Araque, M.V. Hurtado, M.M. Abad. *Un nuevo marco para el Intercambio Electrónico de Datos: XML/EDI*, en el I Encuentro Internacional de Finanzas y SI. Noviembre de 2000, Cádiz.

[6] Bryan, Martin, (1998), Guidelines for using XML for Electronic Data Interchange, Proceedings of SGML/XML Europe'98, 17-21 May, Paris (France), pp. 523-548.

[7] F. Valverde, D. Rodero. *SpiGA: Sistema por Internet de Gestión de Asignaturas.* Proyecto Final de Carrera. 2001. Dirigido por Francisco Araque Cuenca, Universidad de Granada.

[8] Gaitán, J.J, Araque, F. *La plataforma de e-learning como soporte a las prácticas de E-Business*. International Conference on ITC´s in Education. 13-16 Noviembre, 2002. ISBN: 84-95251-70-0.

[9] Inmon, W.H.; Welch, J.D.; Glassey, K., (1996), Managing the Data Warehouse, J. Wiley and Sons.

[10] José Agustín Martínez García,Esteban Pérez Milla. CRM:la nueva filosofía empresarial centrada en el cliente. Telefónica Investigación y Desarrollo. Noviembre, 2001.

[11] Joseph Firestone. *Enterprise Knowledge Portals: What They Are and What They Do*. Journal Of Knowledge And Innovation, October, 2000

[12] M. García, J. Martínez. *Simulación del Comercio Electrónico a través de Empresas y Bancos Virtuales*. Proyecto Final de Carrera. 1998. Dirigido por Francisco Araque Cuenca, Universidad de Granada.

[13] Mark M. Davydov. *Corporate Portals and E-business integration*. McGraw-Hill, 2001.

[14] Richard Hackathorn, Web farming for the data warehouse (1999), Ed. Morgan Kaufmann.

[15] Steven Kaplan, Mohanbir Sawhney. *E-Hubs: The new B2B marketplaces*. Harvard Business Review, may-June 2000.

[16] Thompson Stappenbeck. *The Business Strategy Game*. McGraw-Hill, 1998.

[17] Informe de situación de G.A.P. y notas sobre el II Encuentro Nacional de Estudiantes de G.A.P. Mayo de 2001.

[18] www.um.es/adegap.

[19] www.e-ducativa.com.

[20] www.map.es, www.inap.es.

[21] www.administración.es.

Informática teórica

La teoría de autómatas y lenguajes formales, en la práctica

José A. Troyano, Víctor J. Díaz, Fernando Enríquez de S., Javier Barroso
Dept. de Lenguajes y Sistemas Informáticos
Universidad de Sevilla
41012 Sevilla
e-mail: troyano@lsi.us.es

Resumen

Se presenta una propuesta de trabajo práctico para una asignatura de la materia Teoría de Autómatas y Lenguajes Formales. Tradicionalmente este tipo de asignaturas suelen centrar su temario en aspectos teóricos y no se aprovecha el potencial práctico que supone la implementación y aplicación de los conceptos relacionados con la descripción de lenguajes y modelos de autómatas. En este trabajo se parte del modelo de máquina de Moore, para presentar el modelo de Markov como una extensión probabilística. Por último se propone como práctica de curso el desarrollo de un etiquetador de textos en lenguaje natural basado en modelos de Markov.

1. Introducción

Una de las quejas más habituales de los alumnos de asignaturas teóricas en titulaciones aplicadas como Ingeniería Informática es *¿para qué me va a servir lo que me estás explicando?* En muchas ocasiones este tipo de quejas son injustificadas ya que se deben a la falta de capacidad del alumno para hacer un análisis global de la materia y comprender las implicaciones que puede tener un determinado concepto teórico en un problema aparentemente lejano. Precisamente la tarea del alumno es adquirir esa capacidad y nuestro trabajo como profesor consiste en, además de presentar el concepto teórico, hacerle ver esas conexiones que lo hacen necesario para una aplicación posterior.

Sin embargo, hay veces en las que esta queja puede tener fundamento. Son los casos en los que los programas de las asignaturas son diseñados en base a una bibliografía clásica [4], [5] y pueden contener conceptos interesantes desde un punto de vista teórico pero que no están vinculados con ningún otro concepto posterior ni con una aplicación práctica. Las asignaturas relacionadas con la materia troncal Teoría de Autómatas y Lenguajes Formales, incluida en las vigentes directrices propias del título de Ingeniero en Informática, suelen adolecer en ocasiones de este problema. Es habitual encontrarse en los temarios de estas asignaturas con algunos conceptos que, o bien son difícilmente aplicables en la práctica, o bien sí lo son pero dichas aplicaciones no son incluidas en el temario por considerarse que *no pegan con una asignatura teórica*.

En este trabajo nos fijaremos en uno de esos conceptos, la máquina de Moore, un modelo cercano al de los autómatas finitos y a su vez muy cercano a otro tipo de autómata, el de Markov, que puede dar bastante juego a la hora de plantear una práctica de curso muy interesante.

Sin embargo, nuestra motivación va un poco más allá de presentar una práctica más o menos interesante, nuestro verdadero interés se centra en un debate de mayor calado: cómo conseguir introducir en una asignatura de corte teórico, como es la de Lenguajes Formales y Autómatas, aspectos prácticos que complementen los conceptos teóricos. Consideramos que este planteamiento es de vital importancia en una titulación tan aplicada como lo es la Ingeniería Informática.

2. La máquina de Moore

La máquina de Moore es un modelo derivado de los autómatas finitos. Este tipo de autómata es el

modelo computacional más simple de los que se utilizan en el procesamiento de lenguajes.

2.1. Definición de autómata finito

Formalmente, un autómata finito determinista se define como la tupla $(Q,\Sigma,\delta,q_0,,F)$ donde Q es un conjunto finito de estados, Σ es un conjunto finito de símbolos denominado alfabeto de entrada, $\delta:Q\text{x}\Sigma\rightarrow Q$ es la función de transición, q_0 es el estado inicial y $F\subseteq Q$ es el conjunto de estados finales. La representación más usada a efectos de especificación para los autómatas finitos suele ser la gráfica (figura 1).

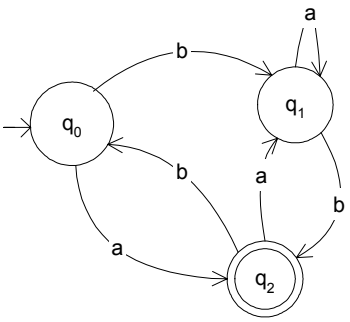


Figura 1: Autómata finito

La aplicación más común de los autómatas finitos en el ámbito del procesamiento de lenguajes es la implementación de reconocedores léxicos para lenguajes formales. Existen multitud de herramientas que permiten especificar un analizador léxico combinando expresiones regulares e instrucciones de un lenguaje de programación, y que dan como resultado una implementación de dicho reconocedor basada en autómatas finitos. Igualmente en la bibliografía se pueden encontrar distintos métodos de obtención de autómatas a partir de expresiones, que van desde los más abstractos planteados en forma de demostraciones de teoremas hasta los más aplicados expresados en forma de algoritmos, que incluso incluyen decisiones de diseño orientadas a reducir el coste computacional del proceso.

2.2. Definición de máquina de Moore

La máquina de Moore es un modelo muy cercano al de los autómatas finitos. De hecho tan solo se diferencia de un autómata en la capacidad que tiene de emitir una salida asociada al estado del autómata.

Formalmente una máquina de Moore se define como la tupla $(Q,\Sigma,\Delta,\delta,\gamma,q_0,)$ donde Q, Σ, δ y q_0 significan lo mismo que en el modelo de autómatas finitos, Δ es un conjunto finito de símbolos denominado alfabeto de salida y $\gamma:Q\text{x}\Sigma\rightarrow\Delta$ es la función de salida que calcula el símbolo emitido en cada estado. Al igual que los autómatas finitos, la representación gráfica es la más utilizada a la hora de describir las máquinas de Moore (figura 2).

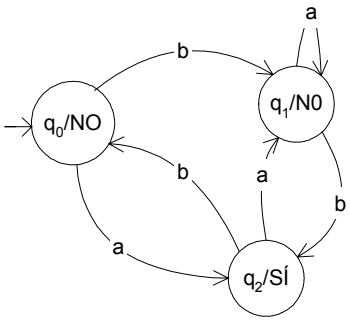


Figura 2: Máquina de Moore

En términos de descripción de lenguajes, los autómatas de las figuras 1 y 2 son equivalentes ya que ambos aceptan las secuencias de letras que conducen al estado q_2, Este ejemplo basta para mostrar que las máquinas de Moore son al menos tan potentes como los autómatas finitos. Es fácil ver que con el concepto de salida asociada al estado no sólo se pueden modelar los conceptos *final* y *no final* de los autómatas finitos sino que se pueden describir situaciones más complejas, ya que el alfabeto de salida puede ser tan amplio como se desee.

La máquina de Moore suele ser presentada en compañía de la máquina de Mealy en la que la salida se asocia a las transiciones del autómata en lugar de a los estados. Existen demostraciones de la equivalencia de ambos modelos, y se suele denominar máquina secuencial al modelo general que subyace tras ellas.

Una aplicación muy usual de las máquinas secuenciales es la de diseño de circuitos electrónicos, concretamente en situaciones en las que la simplicidad del problema a resolver hace

que no sea necesario incluir un microprocesador en el diseño del circuito.

2.3 La máquina de Moore y el procesamiento de lenguajes

La filosofía con la que están planteadas la mayoría de las asignaturas dedicadas a la Teoría de Autómatas y Lenguajes Formales se basa en mostrar la relación que existe entre formalismos de especificación de lenguajes (gramáticas y expresiones regulares) con sus correspondientes modelos de computación. Esta relación teórica es totalmente explotada en las asignaturas correspondientes a la materia troncal Procesadores de Lenguaje, en las que se aplican estos conceptos y se integran en una metodología de desarrollo.

Si es éste el planteamiento, como ocurre en nuestro caso, resulta bastante difícil justificar la inclusión de un tema dedicado a máquinas secuenciales en el temario de la asignaturas de la materia troncal Teoría de Autómatas y Lenguajes Formales (TALF), máxime si tenemos en cuenta que puede que se repitan los contenidos de las asignaturas dedicadas al diseño de circuitos.

Evidentemente, es de gran interés teórico mostrar la generalización que suponen las máquinas de Moore y de Mealy frente al modelo básico de los autómatas finitos, pero esa presentación sería didácticamente más convincente si se acompaña de una aplicación práctica. En este trabajo mostramos una aplicación de una variante de la máquina de Moore al procesamiento de textos en lenguaje natural. Esta aplicación, además de permitir diseñar una práctica de laboratorio interesante, proporciona al alumno una visión alternativa al procesamiento de lenguajes formales a través del tratamiento estadístico del lenguaje natural.

3. El modelo de Markov

El modelo de Markov se acerca mucho al modelo de la máquina secuencial de Moore, también se puede interpretar como una generalización de un autómata finito en la que cada estado tiene asociada una salida. La diferencia fundamental con el modelo de Moore está en la introducción de probabilidades tanto para las transiciones como para la emisión de la salida correspondiente a un estado. De manera que en términos simplistas podríamos decir que un modelo de Markov es una máquina de Moore probabilística.

Formalmente un modelo de Markov μ se define como la tupla (Q, Δ, π, A, B) donde:

- $Q=\{1,...,N\}$ es el conjunto de estados del modelo. Etiquetaremos los estados con números naturales y denotaremos con q_t el estado en el instante de tiempo t.
- $V=\{v_1,v_2,...,v_M\}$ es el conjunto de las posibles salidas o sucesos observables del modelo. Cada estado podrá emitir varias de estas salidas con una determinada probabilidad.
- $\pi=\{\pi_1,\pi_2,...,\pi_N\}$ es la distribución de probabilidad del estado inicial de forma que $\pi_i=P(q_1=i)$. Se cumple:
$$\Sigma_i \pi_i = 1$$
- $A=\{a_{11},...,a_{ij},...,a_{NN}\}$ es la distribución de probabilidad de las transiciones entre estados, de forma que:
$$a_{ij}=P(q_t=j|q_{t-1}=i)$$
Es decir la probabilidad de transitar al estado j condicionada a estar en el estado i en el instante de tiempo anterior. Para simplificar se denotará también con $P(j|i)$. Se cumple:
$$\Sigma_j P(j|i)=1$$
- $B=\{b_{11},...,b_{kj},...,b_{NM}\}$ es la distribución de probabilidad de los sucesos observables, de forma que:
$$b_{kj}=P(o_t=v_k|q_t=j)$$
Es decir la probabilidad de emitir la salida v_k condicionada a estar en el estado j. Para simplificar se denotará también con $P(v_k|j)$. Se cumple:
$$\Sigma_k P(v_k|j)=1$$

Pese a su aparentemente compleja definición, el modelo de Markov no es más que el modelo de Moore en el que los conceptos de estado inicial, transición etiquetada y salida se sustituyen, respectivamente, por probabilidad inicial, probabilidad de transición y probabilidad de emisión de salida.

Originalmente el modelo de Markov fue diseñado por Andrei A. Markov para construir un modelo de las secuencias de letras de las palabras en la literatura rusa. Pero posteriormente se adoptó como herramienta estadística de propósito general para describir procesos estocásticos.

Podemos adaptar la representación gráfica de los autómatas finitos y la máquina de Moore para

dar cabida a las probabilidades del modelo de Markov tal y como se muestra en la figura 3. Dicha figura representa un modelo muy simplificado de la evolución del tiempo atmosférico. Los estados denotan tres posibles situaciones: tiempo *bueno*, *malo* e *inestable*, y la salida observable son los fenómenos atmosféricos más probables en dichas situaciones: *sol*, *nubes* y *lluvia*.

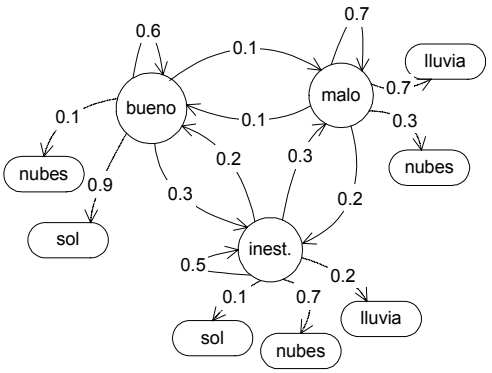


Figura 3: Modelo de Markov

3.1. Propiedades del modelo de Markov

Desde el punto de vista de la descripción de procesos estocásticos, el de Markov es un modelo realmente simple. De hecho, en la definición del modelo que hemos presentado hay implícitas dos simplificaciones o propiedades que marcan notablemente la filosofía con la que se debe utilizar este modelo. Estas dos características son:

- Propiedad del horizonte limitado: esta propiedad permite limitar la dependencia del estado actual con respecto a los anteriores, de manera que sólo se tengan en cuenta los n últimos de los estados anteriores. En los modelos de Markov de orden uno se asume que la dependencia se extiende sólo al estado anterior, por tanto:
  $P(q_t=j|q_{t-1}=i,q_{t-2}=k,...)= P(q_t=j|q_{t-1}=i)$
- Propiedad del tiempo estacionario: esta propiedad establece que la probabilidad de transitar de un estado a otro es la misma independientemente del instante de tiempo. Se expresa:
  $P(q_1=j|q_0=i)=...= P(q_t=j|q_{t-1}=i)=P(j|i)$

Obviamente, estas propiedades dan lugar a modelos que se aproximan a la realidad pero que no la formalizan por completo. En este sentido, la limitación más importante de los modelos de Markov es la incapacidad de modelar adecuadamente relaciones de larga distancia.

4. Procesando textos en lenguaje natural

Los modelos de Markov se han hecho muy populares en el campo del procesamiento del lenguaje natural [6]. Su simplicidad hace que los procesadores obtenidos sean bastante eficientes, y su carácter probabilístico permite que los modelos se aprendan de manera automática a partir de conjuntos de ejemplos.

En general los modelos de Markov se han aplicado con bastante éxito a todos aquellos problemas en los que se necesite asociar una etiqueta (o una categoría) a una palabra en el texto. En términos del modelo de Markov, la palabra sería la observación y la etiqueta que se le asocia sería el estado.

En este trabajo presentaremos la aplicación del modelo de Markov al problema del etiquetado gramatical. Con pequeñas modificaciones se puede adaptar el ejemplo a muy diversos problemas, en el último apartado de esta sección describiremos estos problemas que pueden dar pie a prácticas similares.

4.1. Etiquetado gramatical

El etiquetado gramatical consiste en asociar a cada palabra la categoría gramatical a la que pertenece, esta tarea suele ser una de las primeras etapas en cualquier sistema de procesamiento de textos. La mayor dificultad de este problema viene provocada por la ambigüedad que presentan numerosas palabras, que pueden desempeñar a diferentes funciones gramaticales. Por ejemplo la palabra *casa* puede ser nombre y verbo, desempeñará una función u otra dependiendo de las palabras que la rodeen, es decir su contexto. Esta ambigüedad hace que la solución al etiquetado gramatical no pueda hacerse simplemente consultando un diccionario electrónico y haya que acudir a soluciones más complejas que hagan uso de la información que proporciona el contexto de cada palabra.

Salida	El	perro	bebe	agua
Estado	DET	NOM	VER	NOM

Figura 4: Etiquetado gramatical

Una de estas soluciones pasa por construir un modelo de Markov en el que las palabras son las salidas y las categorías son los estados. La figura 4 muestra un ejemplo de etiquetado gramatical en estos términos, las etiquetas DET, NOM, VER se refieren a las categorías determinante, nombre y verbo respectivamente.

Las probabilidades de transición entre estados modelarán las secuencias de etiquetas posibles, mientras que las probabilidades de emisión de símbolos modelarán qué palabras pueden desempeñar una determinada función gramatical.

En la figura 5 se muestra un fragmento de un modelo de Markov que refleja las relaciones entre categorías gramaticales y palabras del ejemplo anterior.

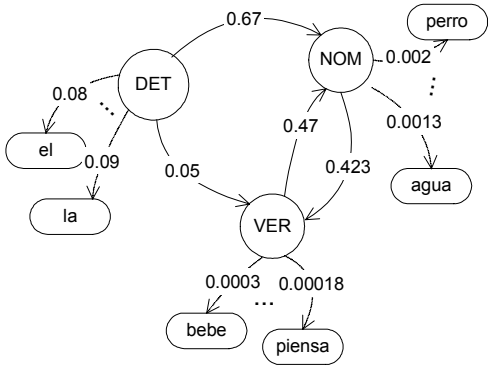


Figura 5: Fragmento del modelo de Markov

4.2. Obtención y aplicación del modelo

Con lo visto hasta ahora sabemos cómo modelar los distintos aspectos de un problema mediante un modelo de Markov, pero aún faltan por concretar dos cuestiones muy importantes, cómo construir y cómo aplicar un modelo de Markov. En el problema concreto que estamos tratando esto se traduce en dar una respuesta algorítmica a las siguientes preguntas:

- ¿Cómo obtener un modelo de Markov que se ajuste a las relaciones reales entre etiquetas gramaticales y palabras?
- Dada una frase sin etiquetar ¿cuál es la etiqueta más probable para cada palabra de la frase según el modelo de Markov?

Para responder a la primera pregunta hay que estimar las probabilidades de transición entre estados $P(i|j)$ y las de emisión de palabras $P(v_k|j)$. Para ello se hace uso de un corpus etiquetado, que no es más que un conjunto de textos en el que cada palabra tiene asignada la etiqueta apropiada. En base a la frecuencias de etiquetas consecutivas y las frecuencias de asignación de etiquetas a palabras en los ejemplos del corpus, las probabilidades anteriores se calculan fácilmente con las siguientes fórmulas:

$$P(i \mid j) = \frac{C(j,i)}{C(j)}$$

$$P(v_k \mid j) = \frac{C(v_k, j)}{C(j)}$$

Donde $C(j,i)$ es el número de veces que la etiqueta i aparece tras la etiqueta j, $C(j)$ es el número de veces que aparece la etiqueta j, y $C(v_k,j)$ es el número de veces que a la palabra v_k se le asigna la etiqueta j.

Se suele denominar entrenamiento al proceso de obtención del modelo a partir de los ejemplos contenidos en el corpus. Uno de los parámetros que marcan la calidad de una estimación es el tamaño del corpus, cuantos más ejemplos tengamos más fiables serán las probabilidades estimadas.

La segunda de las preguntas es la que nos permite solucionar el problema original, es decir determinar cuál es la categoría gramatical más probable de cada palabra de una frase. Aplicando ciertas transformaciones y simplificaciones se demuestra que el mejor etiquetado (el más probable) es aquel que maximiza la siguiente expresión:

$$\prod_{i=1,n} P(w_i \mid t_i) P(t_i \mid t_{i-1})$$

En dicha expresión el índice i indica la posición de cada palabra dentro de la frase (en la definición original del modelo se corresponde con el instante de tiempo i), t_i y t_{i-1} son respectivamente las etiquetas asignadas a las palabras que ocupan la posición i e $i-1$ en la frase y w_i es la palabra que ocupa la posición i.

Para calcular el mejor etiquetado basta con generarlos todos y quedarse con aquel que maximice la fórmula anterior. Utilizando técnicas de programación dinámica, el algoritmo de Viterbi [6] consigue el mismo resultado sin tener que generar todas las posibilidades, lo que redunda de forma significativa en la eficiencia del proceso.

4.3. Otras aplicaciones

Estas ideas se han aplicado con éxito a muchos problemas de procesamiento de textos en lenguaje natural. En general cualquier problema en el que haya que asignar una etiqueta a una palabra o a un conjunto de palabras se puede adaptar para expresarlo en términos de un modelo de Markov. Algunos de ellos son:

- Reconocimiento de entidades con nombre: consiste en la identificación de nombres de personas, lugares, organizaciones.
- Análisis sintáctico superficial: Consiste en identificar estructuras sintácticas en una frase sin realizar el análisis sintáctico completo.
- Desambiguación de significados: consiste en determinar el significado apropiado de una palabra polisémica.
- Extracción de información: consiste en identificar los fragmentos más informativos de un texto conociendo de antemano el tema sobre el que trata.

En general estos problemas son complejos, no obstante con las debidas simplificaciones todos pueden ser apropiados para una práctica similar a la presentada en este trabajo.

5. Planteamiento de la práctica

Ya hemos presentado los conceptos necesarios para plantear la práctica, tan sólo nos falta decidir cómo presentar dichos conceptos a los alumnos.

En primer lugar hay una serie de contenidos que son más apropiados para una clase de tipo teórico, en general todos aquellos que tengan que ver con la definición formal del modelo. No serán conceptos difíciles de presentar ya que a base de pequeñas ampliaciones se puede llegar del modelo básico de autómatas finitos, bien conocido por los alumnos, hasta el modelo de Markov pasando a través de la máquina de Moore tal y como hemos hecho en la presentación.

Tomando, por tanto, como punto de partida el conocimiento teórico por parte de los alumnos del modelo de Markov y de los algoritmos de entrenamiento (estimación de las probabilidades del modelo) y aplicación (Viterbi), lo que queda por presentar es la arquitectura del sistema a implementar y los recursos y herramientas que van a utilizarse.

5.1. La arquitectura

La figura 6 muestra los distintos elementos que constituyen el sistema a implementar. Cabe destacar que tanto el proceso de entrenamiento como el de aplicación contienen un componente importante de tratamiento de textos ya que todas sus entradas y salidas serán ficheros de texto. Esto hace que la práctica no sea sólo interesante por aplicar conceptos vistos en teoría, sino también por el tipo de herramientas que habrá que utilizar para desarrollarla, que deberán incluir aspectos como expresiones regulares que faciliten el tratamiento de estos textos. De esta forma tanto a través del objeto de aplicación como de los medios de implementación se aplican conceptos relacionados con la teoría de autómatas y lenguajes formales.

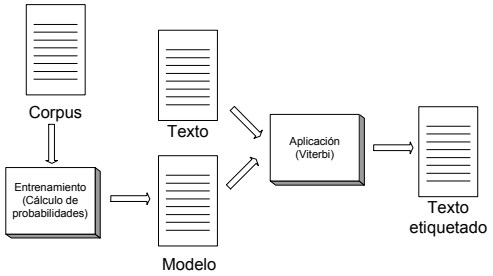


Figura 6: Arquitectura

Una vez propuesta la arquitectura del sistema el siguiente paso será decidir qué recursos y herramientas son necesarios para realizarla. La presentación de la práctica contendrá una propuesta de implementación, no obstante el problema es lo suficientemente abierto y flexible como para invitar a los propios alumnos a que investiguen y busquen otras alternativas. Con unas breves indicaciones y la curiosidad innata en algunos alumnos, en Internet se pueden encontrar con facilidad este tipo de recursos.

5.2 Recursos y herramientas

Los alumnos dispondrán de los siguientes recursos para el desarrollo de la práctica:

- Corpus etiquetado
- Implementación del algoritmo de Viterbi

El primero es imprescindible ya que para obtener unos resultados un poco satisfactorios se requiere un volumen de datos de entrenamiento considerable, lo que hace inviable que el alumno pueda construirse su propio corpus. Este tipo de recursos se pueden encontrar en Internet [7], se proporcionará uno junto con el enunciado, y se invitará a los alumnos a que busquen otras alternativas por su cuenta.

El algoritmo de Viterbi no es muy complejo y es viable que el alumno lo desarrolle por su cuenta a partir de las explicaciones en clase. No obstante existen diversas implementaciones genéricas de los modelos de Markov que ya incorporan dicho algoritmo y evitan ese trabajo [3]. En este sentido consideramos más interesante que el alumno realice el trabajo de transformación de textos para adaptar su entrada al formato que exija una determinada implementación del modelo de Markov, que hacerle desarrollar un algoritmo que desde el punto de vista práctico está más cercano a la asignatura *Estructuras de Datos y Algoritmos*.

La última decisión que queda por tomar es el lenguaje de programación. Dada la importancia de la componente textual, Perl es una magnífica opción debido a su potencia en el manejo de cadenas y expresiones regulares.

Cabe puntualizar que los resultados obtenidos por los alumnos al desarrollar esta herramienta no pasarán de ser un prototipo muy elemental. Las herramientas reales hacen uso de técnicas más complejas y refinadas para obtener resultados más satisfactorios. Esto no resta nada al interés didáctico de la práctica, es más el conocimiento de estas limitaciones por parte de los alumnos puede incitar a aquellos más interesados en el tema a continuar investigando en busca de mejoras al modelo básico.

6. Contexto académico y planificación temporal

En esta sección completaremos la presentación de esta propuesta docente enmarcándola en el contexto de la titulación de Ingeniería Informática de la Universidad de Sevilla [2] y proponiendo una planificación temporal para el desarrollo de la práctica a lo largo de un cuatrimestre.

6.1. Las asignaturas de la materia troncal TALF en Sevilla

El título de Ingeniero en Informática dedica dos asignaturas al desarrollo de los conceptos relacionados con la Teoría de Autómatas y Lenguajes Formales, una en segundo curso denominada *Lenguajes Formales y Autómatas* (LFA) y otra en tercero llamada *Ampliación de Lenguajes Formales y Autómatas* (ALFA), ambas de 4,5 créditos. La práctica presentada en este trabajo está diseñada para que se desarrolle en las sesiones de laboratorio de la asignatura ALFA, ya que requiere unos conocimientos mínimos de procesamiento de lenguajes que el alumno habrá adquirido en la asignatura LFA.

La asignatura LFA está dedicada a la presentación de los conceptos básicos de la Teoría de Lenguajes Formales. En ella el alumno tomará contacto con la idea de lenguaje y con los formalismos generativos y de reconocimiento asociados a las dos familias más simples de lenguajes de la jerarquía de Chomsky, los lenguajes regulares y los independientes del contexto. Desde el punto de vista práctico, se utilizarán herramientas clásicas de procesamiento de lenguajes (flex, bison) que aplican directamente expresiones regulares para especificar los aspectos léxicos de un lenguaje y gramáticas independientes del contexto para los aspectos sintácticos.

En el aspecto teórico, la asignatura ALFA se centra en la ampliación de los conceptos vistos en LFA abarcando los temas de máquinas secuenciales y probabilísticas [1], límites teóricos de lenguajes regulares e independientes del contexto, lenguajes dependientes del contexto y lenguajes con estructura de frases.

La práctica propuesta para la asignatura ALFA cubre varios objetivos al mismo tiempo. Por un lado implementa uno de los aspectos vistos en teoría ya que el modelo de Markov se puede considerar una extensión de los autómatas probabilísticos. En segundo lugar presenta al alumno una aplicación que trata textos en lenguaje natural, de una naturaleza totalmente distinta al

tipo de problemas que resolvió en la asignatura LFA, centrada en el tratamiento de lenguajes formales. Por último las herramientas que se utilizan en la práctica, en concreto el lenguaje Perl, proporcionan al alumno una alternativa de desarrollo de procesadores de lenguajes que para ciertas aplicaciones puede resultar más adecuada que las herramientas clásicas basadas en expresiones regulares y gramáticas independientes del contexto.

6.2. Planificación temporal

La práctica está pensada para que se desarrolle a durante un cuatrimestre. Con idea de planificar el trabajo de los alumnos se dividirá en siete sesiones de laboratorio, de las cuales las tres primeras se dedicarán a presentar el lenguaje Perl, nuevo para ellos, y las cuatro últimas a desarrollar distintos módulos de la práctica. En concreto las sesiones son:

- Introducción al lenguaje Perl
- Expresiones regulares en Perl
- Tablas asociativas en Perl
- Cálculo de frecuencias de etiquetas y palabras a partir del corpus
- Ejecución y prueba de la implementación del modelo de Markov disponible
- Obtención del modelo de Markov a partir de las frecuencias
- Aplicación del modelo a un texto sin etiquetar

Evidentemente algunos de los problemas propuestos son de una cierta envergadura y será muy difícil completar su implementación en una sola sesión. El montaje y prueba final de la aplicación queda como trabajo individual del alumno que deberá ser presentado como práctica final de la asignatura.

7. Conclusiones

Se ha presentado una práctica de curso para una asignatura de Teoría de Autómatas y Lenguajes Formales. Las conclusiones más destacables de esta propuesta son las siguientes:

- La introducción de aspectos muy aplicados en el ámbito de una asignatura de enfoques tradicionalmente teóricos.
- La aplicación de conceptos avanzados sobre autómatas a un problema de procesamiento de lenguajes.
- La presentación al alumno de una aplicación que procesa textos en lenguaje natural, lo que le proporciona una visión alternativa al procesamiento clásico de lenguajes formales.
- La utilización de un lenguaje de programación como Perl que integra mecanismos de manejo de textos de una forma sensiblemente distinta a las herramientas usadas por el alumno en cursos anteriores.

Referencias

[1] M. Alfonseca, J. Sancho, M. Martínez. *Teoría de Lenguajes, Gramáticas y Autómatas.* Ediciones Universidad y Cultura. Madrid. 1987.

[2] Boletín Oficial del Estado. *Plan de Estudios de Ingeniero en Informática de la Universidad de Sevilla*. 18 de noviembre de 1997.

[3]T. Brants. *TnT an statistical Part-of-Speech Tagger*. http://www.coli.uni-sb.de/~thorsten/tnt/

[4] D. I. A. Cohen. *Introduction to Computer Theory*. John Wiley & Sons. New York. 1991.

[5] J. E. Hopcroft, R. Motwani, J. D. Ullman. *Introducción a la Teoría de Autómatas, Lenguajes y Computación*. Addison-Wesley, 2002.

[6]C. Mannig, H. Schütze. *Foundations of Statistical Natural Language Processing*, MIT Press. Cambridge. 1999.

[7]G. Sampson. *Downloadable research resources, Corpus Sussane*. www.grsampson.net

Una alternativa docente a la Máquina de Turing

Morales, Rafael Ramos, Gonzalo Vico, Francisco J. Triguero, Francisco
Dpto. de Lenguajes y Ciencias de la Computación
E.T.S. Ingeniería Informática
Universidad de Málaga
Bulevar Louis Pasteur 35, 29071-Málaga
e-mail: morales@lcc.uma.es

Resumen

En la docencia de la Informática Teórica se necesita definir un modelo de cálculo para introducir formalmente el concepto de calculabilidad, y los que se derivan de éste. Esta introducción se suele hacer en la asignatura Teoría de Autómatas y Lenguajes Formales (TALF), generalmente en segundo año del primer ciclo de Informática, y habitualmente escogiendo la Máquina de Turing (MT) como modelo de cálculo. Presentamos aquí una alternativa docente a la MT basada en utilizar como modelo de cálculo un lenguaje estructurado simple, el lenguaje WHILE. Esta alternativa tiene dos ventajas docentes importantes. La primera es que dicho modelo es más cercano a los conocimientos que ya posee el alumno, por lo que es más fácil su asimilación. La segunda es que las demostraciones de las proposiciones y teoremas relacionados son mucho más cortas y claras.

1. Introducción

Dentro de la Informática Teórica, la Teoría de la Calculabilidad (TC) es una de sus partes más importantes. En la docencia de la TC se suele presentar una dificultad para su asimilación por parte del discente de Informática debido, entre otros factores, al alto grado de abstracción de sus contenidos. Por ello es importante cualquier mejora que podamos introducir en la forma de impartirla.

Un concepto básico, quizás el más importante de toda la TC, es el de modelo de cálculo, que formaliza el concepto intuitivo de algoritmo. Su presentación se suele hacer en la asignatura Teoría de Autómatas y Lenguajes Formales (TALF), generalmente en segundo año del primer ciclo de Informática.

Tradicionalmente se utiliza la MT como modelo de cálculo. Sin embargo, esta elección no se apoya en sus bondades pedagógica. Los principales motivos de su uso extendido son: haber sido uno de los cuatro que se definieron casi simultáneamente en 1936 (λ-cálculo, funciones μ-recursivas, MT y máquina de Post), el enlazar adecuadamente con los autómatas descritos en la primera parte de TALF, gozar de una gran popularidad, incluso en ambientes no informáticos y en la mayor parte de la bibliografía habitual, aunque la razón de mayor peso es el imperativo legal.

Todo el que haya impartido TALF con la MT como modelo de cálculo sabe de sus inconvenientes docentes. Por una parte la MT queda muy alejada del conocimiento que posee el alumno de Informática en segundo año. Incluso cuando, con esfuerzo, comprende el modelo, lo ubica alejado del resto de los contenidos de la carrera, sin mucha conexión con la misma. Por otra parte las demostraciones con MT, debido a su "bajo nivel", suelen ser largas y engorrosas, lo cual dificulta aún más una buena asimilación por parte del alumno.

Para intentar solucionar estos inconvenientes nosotros hemos elegido como modelo de cálculo para impartir TALF un lenguaje estructurado simple, el lenguaje WHILE. Este modelo es, por una parte, más cercano a los conocimientos que ya posee el alumno, por lo que es más fácil su asimilación, por otra, las demostraciones de las proposiciones y teoremas de la TC son mucho más cortas y claras con él. Este modelo se introdujo en una asignatura optativa de segundo ciclo hace unos diez cursos y se utiliza en TALF

desde hace cinco. Nuestra experiencia confirma totalmente lo adecuado de dicha elección.

En este trabajo presentamos este modelo y justificamos su uso. Para ello a continuación mostramos la evolución de los modelos formales de cálculo. Después dedicamos un apartado a presentar el modelo "lenguaje WHILE" que proponemos. En el apartado cuatro mostramos un ejemplo de demostración con dicho modelo, para terminar con un quinto de discusión.

2. Evolución de los modelos formales de cálculo

Tras los modelos iniciales de Church, Kleene, Turing y Post que se definieron casi simultáneamente en 1936, han aparecido otros con diferentes orientaciones. Veamos cómo la aparición de estos modelos ha ido paralela al desarrollo de los ordenadores y los lenguajes de programación. En primer lugar citamos el modelo URM definido por Shepherdson y Sturgis [12] y usado como modelo en [7], que dió origen al modelo RAM de amplio uso [1]. En este modelo se evitan los tediosos recorridos por la cinta de la máquina de Turing, mediante la introducción de registros en los que se almacenan datos, sin importar su tamaño. Junto con esta forma de almacenamiento está un adecuado juego de instrucciones que incluye sumar uno, restar uno, copiar de un registro a otro, salto incondicional y salto condicional. Los modelos derivados de URM tienen un juego de instrucciones que se parece mucho a un ensamblador.

El siguiente paso en los modelos fue pasar de registros a variables, de forma que una "máquina" pasa a tener aspecto de programa. Uno de estos lenguajes se puede ver en libros mas recientes como [3,4]. Es un lenguaje no estructurado con variables de entrada, variables de uso interno y variables de salida.

Volviendo a la evolución de los lenguajes, en 1966 se publica el trabajo [2] sobre la posibilidad de escribir programas con solo dos reglas de formación: secuencia y repetición (aunque habitualmente se habla de tres reglas, la selección es deducible de la repetición). Señalar que en este trabajo se usan diagramas de flujo, que se traducen de forma directa a sentencias de un lenguaje de programación.

Este trabajo junto con otros desarrollos prácticos dio impulso a la programación estructurada y, en el ámbito de la Informática Teórica, surgieron modelos de cálculo basados en programación estructurada en los trabajos de Meyer y Ritchie [8,11]. El lenguaje derivado de los trabajos anteriores tiene sentencias de asignación y una única estructura de control: el bucle indefinido con control al principio, demostrándose su equivalencia con los demás modelos. Si bien esta propuesta inicial fue realizada hace unos 35 años con una sentencia de bucle definido, no queda claro en la bibliografía cuando se produce la ampliación del lenguaje con un bucle indefinido, que solo queda recogida en libros.

Otra versión moderna en una dirección similar se puede encontrar en [5]. No obstante el lenguaje utilizado se basa en listas y las sentencias básicas son de tipo LISP (cabeza y cola), lo cual se aleja de la formación de los alumnos de segundo curso.

Si tomamos la perspectiva de las funciones calculadas por cada modelo, podemos observar como una adecuada secuencia de modelos van refinando la clasificación de las funciones.

Las máquinas de Turing dan origen a las funciones Turing-calculables y el modelo no permite refinar esa clase. Esas funciones, mediante el modelo de Kleene [6], se refinan en recursivas primitivas y μ-recursivas. Consideremos ahora un lenguaje estructurado. A partir de una sentencia WHILE (que corresponde a la minimización o a la i-iteración [13]) se puede definir una sentencia FOR (que corresponde a la recursión primitiva o a la exponenciación [12]). Se podría pensar en el nivel de anidamiento de sentencias WHILE, pero esto no tiene interés pues ya sabemos, por el teorema de la forma normal de Kleene, que cada función calculable se puede expresar mediante un programa que se escribe usando asignaciones, estructuras FOR y una sola estructura WHILE. Pero sí tiene interés considerar el nivel de anidamiento de estructuras FOR, y esto supone un refinamiento de la clase de las funciones recursivas primitivas mediante una jerarquía infinita LOOP0, LOOP1, ..., LOOP*n*, ...

Veamos en el siguiente apartado una descripción del lenguaje WHILE tal y como aparece en [10].

3. El lenguaje WHILE

El lenguaje WHILE contiene sentencias para calcular la constante cero, el incremento, el decremento y la asignación, además de una estructura para el bucle indefinido. Dicho bucle funciona como un bucle de tipo *while*, por lo que los programas definidos en este lenguaje se denominan programas WHILE. Sólo se dispone de estas sentencias simples como primitivas, por lo que otras operaciones como la multiplicación, división, etc. serán implementadas como programas WHILE. Señalar también que este lenguaje no contiene instrucciones de entrada/salida, y que el único tipo de datos utilizado por un programa WHILE es el número natural, que se almacena en variables. Cada programa representa una función matemática.

Definimos la sintaxis del lenguaje WHILE en notación *BNF* de la siguiente forma:

Sea $G = (N, T, R, S)$ una gramática donde

T = { DO, OD, :, =, WHILE, ≠, +, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, X, ; }

N = { *programa*, *asignacion*, *sent*, *sentencias*, *identificador*, *numero* }

El conjunto de reglas R es:

1 *S*=<*programa*> ::= {(*n*,*p*,<*sentencias*>)|*n*,1≤*p*}
2 <*sentencias*> ::= <*sent*>
3 | <*sentencias*>;<*sent*>
4 <*sent*> ::= <*asignacion*>
5 | WHILE<*identificador*>≠0DO<*sentencias*>OD
6 <*asignacion*> ::= <*identificador*>:=0
7 | <*identificador*>:=<*identificador*>
8 | <*identificador*>:=<*identificador*>+1
9 | <*identificador*>:=<*identificador*>−1
10 <*identificador*> ::= X<*numero*>
11 <*numero*> ::= (1|2|3|4|5|6|7|8|9) {0|1|2|3|4|5|6|7|8|9}

Con esta sintaxis: si X1 y X2 son nombres de variables, entonces las expresiones X1:=0, X1:=X1+1, X1:=X1-1 y X1:=X2 son <sentencias> WHILE; si P y Q son <sentencias> WHILE, entonces la secuencia P;Q es válida; siéndolo también la expresión WHILE X1≠0 DO P OD. En este bucle se diferencian la cabecera (WHILE X1≠0 DO), el cuerpo P y la cola (OD) del bucle.

Un programa WHILE es una terna (*n*,*p*,P), consistente en dos números naturales y una secuencia de sentencias P.

Comencemos con la semántica del lenguaje diciendo que las variables que participan en la secuencia P son las variables del programa. El número *p* especifica que {X1,...,X*p*} son los nombres de dichas variables del programa, y el número *n* especifica cuántas de estas variables son de entrada, es decir X1,...,X*n*. La única variable de salida es X1. Cada línea del programa (*n*,*p*,P) contiene: una asignación, una cabecera de bucle *while* o bien una cola de bucle *while*.

Vamos a definir la función $f : \mathrm{N}^n \to \mathrm{N}$ calculada por un programa WHILE (con *n* variables de entrada). Partiendo de $(a_1, ..., a_n) \in \mathrm{N}^n$ llegaremos a determinar $f(a_1, ..., a_n)$:

Sea $(a_1, ..., a_n) \in \mathrm{N}^n$. A partir de estos valores de entrada, el vector de estado inicial es el vector $(a_1, ..., a_n, 0, ...,0) \in \mathrm{N}^p$ formado por esos valores y ceros (correspondientes a las variables que no son de entrada).

El cálculo comienza sobre este vector de estado y en la primera línea del programa, es decir, la iniciación de variables no se considera parte del cálculo. El cálculo procede secuencialmente de acuerdo con la estructura del programa y el significado de cada sentencia, así, un programa WHILE de la forma "P;Q" supone la ejecución primero de P y después (manteniendo los valores de las variables como resultan de ejecutar P) de Q. Como hemos comentado en la sintaxis hay dos tipos de sentencias: asignación y bucle.

La ejecución de una sentencia de asignación (X3:=0, X4:=X5, X2:=X3+1, X1:=X4-1) transforma el valor actual del vector de estado reemplazando una componente por cero, por el valor de otra componente, o bien por el sucesor o predecesor de otro componente, respectivamente. Esta transformación se considera un paso de cálculo. Por otro lado, la secuencia P en un bucle *while* de la forma "WHILE X1≠0 DO P OD" se ejecutará mientras la variable X1 no almacene el valor cero, si tal condición tiene lugar.

Si el programa nunca alcanza la última línea (condición de terminación del cálculo) entonces

calcula indefinidamente, en cuyo caso decimos que, para ese vector de entrada, la función diverge.

Por el contrario, si ejecuta la última línea, entonces el cálculo acaba. El valor calculado (la salida del programa) es el valor almacenado en el primer componente del vector de estado (`X1`). Ese es el valor $f(a_1, ..., a_n)$ de la función calculada por el programa.

Diremos que dos programa son equivalentes se calculan la misma función.

Ilustramos el modelo de programas WHILE con un programa equivalente al que calcula la sentencia IF (con su significado normal), de la forma `IF X≠0 THEN` P `FI`:

```
WHILE X1≠0 DO
    P;
    X1:=0;
OD
```

4. Una demostración

Usando el lenguaje WHILE, desarrollamos en este apartado, una función no computable. Para ello seguiremos el trabajo [9], completado con una demostración sencilla de la insolubilidad del problema de la parada.

Vamos a considerar aquí la clase de programas con una sola entrada (`X1`).

Definición: Sea (*l*,*p*,P) un programa WHILE. Se define la longitud de P como su número de líneas, contando cada asignación y cada cabeza de bucle como una línea.

Por ejemplo, el siguiente programa (*1*,*2*,P) tiene longitud 4.

```
X2:=X1;
WHILE X2≠0 DO
    X1:=X1+1;
    X2:=X2-1;
OD;
```

Definición: La función castor afanoso (*busy-beaver*), Σ, se define como sigue: Para cada *n*, Σ(*n*) es el mayor valor que puede almacenarse en la variable `X1` usando un programa de longitud exactamente *n* que acaba, cuando comienza con `X1=0`.

Explicación intuitiva: dado un valor para *n*, p.e. 15,

- escribir todos los programas de longitud 15,
- seleccionar los que acaban y poner como entrada `X1=0`,
- ver su salida y tomar el valor mayor.

Teorema: La función castor afanoso (Σ) es no calculable, es decir, no existe un programa WHILE que la calcule.

La demostración de este teorema precisa dos resultados previos:

Proposición 1: Para cada *n*, $\Sigma(n+1) > \Sigma(n)$.

Demostración: Sea P el programa que calcula Σ(*n*), esto es, P es de longitud *n* y con entrada `X1=0`, acaba y produce la mayor salida.

Sea P' el programa siguiente:

```
P
X1:=X1+1
```

Este programa tiene longitud *n*+1 y produce como salida Σ(*n*)+1, y puede haber algún otro programa de longitud *n*+1 que produzca una salida mayor, luego $\Sigma(n+1) \geq \Sigma(n)+1 > \Sigma(n)$.

Proposición 2: Si *f* es calculable mediante un programa de longitud *k*, entonces para cada *n* $\Sigma(n+k) \geq f(n)$.

Demostración: Sea P el programa de longitud *k* que calcula *f*. Sea P'el siguiente programa:

```
X1:=X1 + 1;
… (n veces)
X1:=X1 + 1;
P
```

Este programa tiene longitud *n*+*k*; con entrada `X1=0`, tras las *n* primeras líneas `X1` vale *n*, y tras la ejecución de P, que calcula *f*, produce como salida *f*(*n*). Luego tenemos un programa de longitud *n*+*k* cuya salida es *f*(*n*) y puede haber algun otro de esa longitud que produzca una salida mayor, luego $\Sigma(n+k) \geq f(n)$.

Ya estamos en condiciones de demostrar del teorema:

Supongamos que Σ es calculable. $f(x) = 2x$ es calculable. La composición de funciones calculables es calculable. Entonces $ß(n) = Σ(2n)$ será calculable mediante un programa P' de longitud k. Aplicando la proposición 2 a $ß(n)$, tenemos que para cada n: $Σ(n+k) \geq ß(n) = Σ(2n)$.

Tomando $n=k+1$, resulta $Σ(2k+1) \geq Σ(2k+2)$, que entra en contradicción con la proposición 1 (crecimiento estricto de la función Σ).

Veamos ahora la irresolubilidad del problema de la parada a partir de la no calculabilidad de la función castor afanoso.

Supongamos que el problema de la parada es resoluble. Para cada longitud n dada, solo hay un número finito de programas distintos. Como suponemos que el problema de la parada es resoluble, seleccionar los que acaban y obtener su salida. La mayor salida es Σ(n). Con lo cual la función castor afanoso sería calculable, en contra de lo que acabamos de demostrar; por tanto, el problema de la parada es recursivamente irresoluble. Nótese que una demostración directa de la irresolubilidad del problema de la parada (es decir, su predicado asociado es indecidible) requiere una gödelización de los programas y la correspondiente indexación de funciones.

5. Discusión

En este trabajo hemos analizamos la fuerte dependencia del modelo de cálculo en la docencia de la TC. La MT es el modelo más extendido en la programación de TALF. Opinamos que principalmente por razones históricas, dadas las dificultades que plantea su uso en el aula. Basándonos en nuestra experiencia docente destacamos dos puntos débiles: (1) el bajo nivel del modelo, y (2) su complejidad inherente.

Respecto a (1), el bajo nivel de la MT tiene dos consecuencias principales: (a) se separa excesivamente de los lenguajes de programación de alto nivel, lo que a menudo impide al alumno relacionar las propiedades de los modelos de cálculo con otras áreas de conocimiento. Conceptos tan cercanos como por ejemplo la MT universal y un compilador no suelen asociarse espontáneamente; (b) para asimilar el funcionamiento de la MT, el alumno frecuentemente se desvía en exceso de su naturaleza matemática y prefiere verlo como un sistema mecánico, situándolo más cerca de un autómata industrial que de un modelo formal. En relación a (2), la complejidad de la MT se refleja principalmente en las demostraciones de los principios incluidos en la TC. Demostraciones de teoremas como la equivalencia con otros modelos, la no calculabilidad del castor afanoso o el problema de la parada, o aún la definición de la MT universal, resultan largas y elaboradas en exceso, lo que perjudica tanto el seguimiento como la asimilación del teorema.

Los modelos basados en lenguajes estructurados solucionan en parte los problemas de la MT: la asimilación del modelo resulta fácil, ya que presenta muchos de los rasgos de los lenguajes de programación que el alumno conoce, lo que garantiza la conexión con muchos conceptos afines de otras áreas; por otra parte, la demostración de los teoremas se simplifica bastante, quedando mucho más legible y en una forma muy intuitiva, lo que reduce los tiempos de explicación y favorece la asimilación, ya que hacemos uso de un buen número de nociones de las que ya dispone el alumno, como podrían ser la invocación de subrutinas o el funcionamiento de un condicional múltiple.

La introducción del modelo WHILE presenta algunas características adicionales. Por un lado, el modelo es entendido inicialmente por el alumno como un lenguaje de programación de ordenadores, lo que constituye un inconveniente, ya que nos interesa resaltar su naturaleza matemática y su papel como modelo formal. No obstante, esta faceta puede ser utilizada para alcanzar tres objetivos adicionales: (1) la comprensión de que, en realidad, cuando se programa se está demostrando formalmente la calculabilidad de la función definida en las especificaciones; (2) la noción de que los lenguajes de programación actuales descansan en unas primitivas de cálculo muy elementales, y que cada elemento del lenguaje tiene una función de apoyo a la programación, sin alterar su capacidad de cálculo; concretamente, la demostración de que una versión extendida de WHILE, con muchas de las características de un lenguaje evolucionado, se obtiene a partir de la forma básica, enseña al alumno qué es en realidad esencial en la

definición del lenguaje; simultáneamente (3) se justifica el uso de algoritmos en pseudocódigo para demostrar teoremas, práctica muy extendida en algunas materias, especialmente en la primera mitad de la asignatura TALF. Por otro lado, WHILE permite una mejor comprensión de la godelización de programas y por tanto de la indexación de funciones, pudiéndose además ilustrar con ejemplos de desarrollo completo la obtención de índices para funciones familiares.

En definitiva, hemos presentado una alternativa a la MT como modelo de cálculo dentro de la docencia de TALF. Dicho modelo está al final de la evolución teórica de los modelos de cálculo, basándose en un lenguaje estructurado simple, el lenguaje WHILE. Este modelo presenta ventajas para la docencia. Por una parte es más cercano a los conocimientos que posee el alumno en el momento curricular en que se imparte, percibiéndose además como más cercano al resto de los contenidos de la carrera de Informática. Por otra parte hace posible, como hemos mostrado, realizar demostraciones de la TC mucho más cortas y legibles. Además, permite una inclusión fácil del no determinismo, a partir de una sentencia "either" de ramificación binaria.

Por todo ello, basándonos en nuestra experiencia positiva de varios años de utilización, aconsejamos, que si bien se explique a los alumnos en qué consiste la MT por ser el modelo más conocido, para el desarrollo y las demostraciones de la TC se utilice el lenguaje WHILE como modelo teórico de cálculo.

Enlazando con la Teoría de la Complejidad señalamos que el modelo MT es adecuado para el estudio de los recursos espacio y tiempo. El lenguaje WHILE no es adecuado para el estudio del recurso espacial de forma directa (a una variable se le puede asignar un valor arbitrario) y habría que considerar el logaritmo del mayor valor almacenado en una cualquiera de las variables del programa. Sin embargo, sí es adecuado de forma directa para el estudio del recurso tiempo, ya que el aumento de complejidad en tiempo al pasar de un programa WHILE a su correspondiente MT es solo polinómico, operación para la que son estables la mayoría de las clases de complejidad, y en particular las que usualmente se introducen en ese momento de la formación del alumnos, como son P y NP.

Referencias

[1] Aho, A.V., Hopcroft J.E. & Ullman, J.D. *The Design and Analysis of Computer Algorithms*. Addison-Wesley. 1974.

[2] Böhm C. & Jacopini G. Flow Diagrams, Turing Machines and Languages with only two Formation Rules. *Communications of the ACM*, **9**, 5, pp. 366-371, 1966.

[3] Davis M. & Weyuker E. *Computability, Complexity, and Languages: Fundamentals of Theoretical Computer Science*. Academic Press, 1983.

[4] Davis, M.D., Sigal, R. & Weyuker, E.J. *Computability, complexity and languages*. Academic Press. 1994.

[5] Jones, N. *Computability and Complexity. From a Programming Perspective*. MIT Press, 1997.

[6] Kleene S.C. General Recursive Functions of Natural Numbers. *Mathematicsche Annalen*, **112**, 5, pp. 727-742, 1936.

[7] Machtey, M. & Young, P. *An introduction to the General Theory of Algorithms*. Elsevier North-Holland Inc. 1978.

[8] Meyer A.R. & Ritchie D.M. The complexity of loop programs. *Proc. ACM Nat. Meeting*, pp. 465-469, 1976.

[9] Morales-Bueno, R. *Noncomputability is easy to understand.* Comm. ACM, **38**, pp. 116-117, 1995.

[10] Morales-Bueno, R.; Fortes, I.; Mora L. & Triguero, F. *Two classical theorems revisited*. Bull. EATCS, **71**, pp. 204-215, 2000.

[11] Ritchie, D.M. *Program structure and computational complexity*. Doctoral Dissertation. Harvard University. 1968.

[12] Shepherdson, J. & Sturgis, H. Computability of recursive functions. *Journal of the ACM*, **10**, 2, pp. 217-255. 1963.

[13] Sommerhalder R. & van Westrhenen S.C. *The Theory of Computability: Programs, Machines, Effectiveness and Feasibility*. Addison-Wesley. 1988.

Ingeniería del software

Incorporando Extreme Programming como Metodología de Desarrollo en un Laboratorio de Sistemas de Información

Patricio Letelier, José Hilario Canós
Depto. de Sistemas Informáticos y Computación
Universidad Politécnica de Valencia
46022 Valencia
e-mail: letelier@dsic.upv.es
jhcanos@dsic.upv.es

Resumen

Este trabajo describe nuestra experiencia en el uso de Extreme Programming como metodología de desarrollo de software en una asignatura de quinto curso en la Titulación de Ingeniero en Informática en la Facultad de Informática de la Universidad Politécnica de Valencia. Se explica es esquema de trabajo utilizado, incluyendo el método docente y el método de evaluación. Por el interés que han despertado en la actualidad las metodologías ágiles pensamos que nuestra experiencia y recomendaciones pueden ser valiosas para asignaturas similares en otras universidades.

1. Introducción

El objetivo de este trabajo es presentar la definición y los resultados de una experiencia docente llevada a cabo durante el primer cuatrimestre del año académico 2002-2003 en una asignatura de quinto año de la Facultad de Informática de la Universidad Politécnica de Valencia (UPV). El trabajo se enmarcó dentro del contexto de programas de ayuda a la mejora de la enseñanza del Proyecto EUROPA [1]. La asignatura objeto de esta experiencia se denomina Laboratorio de Sistemas de Información[1] (LSI). LSI es una asignatura optativa de quinto año en la Facultad de Informática de la UPV para la Titulación de Ingeniero en Informática, y cuenta con una asignación de 6 créditos, todos ellos son créditos de laboratorio.

En cuanto a contenidos, el objetivo de LSI es que el alumno sea capaz enfrentar un proyecto de desarrollo de software utilizando un enfoque de ingeniería del software desde la perspectiva del triángulo del éxito en desarrollo de software: notación-herramienta-proceso [2]. Esto se materializa mediante la realización de casos de estudio. Algunos de ellos son relativamente pequeños (resueltos en un par de sesiones), trabajando con herramientas CASE (CASE System Architect y Rational Rose) y utilizando la notación UML. La parte central de la asignatura es la realización de un caso de estudio correspondiente al desarrollo de un proyecto siguiendo una metodología. En los años anteriores, se utilizaba exclusivamente el Proceso Unificado de Desarrollo de Software [3] y su versión comercial llamada Rational Unified Process[2] (RUP). En vista del creciente interés que han generado las metodologías ágiles[3] decidimos embarcarnos en una experiencia utilizando una metodología ágil, y después de un breve estudio escogimos Extreme Programming[4] (XP) [4].

LSI lleva implantada 5 años y ha tenido históricamente una matrícula de alrededor de 40 alumnos. Las clases se imparten en un laboratorio que dispone de 20 puestos de trabajo acondicionados para 40 alumnos. Los alumnos matriculados se distribuyen en 2 grupos, cada uno de los cuales asiste a dos sesiones semanales de 2 horas cada una.

La motivación de este trabajo está basada en el convencimiento de que el esquema de

[1] http://www.dsic.upv.es/asignaturas/facultad/lsi/

[2] www.rational.com

[3] www.agilealliance.org

[4] www.extremeprogramming.org

implantación que hemos seguido y los resultados obtenidos pueden ser útiles para asignaturas similares en otras universidades.
El resto de este artículo está organizado como se describe a continuación. En la sección 2 se explica el interés por utilizar XP en el contexto de LSI. En la sección 3 se describe el plan de actividades y el método docente seguido. En la sección 4 se detalla el método de evaluación utilizado. La sección 5 presenta los resultados obtenidos. Finalmente, en la sección 6 se establecen las conclusiones del trabajo y acciones futuras.

2. XP en LSI

LSI ha ido evolucionando desde un laboratorio en el cual se desarrollaban problemas guiados por el profesor y utilizando herramientas CASE hacia un esquema de trabajo en el cual ha ganado protagonismo la recreación de un proyecto de desarrollo de software mediante trabajo en equipos. En la actualidad se dedica al proyecto alrededor del 70% del tiempo de la asignatura y del correspondiente esfuerzo del alumno.
Por un lado al alumno se le permite tener una experiencia más cercana a la realidad respecto de la ingeniería de software. Pero además se potencian una serie de habilidades personales que por lo general no son cubiertas en otras asignaturas de la titulación, como por ejemplo: trabajo en equipo, desempeño de un rol con determinadas responsabilidades, organización de su tiempo mediante planificación de tareas, negociación respecto de los objetivos a cumplir, participación en reuniones de seguimiento, presentación de resultados, etc. Aunque con RUP habíamos conseguido abordar satisfactoriamente estos dos aspectos, teníamos la impresión de que con una metodología ágil podíamos obtener aún mejores resultados en cuanto a potenciar dichas habilidades personales.
Sin embargo, no queríamos renunciar totalmente al uso de una metodología no ágil (llamadas peyorativamente "metodologías pesadas") con el fin de ofrecer una visión más global en cuanto a estrategias de desarrollo.
Así pues, decidimos mantener un par de equipos con RUP y establecer 5 equipos con XP. Escogimos XP por ser una de las más populares metodologías ágiles y una de las que se dispone mayor cantidad de material, tanto en la web como en libros recientemente publicados. Por las restricciones temporales de la asignatura es imposible que el alumno trabaje con metodologías (en dos proyectos) con suficiente detalle durante un cuatrimestre. Para compensar esta posible visión parcial del alumno se establecieron presentaciones introductorias de ambas metodologías con un examen escrito asociado. Además se intentaría realizar comentarios comparativos durante el transcurso del proyecto. Finalmente este último aspecto se vio apoyado porque al comenzar el proyecto detectamos que era necesario que los grupos XP presentaran a la clase sus resultados parciales incluyendo una demo de su producto. En XP no nos pareció adecuado evaluar según los artefactos producidos puesto que éstos eran mínimos y principalmente era el código. Así, se decidió que tanto los equipos RUP como los XP realizarían presentaciones al finalizar cada iteración, además de algunas adicionales de seguimiento.

3. Plan de Actividades y Método Docente

En esta sección nos centraremos en la planificación de actividades relacionadas directamente con el proyecto de desarrollo y particularmente las que afectaban a los equipos XP. En la web de la asignatura puede verse el detalle de todas las actividades realizadas durante el cuatrimestre.
El alumno de LSI tiene 4 horas de clases semanales, repartidas en dos sesiones de 2 horas cada una. El primer día de clases se establecieron los equipos de desarrollo, cada uno de ellos con 6 o 7 integrantes. Los roles asignados fueron: Jefe ("Big Boss"), Encargado de Pruebas ("Tester", encargado sólo de pruebas funcionales) y a la vez Controlador ("Tracker") y Programador. El profesor desempeñó el rol de Preparador ("Coach") para todos los equipos. Normalmente se dispone de unas 24 sesiones efectivas de clases durante el cuatrimestre.
Obviamente, para realizar un proyecto de una envergadura interesante teníamos que renunciar a poder completarlo, pero no es grave puesto que nuestro objetivo se centra más en introducir al alumno en la dinámica del proceso, principalmente el trabajo basado a iteraciones. Así pues se establecieron a priori 3 iteraciones, cada una de 3 semanas (para RUP fueron 1 iteración de

elaboración y 2 de construcción). Más un primer hito de 2 semanas para introducirse a la metodología y para establecer las historias de usuario (en el caso de RUP esto constituyó la iteración de Inicio).
El proyecto abordó la construcción de un sistema de información para una tienda virtual. Sólo se realizó una muy breve presentación del caso de estudio, sin entregar ningún tipo de información asociada. Se planteó que la situación actual del área problema necesitaba de una solución global, incluyendo todos los aspectos de la empresa, tales como: logística, ventas, almacén , transporte, personal, marketing, facturación, etc. Posteriormente, y de acuerdo a negociaciones con el cliente, se acotó el trabajo a los subsistemas de ventas y almacén (en los cuales ya se trabajaba con historias de usuario). Finalmente se consiguió una versión operativa del subsistema de ventas y parcial para el subsistema de almacén.
A continuación se indican las actividades XP realizadas durante el cuatrimestre y la planificación estimada, corregida con algunas pequeñas variaciones que se produjeron en la implantación. No se incluyen las reuniones o sesiones XP realizadas fuera del horario de clases.

Nro. Sesión	Actividad relacionada directamente con el Proyecto de Desarrollo
1	Establecimiento de Equipos Arranque oficial del proyecto
3	Presentación: "Introducción a metodologías ágiles y XP"
4	Control escrito de RUP y XP
7	Presentación de Historias de Usuario y Plan de 1era Iteración
12	Presentación Resultados 1era Iteración y Plan de 2da Iteración
14	Presentación Seguimiento 2da Iteración y Sesión XP
16	Sesión XP
17	Sesión XP
18	Presentación Resultados 2da Iteración y Plan 3era Iteración y Sesión XP
19	Sesión XP
20	Sesión XP
21	Presentación Seguimiento 3ra Iteración y Sesión XP
22	Sesión XP
24	Entrega de Resultados y Evaluación de 3era Iteración

4. Método de Evaluación

En los últimos dos años, en LSI se ha venido aplicando un sistema de evaluación continua multicriterio [5]. Todo el trabajo realizado por los alumnos se evalúa una vez terminada cada actividad realizada. Esta evaluación se realiza durante la sesión de clases, proporcionando los criterios para que el alumno no sólo entienda su evaluación, sino que conozca en el momento oportuno los aspectos en los cuales tiene deficiencias. Para aminorar el esfuerzo en corrección y revisión que esto supondría, se utilizan varias prácticas de evaluación, entre ellas: auto-evaluación, evaluación cruzada entre alumnos y evaluación realizada por los jefes de equipo. Así, el trabajo en la asignatura comprende alrededor de 16 actividades, siendo evaluadas todas ellas con el mismo peso. Entre dichas 16 actividades, más de la mitad son actividades directamente asociadas al proyecto de desarrollo. La evaluación de las actividades aporta el 80% de la nota final. El 20% restante corresponde al examen final (un examen que incluye alrededor de 15 preguntas con respuestas cortas). Las dos últimas sesiones de la asignatura se dedican a un seminario con presentaciones de trabajos voluntarios. Para los alumnos que realizan y presentan un trabajo pueden significar hasta un punto adicional a sumar en su nota final, siendo requisito necesario para quienes aspiren a matrícula de honor (suelen otorgarse al menos una).
El sistema de evaluación continua había sido muy satisfactorio, contando con una acogida favorable de los alumnos (aunque al principio parecen sorprendidos, sobre todo por el hecho que ellos deben participar activamente en la evaluación). El hecho de no tener que acumular para fines del cuatrimestre un montón de documentación y trabajos para revisar, el proporcionar a los alumnos una oportuna valoración de su trabajo, y finalmente el mantener al alumno en un ritmo regular de dedicación a la asignatura, habían merecido el esfuerzo asociado la mayor cantidad de evaluaciones.
La introducción de XP como metodología de desarrollo a primera vista no tenía por qué afectar el sistema de evaluación. Sin embargo, tempranamente nos dimos cuenta que debíamos hacer algunos ajustes. En RUP la evaluación se

centraba en artefactos generados, pero en XP al utilizar mucho menos artefactos y siendo el código el principal de ellos, no nos parecía lo más adecuada seguir con dicha estrategia de evaluación. Decidimos así pasar a un esquema de con presentaciones, en las cuales cada equipo y cada integrante del equipo informaba es estado del proyecto. El jefe informaba de la iteración y su planificación (historias de usuario y tareas abordadas) y de las incidencias ocurridas. El encargado de pruebas y controlador comentaba las pruebas establecidas y las desviaciones en cuanto a las estimaciones. Los programadores hacían una demo del estado del producto. Se hizo una presentación al finalizar cada una de las iteraciones y un par de presentaciones de seguimiento. Al finalizar cada presentación se otorgaban dos calificaciones, una externa asociada a la presentación propiamente tal y a la calidad del resultado parcial obtenido, y otra interna referente a la apreciación del profesor respecto del funcionamiento del grupo y a los objetivos preestablecidos. La primera de estas evaluaciones era una nota igual para cada integrante del equipo. La segunda era una nota que se multiplicaba por el número de integrantes del equipo obteniendo una cifra que se le informaba al jefe para que éste la distribuyera entre los integrantes del equipo. Esto resultó efectivo de cara a ajustar y premiar de acuerdo con las diferencias individuales (juzgadas por el jefe de equipo). Adicionalmente, al final del curso cada miembro del equipo tuvo también la posibilidad de evaluar a sus compañeros y particularmente al jefe. Para los dos equipos utilizando RUP se aplicó el mismo esquema de presentaciones, pero ellas se enfocaron más hacia la revisión y seguimiento de los artefactos utilizados. Con todo esto, se consiguieron 20 calificaciones, eliminándose las 3 peores. Esto último proporciona la flexibilidad necesaria de cara a no forzar en la práctica una asistencia casi obligatoria a todas las sesiones.

Así, con estos ajustes conseguimos mantener nuestra estrategia de evaluación continua y a la vez le dimos un impulso adicional a la participación activa del alumno.

5. Resultados de la Experiencia

En esta sección se resumen los resultados obtenidos haciendo hincapié en los aspectos que puedan servir para experiencias similares y que contribuyan a mejorar nuestro propio esquema de implantación.

5.1 Observaciones Generales

La experiencia de presentaciones y evaluación repartida por cada jefe de equipo resultó interesante y efectiva. Aunque, como se podía suponer, los alumnos en general se asignaron calificaciones buenas. Las evaluaciones desde el jefe hacia los otros miembros jugaron su rol de premio-castigo que no causó problemas destacables en el clima de trabajo en el equipo. La calificación final que recibieron los jefes de parte de los otros integrantes fue alta, lo cual confirmó la impresión de buen trabajo en los equipos XP.

Hay que destacar, que por primera vez (siendo este año el cuarto de aplicación de trabajo en equipos y el tercero trabajando con RUP) se produjeron roces personales entre integrantes de equipos (en los equipos XP). Esto tiene su clara explicación en el hecho que los diferentes ritmos de trabajo y capacidades quedan en evidencia en el esquema de trabajo XP, el cual exige una mayor interacción, participación y protagonismo de los integrantes. Sin embargo, en todos estos casos la situación se apaciguó o al menos se llegó a acuerdos prácticos de cara a los objetivos del equipo. Esto le dio una perspectiva de realismo no esperada al caso de estudio y como tal fue asumida y aprovechada como experiencia.

5.2 Opinión del Alumno

Se realizaron las encuestas docente oficiales pero aún no se dispone de los resultados. Sin embargo, al final del curso el profesor realizó una encuesta anónima en la cual a cada alumno se le pidió que escribiera sus comentarios respecto al método docente, al de evaluación y al enfoque del proyecto de desarrollo. Mayoritariamente las valoraciones fueron muy positivas en estos aspectos. Además, los miembros de equipos XP tenían que opinar acerca de su experiencia con XP usando el siguiente esquema:

1. Para cada práctica de XP debían poner una nota entre 1 y 10 para calificar su grado de aplicación en el proyecto, el grado de dificultad para aprenderla y aplicarla, y la importancia que le atribuían para mejorar la calidad del producto obtenido.
2. Hacer una valoración global de XP.

Práctica XP	Grado de aplicación en el proyecto	Dificultad para aprenderla y aplicarla	Interés que tiene para mejorar el producto final
El juego de planificación	9	4	9
Entregas pequeñas	9	3	9
Metáfora	5	5	6
Diseño simple	8	5	8
Pruebas	7	8	9
Refactoring	6	8	8
Programación en parejas	7	4	8
Propiedad compartida del código	8	6	8
Integración continua	8	5	9
Semana de 40 horas[5]	8	6	8
Cliente en el sitio de desarrollo	7	5	9
Estándares de programación	5	6	8

La encuesta en su parte XP fue contestada por 25 alumnos. Como era más o menos previsible, las prácticas que se cumplieron en menor grado fueron: metáfora (pensamos que se debe principalmente a lo imprecisa que es su definición), refactoring y estándares de programación. En cuanto al grado de dificultad para aprender y aplicar las prácticas XP, las pruebas y el refactoring destacan como las más difíciles para los alumnos. En general, todas las prácticas XP resultaron de interés para el alumno respecto del resultado obtenido (quizás la única excepción fue la práctica Metáfora, por la misma razón mencionada antes). Finalmente, la valoración global de XP fue muy positiva.

[5] Alrededor de 12 horas semanales en el caso de LSI (incluyendo las 4 horas de clases)

5.3 Mejoras y Recomendaciones

Aunque la experiencia ha resultado positiva, como es natural hemos detectado algunas mejoras posibles, y que a su vez sirven como recomendaciones para quienes se embarquen en una experiencia similar. Podemos destacar las siguientes:

a) Realizar al menos una sesión semanal de clases que se corresponda con una sesión XP para subsanar los previsibles problemas al principio del proyecto. Al comienzo se tiene un gran inconveniente, los alumnos no conocen la metodología y no se dispone de mucho tiempo como para dedicar sesiones a desarrollar un ejemplo guiado. Las presentaciones introductorias a la metodología mejoraron parcialmente esta situación puesto que mientras no se metieron de lleno en el proyecto los alumnos no se les presentaron dudas. Aunque el primer hito establecido para identificar historias de usuario pretendía darles tiempo para introducirse a XP ese tiempo no fue aprovechado y hasta último momento los equipos no reaccionaron solicitando sesiones XP con el usuario. Posteriormente, durante la primera iteración prácticamente todos los equipos tuvieron dificultades técnicas con los lenguajes elegidos, lo cual unido a su inexperiencia en el trabajo de una iteración hizo que todos los grupos no cumplieran con lo esperado en la primera iteración. Mientras tanto, en clases se desarrollaron actividades independientes del caso de estudio. No fue hasta finales de la segunda iteración en la cual llegamos a dedicar prácticamente todas las sesiones al proyecto. Esta estrategia nos significó enorme esfuerzo en sesiones XP fuera de clases durante las primera mitad del proyecto (alrededor de 4 horas semanales fuera de clases con cada equipo!). Por lo tanto la mejora en este caso será asignar una dedicación uniforme al proyecto contando con una sesión XP desde el principio de la asignatura. Así, cada semana puede dedicarse

una sesión a actividades independientes y otra al proyecto.

b) Utilizar el tiempo de dedicación por parejas de programadores como referencia para las estimaciones y negociación de historias de usuario. Para poder ajustar las estimaciones al contexto y limitaciones de dedicación de la asignatura en un principio las estimaciones se hicieron considerando horas reales de trabajo, las cuales posteriormente se multiplicaban por 4 para conseguir un ajuste más realista de acuerdo con la dedicación parcial de los alumnos. Sin embargo, a partir de la segunda iteración se detectó que era más fácil e intuitivo el considerar una supuesta dedicación semanal del alumno a LSI de 12 horas semanales. Así, estas horas se multiplicaban por el número de parejas y por el número de semanas de la iteración para obtener las horas reales disponibles para iteración. Un problema adicional en la primera iteración fue que los equipos no consideraron aspectos tales como: aprendizaje de características no conocidas del entorno de implementación, esfuerzo asociado a realización y corrección de pruebas, y refactoring. Con lo cual en la primera iteración se consiguió en general mucho menos de lo previsto.

c) El rol de coach (preparador) desempeñado por el profesor resulto muy importante para encauzar el trabajo del equipo y responder a dudas metodológicas, sin interferir mayormente en la autonomía del equipo. Sin embargo, el tener que desempeñar al mismo tiempo el rol de cliente le restó realismo al proyecto. En este sentido sería conveniente que dicho rol de cliente lo desempeñara otra persona. Algunas posibilidades que hemos barajado se comentan a continuación. Se podría conseguir un cliente real, pero tenemos la dificultad que se trata de al menos 7 equipos, lo cual es mucho pedir en términos de dedicación de un usuario real, y además se pasaría más hacia una práctica en empresa haciendo más difícil la formación guiada por el profesor. Otra alternativa más prometedora es intercalar los profesores desempeñando el rol de cliente para algunos grupos y de consultor para otros.

d) Realización de breves presentaciones de resultados al final de cada iteración y otras adicionales de seguimiento. Como es natural y por falta de hábito, en un principio los alumnos se vieron un tanto agobiados por el hecho que tenían que presentar su trabajo al resto de la clase. Sin embargo, al final del cuatrimestre, después de haber realizado unas 6 presentaciones se pudo constatar la mejora y acostumbramiento en este aspecto. Además, estas presentaciones constituyeron el espacio adecuado para realizar comentarios comparativos respecto de RUP y XP.

e) También teníamos algunos temores respecto de qué tan apropiadas resultaban las instalaciones donde trabajarían los alumnos, puesto que XP hace mucho énfasis en el espacio de trabajo. Aunque lo ideal era que hubiésemos contado con un laboratorio acondicionado acorde con las pautas XP, no fue tan problemático trabajar en los laboratorios con la infraestructura tradicional. Sin embargo, el mejorar este aspecto no queda descartado.

f) Aunque en teoría los alumnos tienen la misma formación, es inevitable encontrarse con casos en los cuales no se tiene un conocimiento igual en cuanto a notaciones y herramientas. Esto se debe a paso de Escuela a Facultad, alumnos repetidores o de promociones con otros planes de estudios, cambios en programas de asignaturas previas, etc. En años anteriores se ha probado estableciendo una plataforma de implementación fija (por ejemplo alguna de las siguientes: Oracle Developer y Oracle BD, Visual Basic y SQL Server, o Java y MySQL), la cual debe revisarse en algunas sesiones para conseguir un grado mínimo de conocimiento para todos, pero se corre el riesgo que el alumno se vea en una asignatura de programación o bases de datos, asumiendo que los problemas técnicos son los más importantes y que además deben ser resueltos por el profesor. Otra alternativa que hemos aplicado los dos últimos años es dejar la elección en manos de cada equipo,

existiendo finalmente muchos inconvenientes iniciales por desconocimiento de algunos aspectos del entorno utilizado y también causando problemas en cuanto a las instalaciones en los laboratorios compartidos del departamento. Esta alternativa afectó importantemente a los equipos XP puesto que tuvieron menos tiempo para sobrepasar los problemas de implementación, lo cual impactó en gran medida las estimaciones y objetivos de la primera mitad del proyecto. En este aspecto aún no nos atrevemos decir cual de las dos alternativas es la menos problemática.

6. Conclusiones

En términos generales el resultado ha sido muy satisfactorio. Desde la perspectiva del docente, se ha constatado un incremento en la motivación y participación de los alumnos, los cuales se han sido mucho más protagonistas durante el desarrollo de la asignatura. Esto se refleja en varios aspectos: la casi total asistencia de los alumnos a clases (aunque hay que considerar el hecho que prácticamente en todas las sesiones se realizaba una evaluación o era fundamental para la calificación obtenida), el incremento en asistencia a tutorías, las cuales se transformaron en gran medida en sesiones XP, trasladándose del despacho del profesor a los laboratorios.

Como era previsible, todo este trabajo de mejora docente repercute también y considerablemente en el esfuerzo de dedicación del profesor. Al respecto conviene destacar lo efectivo que resulta incorporar evaluaciones realizadas por los propios alumnos, tanto por su aspecto formador para ellos como para aminorar el trabajo de evaluación por parte del profesor. Por otra parte, el esquema de trabajo y sobre todo el material de apoyo utilizado, en parte ya existían y estaban probados con lo cual no se trataba de un cambio radical, lo que hubiese significado un mayor riesgo.

Finalmente, aunque la experiencia no pretendía realizar una comparativa entre RUP y XP pensamos que vale la pena hacer los siguientes comentarios:

- Con el mismo tiempo de dedicación la funcionalidad implementada fue mayor en los grupos XP. Aunque hay que destacar que en los grupos RUP la primera iteración fue Elaboración, con lo cual sólo realizaron 2 iteraciones de construcción. Los beneficios de haber dedicado una iteración para establecer la estructura no llegaron a ser apreciados puesto que el proyecto no se terminó. Sin embargo, un par de grupos XP pidieron incorporar como tarea explícita el refactoring en la última iteración, lo cual podría considerarse como un síntoma de problemas en la arquitectura.

- Los equipos RUP se mantuvieron poco activos como grupo y respecto de su interacción con el cliente hasta que llegó el momento de iniciar la primera iteración de construcción, parecían bastante cómodos generando los artefactos establecidos (Visión, Modelo de Casos de Uso, Modelo de Análisis y Diseño, Casos de Prueba, etc.). La primera iteración de construcción RUP provocó un cambio significativo en su actitud y en los modelos que habían realizado. Los equipos XP tuvieron mucho más presión en la primera iteración, pero en las posteriores pensamos que se alcanzó un ritmo y rutina adecuada de trabajo.

- La definición más precisa de los artefactos en RUP y las plantillas proporcionadas ofrecen una guía de la información que debe ser recopilada. Sin embargo, en algunos casos como el documento Visión, se tiene demasiado detalle y repetición de información que produce un efecto negativo. En el caso de XP, los alumnos confeccionaron plantillas para historias de usuario y tareas basándose en los pocos ejemplos que aparecen en los libros pero contenían información que no se utilizó. Respecto a los casos de prueba se les indicó un formato para especificarlas. La precisa descripción de los artefactos en XP podría ser de utilidad.

- En RUP la gestión de requisitos se apoyó utilizando RequisitePro, estableciéndose trazabilidad entre los distintos artefactos. Aunque con sus limitaciones, RequisitePro al parecer resultó útil para organizar los requisitos y artefactos en los grupos RUP. Para los grupos XP no era obligatorio utilizar RequisitePro, de hecho sólo un equipo lo hizo y resultó positivo. En los otros equipos todos los artefactos distintos al código se trabajaron sólo con el

procesador de textos. Así, mientras más artefactos se generaban más difícil se hacía la localización de ellos para revisarlos y para verificar la trazabilidad.

- Por la envergadura del proceso definido por RUP y la configuración a nuestro pequeño caso de estudio, los alumnos se quedan con la sensación razonable de no haberlo conocido completo. En cambio en XP la situación es distinta puesto que han abordado la mayoría de aspectos de la metodología.

- Tal como se reconoce en algunos trabajos publicados, RUP y XP son metodologías alternativas. Sobre todo si consideramos un proyecto de una envergadura pequeña o mediana (relativa y vagamente refiriéndonos a una duración corta en meses y no más de 10 integrantes en el equipo). En este perfil de proyecto encaja nuestro caso de estudio. Sin embargo, como también se ha advertido, la relación contractual y de disponibilidad del cliente es crucial para una exitosa aplicación de XP, hecho que verificamos en el desarrollo del proyecto. Con un poco de esfuerzo conseguimos recrear sesiones de trabajo XP estableciendo citas en los laboratorios en las cuales el cliente pasaba desde media hora hasta más de una hora con el equipo.

Referencias

[1] Proyecto EUROPA: Una Enseñanza Orientada al Aprendizaje. Vicerrectorado de Coordinación Académica y Alumnado. Específicamente se trata de los programas AME2: "Nuevos Métodos de Enseñanza-Aprendizaje" y AME3: "Mejora de los Sistemas de Evaluación". Universidad Politécnica de Valencia, 2001. www.upv.es/europa

[2] Terry Quatrani, Grady Booch. Visual Modeling with Rational Rose 2000 and UML. Addison-Wesley Object Technology Series, 1999.

[3] Ivar Jacobson, Grady Booch y James Rumbaugh. The Unified Software Development Process. Addison-Wesley, 1999.

[4] Kent Beck. Una Explicación de la Programación Extrema, Aceptar el Cambio. Addison-Wesley, 2002.

[5] Patricio Letelier. Una Experiencia en Evaluación Continua Multicriterio Aplicada en un Laboratorio de Desarrollo de Software. Actas de las VIII Jornadas de Enseñanza Universitaria de la Informática (JENUI 2002), pp. 295-302, Cáceres, Julio 2002.

Inclusión de patrones de diseño en un plan de estudios de Ingeniería Técnica en Informática de Gestión

Carlos López Nozal, Raúl Marticorena Sánchez, Judith Antolín Sendino, Ignacio Cruzado Nuño
Escuela Politécnica Superior
Departamento de Ingeniería Civil
Área de Lenguajes y Sistemas Informáticos
Universidad de Burgos
e-mail: {clopezno, rmartico, jantolin, icruzado}@ubu.es

Resumen

El presente trabajo define una propuesta de la inclusión del concepto de patrón de diseño en la titulación de Ingeniería Técnica en Informática de Gestión (ITIG), estableciendo los niveles de desarrollo a tratar en las asignaturas *tipo* de esta titulación.

La utilización de patrones de diseño tiene dos aspectos claramente diferenciados: uno relacionado con la aplicación del patrones y otro relacionado con la búsqueda y selección de un patrón que resuelva un determinado problema. El trabajo se basa en esta diferenciación de aspectos para distribuir las responsabilidades entre las diferentes asignaturas del plan de estudios.

El objetivo final del mismo, es mostrar la experiencia previa de dicho trabajo en la Universidad de Burgos y concluir con una propuesta para la incorporación inmediata y adecuada de dichos temas.

1. Presentación

La importancia de los patrones de diseño hoy en día es evidente, sin embargo, la actual definición de planes de estudio, en la mayoría del ámbito nacional, no señala su inclusión, provocando un vacío evidente a nuestro juicio.

Por otro lado, la última actualización del *Computing Curricula 2001* [1], incorpora en muchas de sus diferentes propuestas de cursos de programación la unidad *SE1.Software Design*[1], dentro del núcleo de dichas asignaturas. Esta unidad incorpora el concepto de patrones de diseño para elevar el nivel de abstracción en el planteamiento de soluciones.

Siguiendo esta recomendación se pueden asignar responsabilidades concretas a las asignaturas implicadas en la enseñanza de patrones, con el fin de conseguir el objetivo de aprendizaje de *"selección y aplicación de los patrones de diseño adecuado en la construcción de una aplicación software"*[1].

El objetivo es cubrir las deficiencias actuales estableciendo los cursos y nivel de profundidad en el desarrollo docente de los patrones, de tal forma que su inclusión en un plan de estudios no sea un cambio traumático, sino natural.

La estructura del documento, consiste en primer lugar, en el planteamiento del contexto particular en el que hemos llevado a cabo este proceso, en nuestro caso bajo el plan de estudios de ITIG en la Universidad de Burgos, para posteriormente detallar la solución implantada en cada una de las asignaturas correspondientes, indicando la responsabilidad específica y cómo se realiza en cada asignatura. Finalizamos con las conclusiones y experiencias obtenidas de la realización de dicho proceso.

2. Contexto propio

Para comprender la propuesta, es necesario conocer el actual contexto en el que se está llevando a cabo.

En el plan de estudios de ITIG en la Universidad de Burgos, vigente desde el curso 1995/1996, se identifican tres asignaturas que pueden estar implicadas en la docencia de patrones: *Metodología de la Programación*,

[1] SE significa el área del Software Engineering y el número 1 la unidad dentro del área.

Análisis e Ingeniería del Software y *Programación Avanzada*.

Dentro del segundo curso de la titulación se imparten las asignaturas troncales de *Metodología de la Programación* y *Análisis e Ingeniería del Software*. En el tercer curso se imparte la asignatura de *Programación Avanzada* de carácter optativo, con claras dependencias de los contenidos vistos en las asignaturas anteriormente mencionadas.

La propuesta se basa en introducir los aspectos relacionados con la aplicación de patrones de diseño en las asignaturas de segundo curso e incidir en búsqueda y selección de patrones en la asignatura de tercer curso. La causa de esta distribución de aspectos viene dada por la necesidad de un conocimiento exhaustivo de un catálogo de patrones para poder realizar el proceso de búsqueda y selección de los mismos. Este requisito no puede ser considerado en las asignaturas de segundo curso por tener que satisfacer otros múltiples objetivos.

La razón de introducir el concepto de patrón en dos asignaturas del segundo curso es debida a la distintas técnicas utilizadas para presentar los elementos descriptivos de un patrón. En la Tabla 1 se muestran dos plantillas descriptivas de un patrón especificadas en [2], [3]. En ella se han sombreado los elementos relacionados con el conocimiento concreto de un lenguaje de programación orientado a objetos. Estos elementos sólo pueden ser tratados en asignaturas orientadas a programación, *Metodología de la Programación* y *Programación Avanzada*.

Si establecemos una equivalencia con lo que se entiende por programación de nivel[2] II y III, se corresponderían con *Metodología de la Programación* y *Programación Avanzada* respectivamente. Por otro lado nos referiremos a la asignatura de *Análisis e Ingeniería del Software* como ingeniería del software. Señalar que uno de los problemas que surge de este planteamiento es el carácter optativo de la asignatura de programación de nivel III, que dificulta el llevar a cabo el proceso completo para todos los alumnos, como hubiese sido la intención de los autores[3].

Plantilla de GoF[4]	**Plantilla de Grand**
Nombre y clasificación	Nombre
Propósito	Sinopsis
También conocido	
Motivación	Contexto
Aplicabilidad	Fuerzas
Estructura	Solución
Participantes	
Colaboraciones	
Consecuencias	Consecuencias
Implementación	**Implementación**
Código de ejemplo	
Usos Conocidos	**Java API**
Patrones Relacionados	Patrones Relacionados

Tabla 1. Plantilla de patrón de diseño.

Para finalizar el flujo de dependencias, señalar también la existencia de un trabajo final de carrera en el que actualmente en el 90% de los casos, se están utilizando lenguajes orientados a objetos (Java y C++) para la parte de programación. Una vez que nuestra propuesta esté implantada, es en esta asignatura donde se puede observar finalmente el éxito o fracaso de la misma. Desgraciadamente dado lo reciente de la experiencia y el pequeño nº de proyectos en los que se ha empezado a aplicar este planteamiento (aunque el objetivo final es abarcar, en mayor o menor medida, la práctica totalidad de proyectos propuestos por el Área de Lenguajes y Sistemas Informáticos) las observaciones obtenidas no son relevantes todavía.

3. Propuesta realizada

La propuesta que aquí se realiza consiste en presentar al alumno el concepto de patrón de diseño y ejemplos de uso en la asignatura de programación de nivel II y en la asignatura de ingeniería del software, mientras que en la asignatura de programación de nivel III se hace un estudio más completo del catálogo de los mismos,

[2] Se entiende por programación de nivel II y III a las asignaturas de programación vistas en el segundo y tercer año de la titulación técnica.

[3] Este posible vacío esperamos que sea revisado en la próxima elaboración del plan de estudios que se está llevando a cabo actualmente.

[4] Gang of Four

razonando sobre su selección y utilización para la resolución de problemas generales.

La mayor parte de la bibliografía referente a patrones [2], [3], [5], [7], [8], utiliza alguna plantilla descriptiva, como la mostrada en Tabla 1 o equivalente, cuyo enfoque se fundamenta principalmente en satisfacer el proceso de aplicación del mismo.

Sin embargo, y por regla general, la bibliografía se limita a utilizar el patrón en un contexto particular. Un elemento clave de la plantilla que interviene en el proceso de búsqueda es el de patrones relacionados, que se describe de forma textual y necesita del conocimiento de un catálogo de patrones. El método didáctico seguido para poder mostrar este proceso de búsqueda y selección es a través de un caso de estudio tal como se propone en el capítulo 2 de [2]. En particular, lo que tratamos de plantear y resolver es precisamente ese vacío existente, que por regla general nos han transmitido los alumnos en otros trabajos [6], existiendo un cierto desánimo a su utilización posterior.

A continuación vamos a detallar las distintas estrategias adoptadas en cada una de las asignaturas para llevar a cabo este proceso.

4. Patrones de diseño en ingeniería del software

La responsabilidad asignada a esta asignatura en lo referente a patrones de diseño se basa fundamentalmente en la aplicación de los mismos.

En las clases de teoría se expone el proceso de desarrollo propuesto por Larman en [5] compuesto por varias iteraciones de las fases de análisis, diseño y construcción. Es en la fase de diseño donde se introducen los patrones, en concreto se trabaja con los patrones propuestos por el autor y recogidos en [4] como catálogo de patrones GRASP (General Responsibility Assignment Software Patterns). Además se exponen algunos patrones aislados del catálogo de propuesto en [2] (Singleton, Fachada, Observador, Comando, etc.).

Para la exposición teórica del patrón se utiliza una descripción de los elementos esenciales: intención, contexto, solución y consecuencias. Una vez expuesto cada patrón, se aplica a un caso práctico guiado, que sirve como referencia para poder obtener los artefactos propuestos en el proceso.

En la parte práctica de la asignatura se propone un colección de requisitos de un sistema a partir de los cuales se deben obtener los diferentes artefactos presentados en teoría. Para poder obtener los artefactos de diseño y las posibles alternativas se deben aplicar los patrones expuestos. Aunque existe un pequeño proceso de búsqueda sobre el catálogo GRASP, no se puede considerar suficiente por el escaso número de patrones que contiene este catálogo. Además los patrones utilizados son encontrados casi sistemáticamente por el hecho de proponer contextos muy similares al caso práctico mostrado en teoría.

5. Patrones de diseño en programación de nivel II

La asignatura de programación de nivel II, es la primera asignatura en el contexto particular de nuestro plan de estudios en que se introduce al alumno en el paradigma de la programación orientada a objeto (POO).

Aunque dicha inclusión puede llevar a discusiones, en nuestro caso particular y dadas las dependencias existentes en el plan de estudios actual, este hecho es necesario para completar los objetivos de la formación del alumnado.

Además esto se ve confirmado por otras fuentes como la obligatoriedad de un módulo de POO en la formación obligatoria por parte de *Computing Curricula Computer Science* [1].

El lenguaje elegido para la realización de prácticas es Java por cumplir en su mayoría, las características propias un lenguaje orientado a objeto puro[5], con una gran disponibilidad de bibliotecas, entornos y documentación. Además la facilidad de comprensión del lenguaje permite un rápido desarrollo de las prácticas.

El planteamiento en la parte de teoría es presentar al alumno el uso particular de algún patrón de diseño en contextos muy particulares. Siempre explicando el nombre, problema, solución y consecuencias de la aplicación del patrón. En el desarrollo teórico de la asignatura se aprovechan las múltiples circunstancias en las que

[5] Siempre criticable esta afirmación por la inclusión de tipos primitivos en el lenguaje.

el empleo de patrones de diseño resuelve nuestros problemas. Como ejemplo tenemos la utilización de patrones en la categoría de patrones esenciales como Interface y Delegación [3] para la simulación de herencia múltiple con lenguajes que no la soportan. El planteamiento docente es primero mostrar y usar, para después razonar más detenidamente sobre el sentido de reutilización de diseño que nos están aportando.

Como parte del temario, finalizamos en los últimos temas definiendo formalmente la idea de patrón y recapitulamos sobre el empleo "inconsciente" (pero siempre guiado por el profesor) de los patrones para la resolución de un problema general en un contexto particular. Una vez presentado el concepto se puede finalizar con el estudio más detallado de algún patrón concreto, con fuerte uso de las características propias de la orientación a objeto, de esta forma, el alumno confirma y refuerza los conceptos teóricos, viendo una aplicación a posteriori más allá del simple ámbito de la asignatura.

5.1. Práctica planteada

El enfoque en prácticas para la asignatura de programación de nivel II consiste en la resolución de un problema general a través de la utilización de uno o varios patrones de diseño. La finalidad es, de manera transparente, comenzar a trabajar con patrones con un lenguaje orientado a objeto.

En particular, la práctica realizada en el presente curso, plantea la búsqueda de un elemento concreto dentro de estructuras compuestas, que a su vez pueden contener a nuevas estructuras compuestas u objetos finales. Ésto forma árboles en los que la raíz y nodos intermedios son composiciones y los elementos finales son hojas.

Dichas estructuras se encuentran en un gran número de ejemplos de la vida real, como las estructuras de directorios y ficheros, las interfaces gráficas con paneles y componentes gráficos, los paquetes y las clases en Java, etc.

El patrón que se adapta de manera natural a la resolución de dicho problema es el patrón Compuesto[6] [2], [3]. La jerarquía de herencia se establece entre lo que denominamos clases Elemento y dos descendientes directos, Composición y Hoja.

En la Figura 1 podemos ver la solución planteada, señalando que el diseño se proporciona a los alumnos.

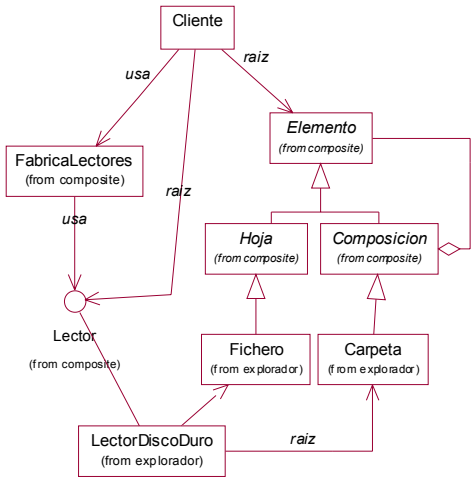


Figura 1. Diagrama de clases del marco de acceso a composiciones de elementos.

Una vez determinada la solución de alto nivel, se solicita que utilizando los mecanismos propios de herencia en Java, se reutilice parte de la implementación realizada (concentrada en la implementación del patrón Compuesto) para resolver problemas concretos en un ambiente más real como la búsqueda de ficheros en directorios.

Para facilitar la labor al alumno, la lectura y carga de directorios y ficheros se simula con una clase *LectorDiscoDuro*, con lo que la realización de la práctica se centra en la resolución del problema inicial, evitando el desviarnos con el uso avanzado de alguna API concreta.

El objetivo final de la práctica es que el alumno, además de afianzar los conceptos propios de la POO (herencia, polimorfismo, ligadura dinámica, etc.), enfoque el resultado final a la reutilización mediante la construcción de un marco[7] basándose en uno o varios patrones.

[6] Traducción del inglés Composite

[7] Traducción del término inglés framework

6. Patrones de diseño en programación de nivel III

La experiencia obtenida con la aplicación del plan propuesto en [6], donde se elegía únicamente un patrón, vislumbró que los alumnos eran capaces de aplicar, sin mucha dificultad, el patrón propuesto de forma aislada, pero se encontraban con graves problemas a la hora de crear un sistema utilizando varios patrones relacionados.

Para poder superar esta deficiencia es necesario el conocimiento de un catálogo de patrones que es expuesto en las clases de teoría; en concreto se trabaja con el catálogo propuesto en [2]. El enfoque utilizado en esta exposición se basa fundamentalmente en la aplicación de los patrones.

En la parte práctica es donde realmente se incide en el proceso de búsqueda y selección. Para ello se propone un enunciado donde se especifican unos requisitos de diseño que los alumnos deberán resolver. La metodología didáctica empleada se basa en talleres donde los alumnos discuten y proponen sus soluciones hasta llegar a un diseño por consenso que será utilizado para la implementación del sistema propuesto. En estos talleres el profesor actúa como moderador guiando este proceso de búsqueda y selección de los patrones.

A continuación se describe la materialización práctica de la estrategia mencionada proporcionando el enunciado de la práctica y las soluciones de diseño empleadas.

6.1. Práctica planteada

La práctica consiste en realizar una aplicación Java que permita visualizar un conjunto de imágenes cuyos datos (autor, descripción y nombre de fichero) se encuentran almacenados en un sistema de almacenamiento persistente. La Figura 2 muestra un posible diseño de la interface de usuario para interactuar con los datos. Para la resolución de la práctica se tienen en cuenta una serie de restricciones de diseño:

1. En previsión del crecimiento de la aplicación y para mejorar la reutilización y mantenimiento de la misma, se quiere estructurar su desarrollo con una arquitectura de tres capas (Presentación – Dominio – Servicios) respetando las dependencias mostradas en la Figura 3.
2. En previsión de futuros cambios del sistema de almacenamiento persistente se quiere mejorar la reutilización manteniendo una independencia del mismo.
3. Intentar minimizar el acoplamiento existente entre las clases del paquete presentación para prever una futura reutilización de los mismos.
4. Encapsular las diferentes peticiones que se pueden efectuar desde la interfaz gráfica como objetos.

Figura 2. Interfaz gráfica de la aplicación.

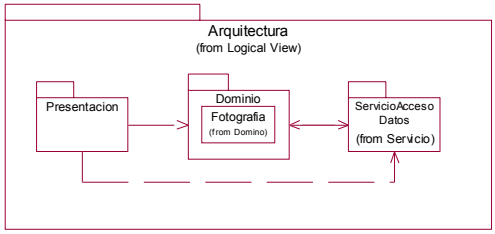


Figura 3. Diseño arquitectónico.

6.2. Soluciones finales de diseño

En este apartado se recogen las soluciones de diseño obtenidas a través de los distintos talleres. El enfoque seguido es ir tratando de forma independiente cada solución seleccionando y buscando patrones que las resuelvan.

- Solución del problema de diseño 1.

El patrón fachada facilita la división de un sistema en capas proporcionando una única clase que permite acceder al resto clases contenidas

dentro del paquete correspondiente. De esta forma se minimiza el acoplamiento de los clientes ya que únicamente tienen que conocer una clase, la fachada, para acceder a los servicios que le pueden proporcionar las clases contenidas en el paquete. Se podría añadir una fachada para el paquete de Dominio y otra para el paquete de ServicioAccesoDatos. Si se analizan las clases del dominio de la aplicación únicamente se identifica la clase Fotografía cuyo tipo se especifica en la Figura 4. Añadir una indirección a través de la fachada para poder acceder a los servicios proporcionados por la clase Fotografía no reporta ningún beneficio, por tanto no se añadirá una Fachada para poder acceder a las clases de dominio.

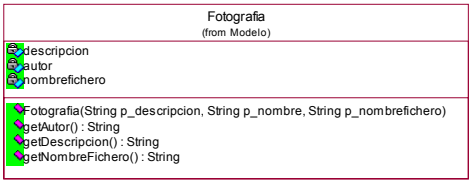


Figura 4. Clase Fotografía.

Para poder acceder a los datos de un sistema de almacenamiento persistente (fichero o bases de datos), se tiene que conocer un conjunto de clases del API (java.io o java.sql), cómo se relacionan y cómo se usan. En este caso sí parece interesante incluir una clase Fachada que permita acceder al sistema de almacenamiento persistente. De esta forma se libera al cliente del conocimiento de las clases del API concreta. En el diagrama de clases de la Figura 5 se muestra un posible diseño de una fachada para acceder a los datos del sistema de almacenamiento persistente, en este caso una base de datos. Hay que observar que la creación de datos del modelo (Fotografía) puede recaer sobre la Fachada o sobre el cliente. Con esta segunda opción se elimina la dependencia de la capa de servicios sobre la capa de modelo a costa de añadir más complejidad en las clases cliente.

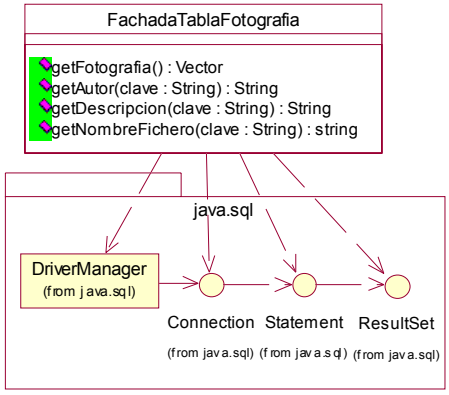


Figura 5. Fachada del servicio de acceso a datos.

Una vez que los datos del modelo están cargados, los objetos de la capa de presentación utilizaran los objetos del modelo para acceder a los datos.

- Solución del problema de diseño 2

Si se quiere soportar diferentes sistemas de almacenamiento persistente se debe establecer una relación de herencia respecto a las fachadas de acceso a datos, donde la superclase define las operaciones abstractas que permiten acceder a los datos y las subclases concretas definen estas operaciones para el sistema de almacenamiento concreto. De esta forma los clientes pueden utilizar objetos de las subclases concretas mediante referencias de la superclase (polimorfismo). En este enfoque la única dependencia que tiene el cliente sobre el sistema de almacenamiento concreto está en el momento de instanciar u obtener la fachada que da acceso al mismo. Para poder eliminar esta dependencia se puede utilizar el patrón de diseño método de fabricación. La Figura 6 muestra la solución estructural de este problema.

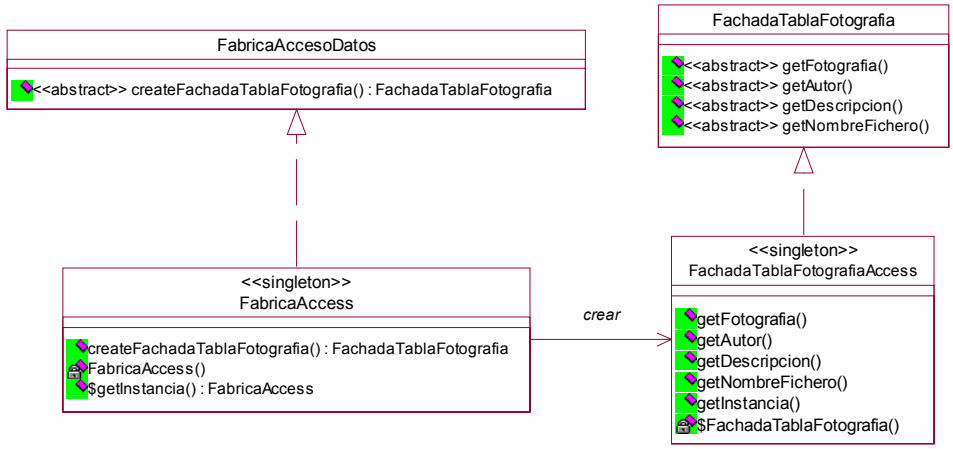


Figura 6. Independencia del sistema de almacenamiento

- Solución del problema de diseño 3.

Cuando se diseñan interfaces gráficas suelen aparecer muchas dependencias entre las clases que contienen componentes gráficos. Estas dependencias hacen que dichas clases tengan un alto acoplamiento dificultando su posterior reutilización. El patrón mediador minimiza el número de dependencias incorporando en el diseño una nueva clase Mediador que se encarga de tratarlas.

En el diseño gráfico presentado en la Figura 2 se identifican los siguientes componentes gráficos:

Figura 7. Panel barra de desplazamiento (PBarraDesplazamiento).

Figura 8. Panel de datos textuales (PDatos).

Figura 9. Canvas para visualizar las imágenes (VisorImagen).

En el diagrama de clases de la Figura 10 se puede observar la aplicación del patrón mediador en este contexto.

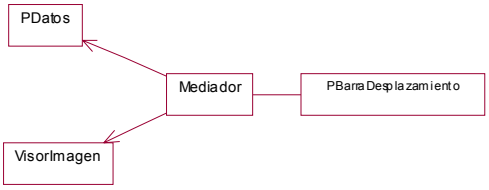


Figura 10. Aplicación del patrón mediador para minimizar dependencias.

- Solución del problema de diseño 4

Las posibles peticiones del usuario al sistema están relacionadas con la forma de obtener los datos. El patrón comando permite encapsular cada

petición como una clase. La Figura 11 muestra un diseño estructural para encapsular estas peticiones.

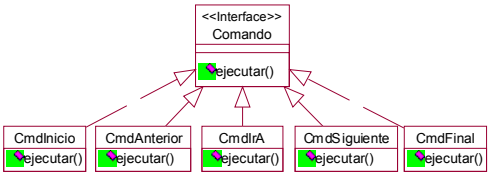


Figura 11. Encapsulamiento de peticiones de la interfaz.

7. Conclusiones

A la vista de los resultados obtenidos planteamos la integración de nuestra propuesta en la titulación de ITIG. Pese a las dificultades de la implantación de dichos temas y a la realización de unas prácticas orientadas, desde un principio, a llevar al alumno a un automatismo en el uso de patrones, los positivos resultados obtenidos nos animan a seguir desarrollando estas ideas, incluso con una labor de coordinación más fuerte entre las asignaturas.

El planteamiento final que realizamos es el mostrado en Tabla 2.

Asignatura	Teoría	Práctica
Ingeniería del software	Presentación del concepto y proceso de búsqueda de patrones en análisis y diseño (6 horas).	Obtención de artefactos de diseño a partir de un conjunto de casos propuestos (6 horas).
Programación de nivel II	Presentación del concepto de patrón y uso de la POO para su implementación (3 horas).	Programación de uno o varios patrones para la resolución de las prácticas planteadas (8 horas).
Programación de nivel III	Presentación de varios catálogos (Gamma & Concurrencia) y ejemplos de uso en contextos determinados (30 horas).	Planteamiento de problemas generales y talleres de búsqueda y selección de patrones para la implementación de la solución del problema (12 horas).

Tabla 2. Propuesta de la planificación de inclusión de patrones de diseño.

La ampliación de esta propuesta inicial puede extenderse introduciendo los patrones en otras muchas asignaturas del plan de estudios. Los patrones homogeneizan la forma de presentar los problemas y sus soluciones. Existen muchos catálogos que pueden ser aplicados en los contenidos de otras asignaturas: catálogos de concurrencia [3] en asignaturas dedicadas a la programación concurrente, catálogos de interfaces [4] en asignaturas dedicadas al diseño de interfaces gráficas de usuario y catálogos de J2EE [8] para asignaturas dedicadas a componentes distribuidos, en este caso especial con Java.

Esta propuesta, teniendo en cuenta las enormes posibilidades abiertas, se puede extender a las asignaturas de un segundo ciclo en Ingeniería Informática, en el caso de que los resultados obtenidos sean positivos.

Como conclusión final creemos cubrir de esta forma el vacío existente en cuanto al tratamiento correcto de los patrones de diseño dentro de una titulación técnica en informática de gestión.

Referencias

[1] ACM & IEEE. *Computing Curricula 2001. Computer Science* (2001). The Joint Task Force on Computing Curricula IEEE Computer Society. Association for Computing Machinery. Ed IEEE-CS y ACM.

[2] Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1995). *Design Patterns. Elements of Reusable Object-Oriented Software*. Addison Wesley.

[3] Grand, M. (1998). *Patterns in Java. Volume 1*. Wiley Computer Publishing.

[4] Grand, M. (1999). *Patterns in Java. Volumen 2*. Wiley Computer Publishing.

[5] Larman, C. (2002). *UML y Patrones. Una introducción al análisis y diseño orientado a objetos y al proceso unificado*. Ed. Prentice Hall.

[6] Marticorena, R., López, C., García, C. I, Pardo C. (2002). *Aprendizaje práctico de patrones de diseño en asignaturas de programación de nivel III*. Actas JENUI 2002, pág 471.

[7] Metsker, S. J. (2002). *Design Patterns Java WorkBook*. Ed Addison Wesley.

[8] Stelting, S. & Maassen, O. (2002). *Applied Java Patterns*. Ed The Sun Microsystems Press. Java Series.

Aplicación práctica de un proceso basado en UML

Jesús García Molina, Marcos Menárguez Tortosa, Joaquín Nicolás Ros
Dept. de Informática y Sistemas
Universidad de Murcia
30071 Murcia
e-mail: (jmolina | marcos | jnr)@um.es

Resumen

En las anteriores JENUI de 2002 presentamos una propuesta para organizar la enseñanza de la orientación a objetos a través de dos asignaturas obligatorias, una de *Introducción a la Programación Orientada a Objetos* y otra de *Análisis y Diseño Orientado a Objetos*. Ahora nos centraremos en el principal objetivo de la segunda asignatura: los alumnos deben conocer y aplicar un proceso software basado en UML. En este trabajo discutimos cómo hemos planificado las prácticas de dicha asignatura para conseguir el objetivo formativo propuesto, presentando también una valoración de los resultados.

1. Introducción

En la actualidad, la mayoría de universidades incluyen asignaturas relacionadas con la orientación a objetos, OO. En el caso de las universidades españolas, si se observan los programas de las asignaturas relacionadas con la programación y la ingeniería del software OO, resulta evidente que no existe un consenso en cuanto a la forma de plantear la enseñanza de la construcción de software OO. En las JENUI de 2002 presentamos una propuesta [5] sobre cómo organizar la enseñanza de esa materia y ahora nos centraremos en un aspecto esencial de ella: los alumnos deben conocer y aplicar un proceso software basado en UML.

El objetivo del presente trabajo es describir cómo hemos organizado, durante los últimos tres cursos, la enseñanza de un proceso UML, en concreto el proceso presentado por Craig Larman en su libro *"UML y Patrones"* [7], que se ajusta al Proceso Unificado (UP) [6].

Las aportaciones que pretendemos realizar con el trabajo son: i) mostrar una posible organización docente para enseñar un proceso UML, ii) valorar el proceso de Larman, iii) ofrecer guías que ayuden a aplicar nuestro enfoque o uno similar.

Hemos comprobado que todos los planes de estudio de la titulación *Ingeniero en Informática*, incluyen al menos una asignatura en la que se enseña, de uno u otro modo, "métodos OO" y que casi siempre aparece el libro de Larman en la bibliografía, por lo que creemos que este trabajo puede ser de interés para un buen número de profesores que enseñan una materia que todavía no está bien definida en el plano docente.

El trabajo se organiza del siguiente modo. En la siguiente sección se describe la asignatura en la que se enmarca la enseñanza del proceso UML. En la sección 3 se describe cómo se desarrolla el trabajo práctico de la asignatura y en la sección 4 se ofrece una serie de recomendaciones para cada una de las etapas del proceso. Finalmente se incluyen las conclusiones y las referencias.

2. Una asignatura sobre Análisis y Diseño OO

La propuesta curricular *Computing Curricula 2001* de ACM/IEEE [1] ha reconocido la importancia adquirida por la programación OO en la pasada década y ha añadido este área al núcleo de conocimientos básicos de la disciplina, estableciendo que cualquier estudiante de informática debería tener una formación en Programación OO, Análisis y Diseño OO, y Patrones de Diseño.

En [5] planteamos una organización de la enseñanza de la OO, que se ajustaba a [1], a través de dos asignaturas cuatrimestrales de seis créditos:

Introducción a la Programación OO y *Análisis y Diseño OO*. En la primera se introducen los conceptos básicos que caracterizan al paradigma de programación OO y la segunda se dedica al estudio y aplicación práctica de un proceso software UML y de los patrones de diseño GoF [3]. En [5] se presentó una visión general de la asignatura de *Análisis y Diseño OO* (nos referiremos a ella como ADOO), por lo que en esta sección profundizamos en aspectos relacionados con sus contenidos teóricos, que establecen la base para analizar la organización de las prácticas.

En nuestro caso, ADOO es una asignatura troncal de cuarto curso de seis créditos (3 teóricos y 3 prácticos). Sus contenidos teóricos incluyen los siguientes temas: i) El Lenguaje UML (7 horas), ii) Un proceso basado en UML (8 horas), iii) Patrones de Diseño (12 horas), y iv) Persistencia OO (3 horas).

Actualmente es indiscutible que UML se ha convertido en un lenguaje de modelado estándar y que cualquier proceso software OO debe utilizarlo como notación. Se han definido muchos procesos para UML, siendo RUP (*Rational Unified Process*) [9] el más conocido y extendido. En realidad, RUP es una materialización particular del *Proceso Unificado* (*Unified Process*, UP) [6]. UP establece un proceso más genérico que ha sido ampliamente aceptado, de modo que cualquier proceso basado en UML debería encajar o ser compatible con UP.

Sin embargo, RUP es un proceso muy grande y complejo que no puede ser abordado en el contexto de una asignatura. Es necesario un proceso UML más simple, a ser posible también compatible con UP. En este sentido creemos que el proceso descrito por C. Larman en su excelente libro *"UML y Patrones"* [7] es muy apropiado, ya que encaja con UP y es muy sencillo, permitiendo abordar de una manera sistemática y realista la construcción de software OO. Además posee las propiedades que deben caracterizar a todo proceso UML: dirigido por casos de uso, iterativo, incremental y centrado en la arquitectura.

Creemos que el libro de Larman es el mejor texto publicado para un curso de análisis y diseño OO, ya que combina un enfoque práctico (utiliza un único caso de estudio que se sigue a lo largo de todo el libro), con un tratamiento muy riguroso de las cuestiones que van surgiendo durante todo el proceso de desarrollo en sus diferentes etapas, y además está escrito de un modo muy didáctico.

En la asignatura se sigue en gran medida este texto pero con algunas variaciones: i) comenzamos con una descripción completa de UML, en vez de introducir los modelos cuando se necesitan , ii) se introduce una etapa de modelado de negocio previa al modelado de requisitos, iii) se desdobla la etapa de *Modelado de Diseño* en dos: *Análisis* y *Diseño*, y iv) se estudian en profundidad todos los patrones GoF.

Nosotros complementamos el proceso de Larman con una etapa de *Modelado de Negocio* que precede al *Modelado de Casos de Uso y Modelado del Dominio*. El *Modelado del Negocio* es una disciplina incluida en UP que tiene como objetivo identificar y describir los procesos de negocio (casos de uso del negocio). Mostramos cómo este modelado es útil para identificar los casos de uso y vocabulario del sistema, aplicando las técnicas descritas en [4]: los procesos del negocio se describen a través de escenarios (diagramas de secuencia) y diagramas de proceso (diagramas de actividad) en los que se muestra cómo interactuan los actores (agentes que participan) y qué tareas o actividades ejecuta cada actor; finalmente, se extraen los *casos de uso del sistema*, *conceptos del dominio* y *reglas de negocio* a partir de los diagramas de proceso, aplicando unas reglas muy simples.

Por otra parte, descomponemos el *Modelado de Diseño* de Larman en dos etapas: una más cercana a los requisitos, que llamamos *Modelo de Análisis*, en la que el objetivo es obtener un diseño preliminar del sistema, determinando las colaboraciones, entre objetos de las clases del dominio, que implementan los casos de uso; y otra más cercana a la implementación, que llamamos *Modelo del Diseño*, en la que se refina el modelo del análisis introduciendo los requisitos no funcionales, y considerando cuestiones como la definición de la arquitectura del sistema, aplicación de patrones de diseño, la organización en paquetes, la persistencia o la distribución.

El estudio de los patrones GoF ocupa una buena parte de las clases teóricas del curso. Nos centramos en los aspectos de *motivación*, casos de *aplicabilidad*, *estructura* de clases, *colaboración* entre objetos participantes y *consecuencias* de aplicación, pero no se aborda la *implementación* de los patrones en un determinado lenguaje. El

objetivo es que los alumnos reconozcan situaciones en las que se puede y debe aplicar un determinado patrón, conozcan bien la estructura y comportamiento de cada patrón para su correcta aplicación, y las ventajas y problemas de su uso. Antes de comenzar con la discusión de los patrones GoF se presentan los patrones GRASP de Larman como patrones generales para el reparto de responsabilidades entre clases que siempre hay que tener en mente.

Finalmente, señalar que en el segundo ciclo de nuestro plan de estudios de Ingeniero en Informática se imparten otras dos asignaturas troncales de ingeniería del software: una centrada en ingeniería de requisitos, sobre todo en estándares y guías, y otra de gestión y planificación de proyecto informáticos. Conviene destacar que la primera de ellas se imparte antes de ADOO, pero no se ocupa del modelado de requisitos con casos de uso.

3. Organización del trabajo práctico

El objetivo principal del trabajo práctico que deben desarrollar los alumnos es la aplicación del proceso de Larman descrito en las clases teóricas, prestando también atención a la aplicación de los patrones GoF.

Las clases prácticas comienzan la tercera semana del inicio del cuatrimestre y se dedican a sesiones de ejercicios de modelado con UML de una hora y media de duración: tres de modelado de caso de uso, una de modelado conceptual, tres de modelado de colaboraciones, y una sesión para comentar la práctica del curso anterior y la documentación a entregar. Al acabar de explicar el proceso UML en clases de teoría, se dedican tres sesiones de laboratorio de hora y media cada una a explicar la herramienta de modelado que utilizarán los alumnos, en nuestro caso *Rational Rose 2001*. A mitad del cuatrimestre, entregamos a los alumnos la especificación del caso práctico para el que deben aplicar el proceso (se forman grupos de dos alumnos). Como se puede observar, se combinan prácticas presenciales junto con trabajo práctico que los alumnos realizan de acuerdo a su propia planificación. La Figura 1 muestra la ordenación temporal de las clases teóricas y prácticas (dos horas semanales de teoría y hora y media semanal para clases prácticas).

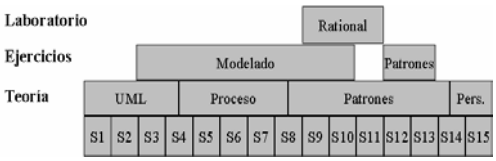


Figura 1. Ordenación temporal de los contenidos

En las sesiones de ejercicios de modelado se discute el ejemplo del terminal de punto de venta del libro de Larman y partes interesantes de trabajos prácticos de cursos anteriores, junto con otros ejercicios cuyo enunciado se les entrega con antelación. Estas sesiones finalizan ilustrando a través de un caso de uso cómo aplicar el proceso completo.

Dentro del curso, la herramienta de modelado no juega un papel central ya que nuestra principal preocupación es el aprendizaje de las técnicas relativas al modelado OO. Nosotros utilizamos *Rational Rose 2001*, pero igualmente se podría utilizar otras como *Together*, *Poseidon* o *ArgoUML*. En las tres sesiones de laboratorio con la herramienta se muestra a los alumnos cómo construir los diferentes diagramas de UML a través de ejemplos. También se presta atención a cómo organizar los modelos en paquetes y a la generación de código.

En cuanto a la aplicación del proceso por parte de los alumnos, el trabajo se organiza en las tres etapas que hemos comentado anteriormente: *Modelado de Requisitos*, *Modelado de Análisis* y *Modelado del Diseño*. La figura 2 ilustra las relaciones entre las tres etapas. Cómo se puede observar se ha excluido la etapa de *Modelado del Negocio* comentada en la anterior sección. Esta etapa fue obligatoria en el primer año, pero notamos que consumía mucho tiempo y que no era posible abordarla teniendo en cuenta la carga de créditos prácticos de la asignatura, por lo que decidimos que fuese opcional. Un modelado de negocio de interés, por muy simple que sea, requiere un esfuerzo importante, que es preferible dedicarlo a las otras etapas.

Si el caso práctico se presta a un *Modelado del Negocio* (involucra procesos de negocios con agentes que cooperan para cumplimentarlos), el alumno reconoce el interés de realizar los diagramas de procesos ya que les facilita la

identificación de los casos de uso del sistema. El caso práctico del pasado curso fue la gestión de un videoclub con máquinas automáticas de alquiler y ningún grupo optó por realizar esos diagramas de procesos, a diferencia de hace dos cursos en el que la mayoría de grupos decidieron realizar dichos diagramas, ya que el caso práctico era la gestión de pedidos en una empresa de fabricación bajo demanda.

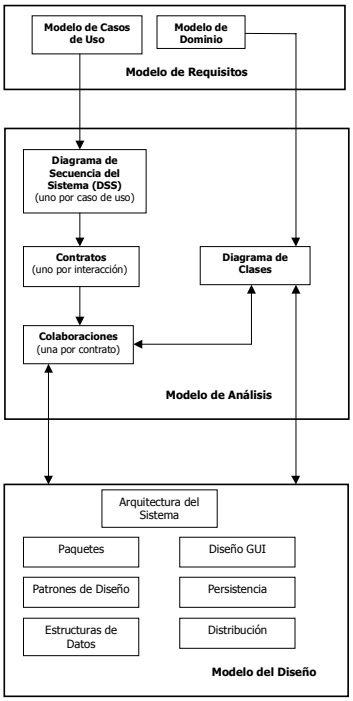


Figura 2. Etapas del proceso

En cuanto al tamaño del caso práctico, planteamos problemas en los que el alumno puede identificar de quince a veinte casos de uso, pero hay cinco o seis, que son los que corresponden a los *objetivos básicos,* en los que centra su atención; el resto son casos de uso *accesorios* (en [8] se puede encontrar esta diferenciación entre casos de uso básicos y accesorios). El problema de la gestión de un videoclub ilustra bien este tamaño de práctica.

Cada grupo debe mantener una entrevista con el profesor al final de la primera y segunda etapas, en la que se revisan los modelos realizados y se establece una lista de cambios obligatorios, cuestiones a repasar y posibles mejoras. El grupo entrega los documentos del modelado, el profesor los revisa y entonces acuerdan una cita. El alumno no puede pasar a la siguiente etapa si no es con la aprobación del profesor.

Los grupos se enfrentan primero al *Modelado de Requisitos*, debiendo realizar el *Modelo de Casos de Uso* y el *Modelo del Dominio*. Los casos de uso son especificados con una plantilla que incluye los campos: *Nombre, Objetivo, Actor Principal, Precondiciones, Flujo Principal* y *Flujos Alternativos*, siguiendo el estilo de Larman. En realidad, para esta etapa los grupos mantienen dos entrevistas con el profesor. Una para comprobar que la identificación de casos de uso es correcta y otra al final, una vez se han completado los dos modelos. El objetivo de la primera entrevista es evitar el esfuerzo inútil de escribir casos de uso que no tienen sentido.

No se tienen en cuenta los requisitos no funcionales, salvo algunos relacionados con la extensibilidad y reutilización del sistema final, que posibilitan la aplicación de patrones.

Una vez realizado el *Modelado de Requisitos*, se realiza el *Modelado del Análisis*. Durante esta etapa se especifican los *Diagramas de Secuencia del Sistema* (DSS), los *Contratos*, las *Colaboraciones* y el *Modelado de Clases*, tal y como propone Larman. Para cada caso de uso se define un DSS (diagrama de secuencia UML) que muestra la secuencia de eventos generados por el actor sobre el sistema a lo largo del caso de uso (en realidad sería un DSS para el flujo principal y otro para cada flujo alternativo).

Los contratos describen el comportamiento del sistema asociado a cada evento de un DSS (objetos y asociaciones que se crean o destruyen, cambios de estado en los objetos). Antes de pasar a realizar las colaboraciones, una para cada contrato, algunos grupos prefieren discutir con el profesor los DSS y los contratos, para tener la certeza de que han comprendido su utilidad.

Conforme desarrollan las colaboraciones, los alumnos van construyendo el diagrama de clases a partir del diagrama conceptual del *Modelo del Dominio*, incorporando los métodos a las clases, añadiendo nuevas asociaciones, y estableciendo la navegabilidad y visibilidad entre clases. Las colaboraciones siempre tienen el mismo esquema: un evento de la interfaz de usuario es recibido por

un objeto controlador, que invoca una interacción entre objetos de las clases del dominio.

Cada grupo mantiene una entrevista con el profesor para repasar las colaboraciones realizadas. En realidad sólo discutimos las cuatro o cinco más significativas que corresponden a los casos de uso básicos. Esta entrevista requiere más tiempo que la primera, ya que el diseño de las colaboraciones no resulta sencillo a los alumnos.

Los únicos aspectos de diseño que se consideran en la etapa del *Modelado del Diseño* son los patrones y la organización en paquetes (lógicamente no sería la definitiva), dejando a un lado las cuestiones relacionadas con las estructuras de datos, persistencia, distribución y diseño de interfaces de usuario. Finalmente se acaba escribiendo el código Java de las clases incluidas en el diagrama de clases final, expresando la secuencia de mensajes de las colaboraciones cómo código de los métodos.

En una última entrevista se revisa todo el modelado realizado por el grupo. Al haber mantenido, al menos, dos revisiones previas, los grupos presentan trabajos que se ajustan al diseño "ideal" previsto por el profesor, quedando sólo por discutir las cuestiones relacionadas con la aplicación de patrones de diseño y la implementación.

En las últimas semanas del curso se dedican dos sesiones prácticas a discutir ejercicios de patrones (puede encontrarse una colección de estos ejercicios en http://dis.um.es/~jmolina), que complementan el trabajo con patrones realizado para el caso práctico.

La documentación que entregan los alumnos refleja el proceso seguido: *Modelo de Casos de Uso* y *Modelo del Dominio*; *DSS*, *Contratos* y *Colaboraciones* (ordenados por caso de uso); *Diagrama de Clases* (Modelo y Controladores); descripción de los patrones de diseño aplicados y código de las clases.

En cuanto a la evaluación, se valora cada entrevista y la documentación final. En la nota final, la teoría tiene un peso del 40% y las prácticas del 60%. El examen teórico consta de un conjunto de ejercicios sobre patrones de diseño.

4. Valoración de la aplicación del proceso

En esta sección presentaremos una valoración del trabajo práctico de nuestros alumnos descrito en la sección anterior, fruto de nuestra experiencia en los últimos tres cursos en los cuales hemos revisado unas 120 prácticas de modelado con UML. Como resultado de este trabajo de revisión y de las entrevistas con los alumnos hemos detectado las principales dificultades que encuentran, los errores más comunes y los beneficios e inconvenientes de nuestro enfoque para el aprendizaje de los alumnos. Hemos organizado la discusión de esta sección de acuerdo con la etapas del proceso y al final comentamos la valoración de los alumnos.

4.1. Modelado de Casos de Uso

En la explicación del proceso se debe insistir mucho en que especificar casos de uso no es una actividad de dibujar un diagrama de casos de uso sino de escribir con precisión el flujo principal y los flujos alternativos: "centrado en la escritura en vez del dibujo", tal y como refleja el título de la principal referencia sobre casos de uso [2]. También hay que insistir en no preocuparse demasiado por las relaciones entre casos de uso (*inclusión*, *extensión* y *generalización*), ni de las relaciones entre actores, ya que ello supone una perdida de tiempo en discusiones que, normalmente, no aportan nada.

Hay que conseguir que el alumno comprenda que el objetivo inicial es identificar los actores y a partir de sus objetivos encontrar los casos de uso, siendo el diagrama de casos de uso una ayuda visual que permite ver de forma simple y directa qué actores hay y cuáles son sus objetivos.

No obstante, unos pocos grupos dedican mucho tiempo a encontrar relaciones de inclusión y extensión, llegando a establecer redes complicadas, por entender que su modelado de casos de uso tiene mayor valor si logran obtener un complicado diagrama.

A través de los ejemplos que se discuten en las sesiones de ejercicios, se consigue que los alumnos adquieran una idea de la *granularidad* o tamaño de un caso de uso. Creemos que la introducción al modelado de negocio ayuda a que adquieran esa idea, al conocer la existencia de procesos de negocio que involucran tareas (objetivos del usuario). Por otra parte, al principio muchos alumnos identifican funciones elementales como casos de uso; por ejemplo, dentro de un sistema de compra por internet,

modelan como caso de uso *"Añadir producto al carro de la compra",* en vez de definir un caso de uso *"Realizar Pedido"*.

Un error común es no encontrar los casos de uso que corresponden a actividades que lanza el sistema de forma automática; por ejemplo *"Anular reservas pasado cierto número de días"* o *"Enviar catálogo al inicio del mes"*. Otro error frecuente es englobar bajo un único caso de uso *"Gestión de X"* todas las operaciones *CRUD* sobre un objeto de negocio (alta, consulta, borrado, actualización). Les aclaramos que eso no es un caso de uso, y que lo indiquen aparte como una función del sistema, y sólo incluyan como casos de uso aquellas operaciones *CRUD* relevantes para el sistema: por ejemplo, *"Alta de un cliente en una tienda virtual"* sí debería modelarse como un caso de uso, pero no *"Alta de un producto"*.

Un problema frecuente en la identificación de actores es no establecer bien la frontera del sistema, de modo que se definen actores que realmente no interactúan con el sistema, como es el caso frecuente de clientes de un negocio que no tienen acceso a la aplicación. Los alumnos suelen caer en este error, aunque el caso de estudio del libro de Larman, el terminal de punto de venta, sirve para ilustrar bien el problema, ya que el cliente que realiza la compra no es un actor del sistema, sino que el único actor es el cajero.

También hemos conseguido que los alumnos escriban casos de uso independientes de la interfaz de usuario, al nivel *esencial* que comenta Larman. Los casos de uso que tienen en cuenta las interfaces de usuario se escribirían antes de la implementación, pero en nuestro caso no se hace ya que no llegamos a esa etapa.

Finalmente, destacamos que el libro de Larman incluye un excelente capítulo sobre casos de uso que permite a los alumnos elaborar una lista de las cuestiones básicas a tener en cuenta en el modelado de casos de uso.

4.2. Modelo del Dominio

Los alumnos no tienen problemas con el modelado del dominio ya que tienen experiencia de asignaturas anteriores con los modelos de datos *Entidad-Relación*. Les recomendamos que: i) realicen el modelado del dominio al mismo tiempo que escriben los casos de uso (conforme aparecen los conceptos se incluyen en el diagrama), ii) no se preocupen demasiado por completar las asociaciones y iii) siempre que identifiquen una clasificación, la modelen como una generalización, sin preocuparse si luego dará lugar o no a una herencia de clases.

Los alumnos no identifican todos los conceptos la primera vez; les insistimos en que incluyan los conceptos del dominio sin plantearse si darán lugar o no a clases. Otro error común es no distinguir entre la especificación de un concepto y un ítem de ese concepto; por ejemplo distinguir entre la descripción de una película y una cinta concreta de esa película. También encuentran dificultad para ver la conveniencia de modelar como un concepto, en vez de cómo atributos, informaciones tales como cantidades con unidades o ciertos identificadores (película, cuenta bancaria) que están formados de varias partes, a los que más tarde tendrá sentido asociarles alguna operación.

En general, a los alumnos les cuesta trabajo entender que no se trata de *pensar en clases* sino en el vocabulario del dominio extraído de los requisitos, a partir del cual se establecerán las clases del *Modelo* (o del *Negocio*) del sistema.

4.3. Modelo del Análisis

Los DSS resultan útiles para identificar los eventos generados por un actor hacia el sistema durante un caso de uso. Asociar a cada evento una operación a realizar por el sistema, descrita por un contrato, constituye un modo sistemático de identificar las colaboraciones que realizan un caso de uso y que finalmente darán lugar al código de la aplicación. De esta forma se rompe la falsa idea de que un caso de uso se realiza mediante una única colaboración.

La especificación de un DSS no suele presentar problemas; el alumno lee el flujo de eventos del caso de uso (el principal o uno alternativo) para encontrar cada evento lanzado al sistema por el actor.

Aunque en la segunda edición de su libro Larman comenta que los contratos sólo son de interés en aquellas situaciones en los que *"los detalles y complejidad de los cambios de estado requeridos son difíciles de capturar en los casos de uso",* nosotros obligamos a los alumnos a especificar siempre el contrato, ya que creemos

que son de gran ayuda para modelar las colaboraciones y no suponen mucho trabajo extra.

Sólo deben escribir la precondición y postcondición de la operación del sistema. Lo más importante es, sin duda, la especificación de la postcondición. En cuanto a la forma de escribir estos asertos, les aclaramos que no es necesario usar un lenguaje formal, sino que el lenguaje natural es suficiente. La mayoría prefieren el modo informal y sólo unos pocos grupos escogen OCL o la notación de asertos de Eiffel que ya conocen de la asignatura de *Introducción a la Programación OO*.

También se debe insistir en no dediquen mucho tiempo a escribir postcondiciones lo más completas posibles, sino que lo consideren una primera aproximación que les ayudará a establecer la colaboración; esto es una alternativa a comenzar directamente con el diagrama de colaboración: "*pensar antes de dibujar"*.

El diseño de las colaboraciones es la tarea más compleja a la que se enfrentan los alumnos durante su trabajo práctico y a la que dedican más tiempo. Esto es lógico ya que es bien conocido que la distribución de responsabilidades entre clases y el establecimiento de las interacciones entre objetos es la parte más difícil del modelado OO [3,7], y su dominio es imprescindible para una buena programación OO.

Cabe señalar que aquellos alumnos que no han recibido un curso de *Introducción a la Programación OO*, ya que proceden de universidades en las que no es obligatorio dentro del plan de estudios o proceden de las titulaciones técnicas de nuestro Centro (hasta el nuevo plan que ha comenzado a impartirse este curso, dicha asignatura era optativa), se encuentran con grandes problemas cuando deben modelar las colaboraciones y con frecuencia abandonan la práctica (estos alumnos tienen muchas dificultades para aprobar la asignatura, necesitando más de un curso, ya que deben dedicar un esfuerzo personal para comprender los principios de la OO).

Se recomienda que primero construyan diagramas de colaboración en vez de diagramas de secuencia, ya que pueden especificar los enlaces entre objetos y conocer si es posible que un objeto pueda enviar cierto mensaje a otro objeto que es accesible, aunque finalmente también manejen diagramas de secuencia por resultarles más útiles al mostrar la secuencia ordenada de mensajes.

Los errores más comunes que cometen los alumnos al diseñar colaboraciones son: i) un objeto envía un mensaje a un objeto inaccesible, ya que no tiene visibilidad sobre él; el problema puede ser que falta una asociación entre las clases o que es necesario delegar la responsabilidad en una cadena de objetos o es necesario que el controlador del caso de uso obtenga acceso a ese objeto, ii) se viola el patrón GRASP *Experto*, no asignando una responsabilidad a la clase que tiene la información para cumplimentarla, iii) no se obtiene el objeto que corresponde a un *string* de entrada capturado a través de la interfaz, y se pasa dicho *string* como argumento a métodos que en realidad requieren el objeto, iv) se viola el patrón GRASP *Creador*, un objeto no es creado por quién debería hacerlo, v) se olvida la creación de cierto objeto, y luego se accede a él, vi) un objeto se envía sobre sí mismo un mensaje para obtener una información de la que dispone, asociada a un atributo, vii) se confunde la clase catálogo con la estructura de datos que almacena los objetos del catálogo. Sin duda, los tres primeros errores son los más importantes y los que se cometen con facilidad cuando no se tiene experiencia modelando colaboraciones o en programación OO.

Siempre procuramos que el caso práctico propuesto suponga el modelado de cinco o seis colaboraciones complejas (siete u ocho objetos) que requieran que el alumno realmente adquiera destreza en la técnica de encontrar la colaboración apropiada para un caso de uso.

Hemos observado que en muchos cursos sobre análisis y diseño OO la aplicación de un proceso se reduce a obtener casos de uso y diagramas de clases, pero no se presta atención al diseño de colaboraciones, sobre todo en cursos de ingeniería del software que abordan tanto los métodos estructurados como los procesos OO. Esto supone no realizar un verdadero modelado OO.

4.4. Modelo del Diseño e Implementación

Debido a la carga de créditos prácticos de la asignatura, sólo es posible aplicar una iteración y no se tratan cuestiones importantes del diseño, las que realmente dan quebraderos de cabeza en el desarrollo real de una aplicación: incorporar al diseño todas las restricciones impuestas por los requisitos no funcionales. Al menos sería

conveniente que cada grupo llegase a construir un prototipo de la aplicación (sería un primer prototipo evolutivo no desechable) a partir del modelado que han realizado, pero no se les exige ya que no hay tiempo para ello. Los alumnos tan sólo codifican parcialmente los métodos, siguiendo la secuencia de mensajes de las colaboraciones.

Como hemos comentado anteriormente, la práctica permite aplicar los patrones GoF más comunes en el caso de un problema real. La codificación en Java a partir de los modelos creados no les presenta ninguna dificultad y los patrones que suelen aparecer son: *Iterador*, *Composite*, *Estrategia*, *Estado*, *Factoria*, *Método Plantilla* y por supuesto los GRASP.

Con la separación entre *Modelo de Análisis* y *Modelo de Diseño* se consigue que el alumno tenga conciencia de la existencia de una visión ideal del sistema obtenida al contemplar solo los requisitos funcionales y su realización a través de un conjunto de colaboraciones, frente a una visión más detallada del sistema, previa a la implementación, obtenida mediante un refinamiento de la anterior al considerar los requisitos no funcionales.

Los alumnos valoran muy positivamente las prácticas de la asignatura y entienden que han estudiado conceptos y técnicas que les van a ser de una utilidad práctica. En cuanto al proceso, consideran que realmente les ayuda a construir software y valoran sobre todo su trazabilidad: es posible ir desde un caso de uso al código que realmente lo implementa a través de las colaboraciones. También lo consideran un método simple que ofrece un modo sistemático de llegar desde los requisitos a la implementación.

No obstante los alumnos son conscientes que han aplicado el proceso de una forma simplificada, que sirve de punto de partida para su aplicación en un caso real y para que tengan un conocimiento de las técnicas y conceptos que subyacen al análisis y diseño OO.

5. Conclusiones

Creemos que el enfoque presentado en este trabajo es adecuado para que los alumnos adquieran un buen conocimiento de los aspectos básicos que subyacen a la aplicación de un proceso UML en el desarrollo de software OO. A través de la aplicación de un proceso simple pero realista a un caso práctico no trivial, los alumnos adquieren destreza en las técnicas básicas del modelado OO con UML: modelado de casos de usos, modelado conceptual, modelado estructural y de comportamiento (colaboraciones).

El enfoque ideal sería aplicar el proceso a un problema real, llegando a la implementación a través de varias iteraciones y prestando atención al diseño. Nuestro planteamiento supone un compromiso entre el ideal y las restricciones que impone la asignatura en cuanto a carga docente. Con una carga práctica de seis créditos se podría alcanzar ese ideal.

Referencias

[1] *Computing Curricula 2001*, Final Report, December 2001, ACM e IEEE.

[2] A. Cockburn, Writing Effective Use Cases, Addison-Wesley, 2001.

[3] E. Gamma et al., *Design Patterns, Elements of Reusable Object-Oriented Software*, Addison-Wesley, 1995.

[4] J. García Molina et al. *Towards Use Case and Conceptual Models Through Business Modeling*, ER2000: 19th Int. Conf. on Conceptual Modeling, USA, 2000.

[5] J. García Molina et al., *Una Propuesta para Organizar la Enseñanza de la Orientación a Objetos*, VIII Jornadas de Enseñanza Universitaria de la Informática (JENUI'02), Cáceres, 2002.

[6] I. Jacobson et al., *El Proceso Unificado de Desarrollo de Software*, Addison-Wesley, 2000.

[7] C. Larman, *UML y Patrones*, 2ª edición, Prentice-Hall, 2002.

[8] S. Lilly, *How to Avoid Use-Case Pitfalls*, 2000, http://www.sdmagazine.com

[9] RUP. Debe ser adquirido de *Rational Software*: documentación on-line, http://www.rational.com/products/rup

Elaboración de Documentos de Requisitos en Asignaturas de Ingeniería del Software

Francisco J. García Peñalvo, Mª N. Moreno García
Dpto. de Informática y Automática
Universidad de Salamanca
37008 Salamanca
e-mail: fgarcia@usal.es, mmg@usal.es

Amador Durán Toro
Dpto. de Lenguajes y Sistemas Informáticos
Universidad de Sevilla
41012 Sevilla
e-mail: amador@lsi.us.es

Resumen

La relación entre la calidad de los requisitos y del producto final es hoy en día algo ampliamente asumido dentro de la Ingeniería del Software. Al contrario que hace unos años, en los que simplemente se hablaba del *análisis* de requisitos que se suponían proporcionados por el cliente, actualmente se reconoce como necesario un proceso mucho más complejo, la *Ingeniería de Requisitos*, en el que deben participar de forma activa tanto clientes y usuarios como desarrolladores de software. Tradicionalmente, la enseñanza de los temas relacionados con los requisitos se ha centrado más en técnicas de análisis, básicamente modelado, que en otras más importantes como puede ser la elicitación. En este artículo se presenta cómo se aborda la Ingeniería de Requisitos en la Ingeniería Informática en la Universidad de Salamanca, y el impacto que ha tenido la adopción de la metodología desarrollada en el Departamento de Lenguajes y Sistemas Informáticos de la Universidad de Sevilla.

1. Introducción

La rama de la ingeniería de software conocida como *Ingeniería de Requisitos*[1] (IR) está recibiendo cada vez más atención por parte de las comunidades académica y profesional. Los resultados presentados en los informes *CHAOS* [14] y *ESPITI* [8] confirman con resultados de proyectos reales que las principales causas, tanto de éxito como de fracaso, de los proyectos de desarrollo de software están directamente relacionadas con los requisitos. En [12] se pone también de manifiesto que la formación en IR es una de las principales carencias detectadas por los profesionales de las tecnologías de la información una vez que abandonan la Universidad y comienzan su carrera profesional.

Este cambio de mentalidad respecto a la importancia de los requisitos se puede apreciar claramente en la nomenclatura utilizada para hacer referencia a las actividades relacionadas con los mismos. Hasta principios de los 90, se consideraba que la única actividad relacionada con los requisitos era el *análisis de requisitos*. Es decir, se asumía que el cliente proporcionaba los requisitos y que los *analistas* se limitaban a *analizar* las peticiones de los clientes a partir de las cuales se diseñaba y se implementaba el nuevo sistema. Una vez que se observó que está actitud *pasiva* de los analistas conducía al fracaso de la mayoría de los proyectos, se impuso la idea de que la elaboración de los requisitos no podía considerarse una responsabilidad única del cliente sino que debía ser una labor conjunta entre clientes, usuarios, desarrolladores y cualquier otra persona con algún interés en el sistema a desarrollar, lo que se conoce como *stakeholder*.

1.1. El Proceso de Ingeniería de Requisitos

Este cambio de actitud hace necesaria un nuevo proceso, la IR, que ya no se limita a analizar las peticiones de los usuarios, sino que incluye los siguientes subprocesos (ver Figura 1):

[1] Algunos autores sitúan a la IR como parte de la *Ingeniería de Sistemas*, en la que se desarrollan sistemas compuestos tanto por hardware como por software. En este artículo consideraremos que los sistemas de nuestro interés son compuestos básicamente por software, por lo que la IR de la que hablamos será de la *IR de sistemas software*.

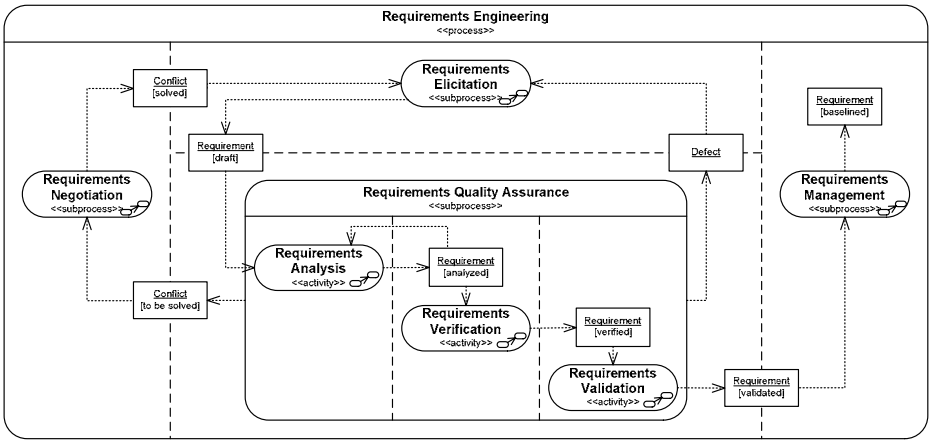


Figura 1. Proceso de Ingeniería de Requisitos (sólo se muestran algunos productos)

- *Elicitación de requisitos*: este subproceso, probablemente el más crítico y el más difícil de realizar, tiene como objetivos buscar, investigar y ayudar a los clientes y usuarios a documentar sus necesidades. La documentación de los requisitos deberá hacerse siempre usando el vocabulario de clientes y usuarios, de forma que éstos puedan entenderlos, siendo lo más habitual emplear lenguaje natural. Las técnicas más comunes son las entrevistas, reuniones en grupo, estudio *in situ*, etc. [2][13].
- *Análisis de requisitos*: esta actividad, parte del subproceso de aseguramiento de la calidad, tiene como objetivo principal detectar conflictos en los requisitos elicitados, normalmente mediante técnicas de modelado conceptual (estructuradas, orientadas a objetos o formales) y de prototipado de interfaz de usuario. Los modelos generados son también una importante herramienta de comunicación con diseñadores y programadores [2][13].
- *Verificación de requisitos:* esta actividad de calidad tiene como objetivo detectar defectos en los requisitos previamente analizados, normalmente mediante técnicas como revisiones formales, listas de comprobación (*checklists*), etc.
- *Validación de requisitos:* esta tercera actividad de calidad intenta asegurar que los requisitos verificados reflejan realmente las necesidades de clientes y usuarios. Las técnicas empleadas suelen ser reuniones en las que se revisan los requisitos mediante el apoyo de prototipos de interfaz de usuario.
- *Negociación de requisitos:* el objetivo de este subproceso es buscar soluciones a los conflictos detectados que satisfagan a los distintos *stakeholders*.
- *Gestión de requisitos:* este subproceso gestiona todo el proceso, en especial las peticiones de cambios en los requisitos, el impacto de dichas peticiones, las distintas versiones de los requisitos, etc.

1.2. Educación en Ingeniería de Requisitos

Normalmente, los cursos de introducción a la Ingeniería del Software, en lo tocante a los tópicos relacionados con los requisitos, se centran fundamentalmente en aspectos de modelado. Esto conlleva un estudio muy somero (incluso una supresión total) de todo lo relacionado con la elicitación y documentación de los requisitos.

Esta situación provoca diversos problemas y malos entendidos a la hora de llevar a cabo una documentación de requisitos cuando el alumno tiene que enfrentarse a la realización de una Especificación de Requisitos del Software (ERS), que incluye tanto requisitos en lenguaje natural como modelos conceptuales, en proyectos software no triviales como puede ser el caso de los proyectos fin de carrera (PFC) de una Ingeniería Informática o un proyecto de desarrollo de software real.

Tratando de evitar esta situación en los estudios de Informática de la Universidad de Salamanca, se ha buscado como objetivo la introducción de una forma unificada de documentar requisitos. Existen diversas propuestas para especificar los requisitos, como por ejemplo el estándar IEEE Std. 830-1998 [10], sin embargo nosotros hemos comprobado que la metodología de elicitación y documentación de requisitos [3][5][6], desarrollada en el Departamento de Lenguaje y Sistemas Informáticos de la Universidad de Sevilla (al que nos vamos a referir como Metodología de Durán y Bernárdez, abreviadamente MDB), se ajusta mejor a los objetivos perseguidos.

Así, en el curso 1998-1999 se introdujo en la asignatura de Ingeniería del Software de la Ingeniería Técnica en Informática de Sistemas (ITIS) dicha metodología MDB, siendo ésta satisfactoriamente aceptada por los alumnos de esta titulación tanto en las prácticas obligatorias de la asignatura de Ingeniería del Software como en la documentación técnica de sus PFC.

Concretamente en este artículo se presenta la evolución del uso que ha tenido esta metodología en la titulación ITIS en la Universidad de Salamanca. El resto del artículo se organiza como sigue: la Sección 2 explica los fundamentos básicos de la metodología de elicitación y documentación de requisitos elegida; la Sección 3 presenta el uso de la metodología MDB en ITIS desde el curso 1998-1999 hasta la actualidad; la Sección 4 introduce someramente diversas herramientas CASE que ofrecen un soporte a esta metodología; finalmente, la Sección 5 cierra el artículo con las conclusiones del mismo.

2. Visión General de la MDB

En este apartado se va a hacer una revisión de los aspectos más destacados de esta metodología con el objeto de que el lector de este artículo conozca sus fundamentos, para una información más detallada de la misma se recomienda la consulta de [3][5][6]. Las tareas propuestas en esta metodología son:

1. Obtener información sobre el dominio del problema y el sistema actual.
2. Preparar y realizar las reuniones de obtención/negociación.
3. Identificar/revisar los objetivos del sistema.
4. Identificar/revisar los requisitos de información.
5. Identificar/revisar los requisitos funcionales.
6. Identificar/revisar los requisitos no funcionales.
7. Priorizar objetivos y requisitos.

Por parte de los autores de esta metodología se recomiendan diversas técnicas para llevar a cabo las tareas anteriores, pero las más importantes son los casos de uso, las plantillas y los patrones lingüísticos.

Los casos de uso son una técnica sumamente conocida y utilizada en los métodos de desarrollo de software actuales. Esta técnica fue propuesta inicialmente por Ivar Jacobson [11] y actualmente se encuentra incluida dentro de UML (*Unified Modeling Language*) [1].

Para la correcta descripción de los casos de uso (que son una forma de expresar requisitos funcionales), los requisitos no funcionales, los requisitos de información, así como de otros requisitos del sistema, se recurre a la utilización de diversos tipos de plantillas, concretamente:

- *Objetivos del sistema.*
- *Requisitos de información.*
- *Requisitos de restricción (reglas de negocio)*
- *Actores.*
- *Casos de uso.*
- *Requisitos funcionales (expresados de forma tradicional, como texto libre).*
- *Requisitos no funcionales.*

Cada plantilla presenta los campos de información necesarios para especificar el concepto que está representando, como ejemplo de plantilla en la Figura 2 se recoge la plantilla utilizada por esta metodología para los casos de uso, que describen requisitos funcionales de una manera operacional.

3. Evolución del Uso de la MDB en la Universidad de Salamanca

Los estudios de Informática en la Universidad de Salamanca se organizan en un primer ciclo de tres años de duración (Ingeniería Técnica en Informática de Sistema) y en un segundo ciclo de dos años de duración (Ingeniería Informática).

Los tópicos relacionados con los requisitos se introducen en la asignatura de Ingeniería del Software, que aparece en el Plan de Estudios de la Ingeniería Técnica en su tercer año. La superación de la parte práctica de esta asignatura obliga a que los alumnos, organizados en grupos, realicen la especificación de un sistema a nivel de análisis y diseño. Esta especificación debe incluir un catálogo de requisitos, pero los estudiantes pueden elegir el método de especificación.

La MDB se introdujo en el temario de la asignatura de Ingeniería del Software en la Universidad de Salamanca en el curso 1998-1999. Desde el momento de su introducción, su uso global en las prácticas de la asignatura sólo fue menor al uso de otros métodos de especificación durante el primer curso, incrementando su presencia en años sucesivos hasta alcanzar un porcentaje de utilización del 85,4% en el curso 2002-2003.

La Figura 3a compara el número de trabajos obligatorios que, dentro de la asignatura de Ingeniería del Software, han utilizado esta metodología de obtención de requisitos frente a otros métodos, típicamente el IEEE Std. 830 [10]. Por su parte, la Figura 3b realiza la misma comparación que la Figura 3a, pero expresando dicha comparativa en términos porcentuales.

La Ingeniería Técnica en Informática de Sistemas finaliza con un PFC. Normalmente, estos proyectos requieren la realización de una documentación técnica que incluirá en alguno de sus anexos un catálogo de requisitos. La utilización de la MDB comienza a aparecer en la documentación de estos proyectos en el año 2000. Durante el año 2000, 17 de los 38 PFC defendidos en la Ingeniería Técnica utilizaron la MDB, esto supone una presencia en el 44,7% de los proyectos leídos en este año. Pero cabe destacar el espectacular incremento de uso de esta metodología en los siguientes años, alcanzando una cota máxima de utilización durante el año 2001 con una presencia en el 84,4% de los PFC defendidos en la Ingeniería Técnica. Esta tendencia se ve refrendada en el año 2003, donde el 88,9% de los proyectos leídos hasta la fecha utilizan este método (actualmente, sólo se tienen datos de la convocatoria de febrero de 2003).

UC-*<id>*	***<nombre/objetivo del caso de uso>***	
Versión	*<nº de la versión actual>* (*<fecha de la versión actual>*)	
Autores	• *<autor$_1$ de la versión actual>* (*<organización del autor$_1$>*) ...	
Fuentes	• *<fuente$_1$ de la versión actual>* (*<organización de la fuente$_1$>*) ...	
Dependencias	• OBJ-*x* *<nombre del objetivo del que depende este caso de uso>* • R*x*-*y* *<nombre del requisito del que depende este caso de uso>* ...	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso { abstracto durante la realización de los siguientes casos de uso: *<lista de casos de uso>*, cuando *<evento de activación>* [o durante la realización de los siguientes casos de uso: *<lista de casos de uso>*]}	
Precondición	*<precondición del caso de uso>*	
Secuencia normal	**Paso**	**Acción**
	p_1	{El actor *<actor>*, El sistema} *<acción/es realizada/s por actor/sistema>*
	p_2	Se realiza el caso de uso *<caso de uso (UC-x)>*
	p_3	Si *<condición>*, {el actor *<actor>*, el sistema} *<acción/es realizada/s por actor/sistema>*
	p_4	Si *<condición>*, se realiza el caso de uso *<caso de uso (UC-x)>*
	…	…
Poscondición	*<poscondición del caso de uso>*	
Excepciones	**Paso**	**Acción**
	p_i	Si *<condición de excepción>*, {el actor *<actor>*, el sistema} *<acción/es realizada/s por actor/sistema>*, a continuación este caso de uso {continúa, queda sin efecto}
	p_j	Si *<condición de excepción>*, se realiza el caso de uso *<caso de uso (UC-x)>*, a continuación este caso de uso {continúa, queda sin efecto}
	…	…
Rendimiento	**Paso**	**Cota de tiempo**
	p_i	*m <unidad de tiempo>*
	…	…
Frecuencia	*<nº de veces>* veces / *<unidad de tiempo>*	
Importancia	*<importancia del caso de uso>*	
Urgencia	*<urgencia en la implementación del caso de uso>*	
Estado	*<estado de desarrollo del caso de uso>*	
Estabilidad	*<estimación de la estabilidad del caso de uso>*	
Comentarios	*<comentarios adicionales sobre el caso de uso>*	

Figura 2. Plantilla para casos de uso

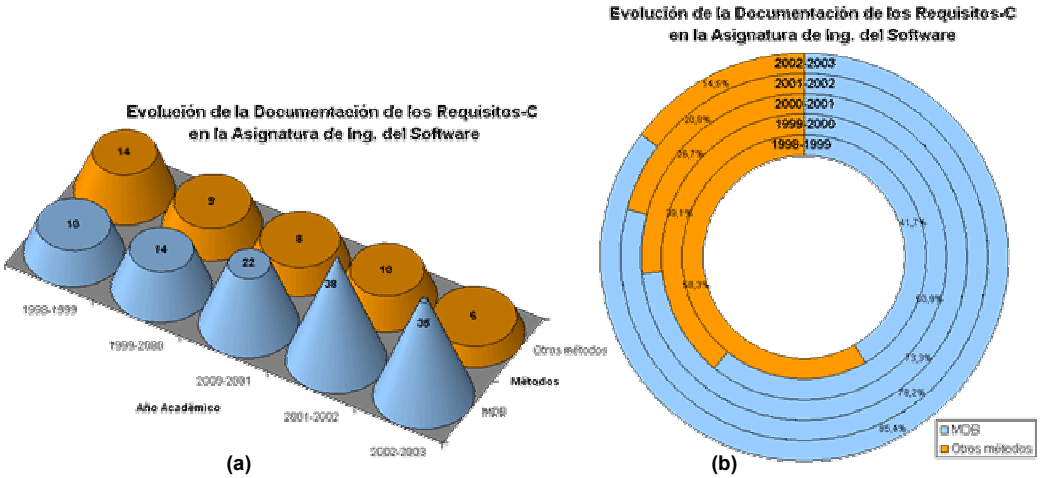


Figura 3. Comparación del Nº de trabajos de Ing. del Software que han empleado la MDB entre 1998-99 y 2002-03

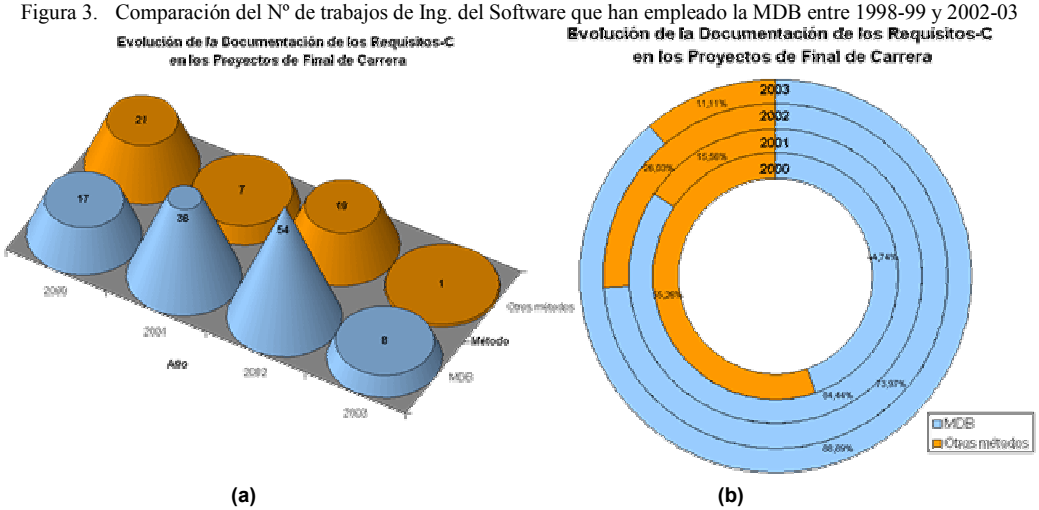


Figura 4. Comparación del Nº de PFC que han empleado la MDB entre los años 2000 y 2003

La Figura 4 presenta la misma comparación realizada con los trabajos prácticos de la asignatura de Ingeniería del Software, pero en este caso orientada a los PFC de la Ingeniería Técnica en Informática de Sistemas defendidos entre 2000 y 2003.

4. Soporte CASE para la MDB

Desde el curso académico 2001-2002 la metodología de obtención y documentación de requisitos está soportada por herramientas CASE.

En esta sección se van a presentar tres herramientas CASE que dan cobertura a esta metodología. La primera de ellas, denominada REM, ha sido desarrollada en la Universidad de Sevilla por uno de los autores de la metodología. Las otras dos han sido desarrolladas en el Departamento de Informática y Automática como soporte para las prácticas de la asignatura Ingeniería del Software y para los PFC.

4.1. REM

REM (*REquirements Manager*) [4] es una

herramienta de gestión de requisitos experimental desarrollada por uno de los autores como parte de su tesis doctoral [3] y recientemente presentada en la sesión de *research tools* de la IEEE *Joint International Conference on Requirements Engineering* celebrada en Eseen (Alemania), en septiembre de 2002 [7]. En REM, un proyecto de Ingeniería de Requisitos está compuesto por los cuatro documentos correspondientes a las cuatro vistas que pueden verse en la figura 5: requisitos (elicitación), modelos conceptuales (análisis), conflictos y defectos (análisis, verificación, validación y negociación), y peticiones de cambio (gestión).

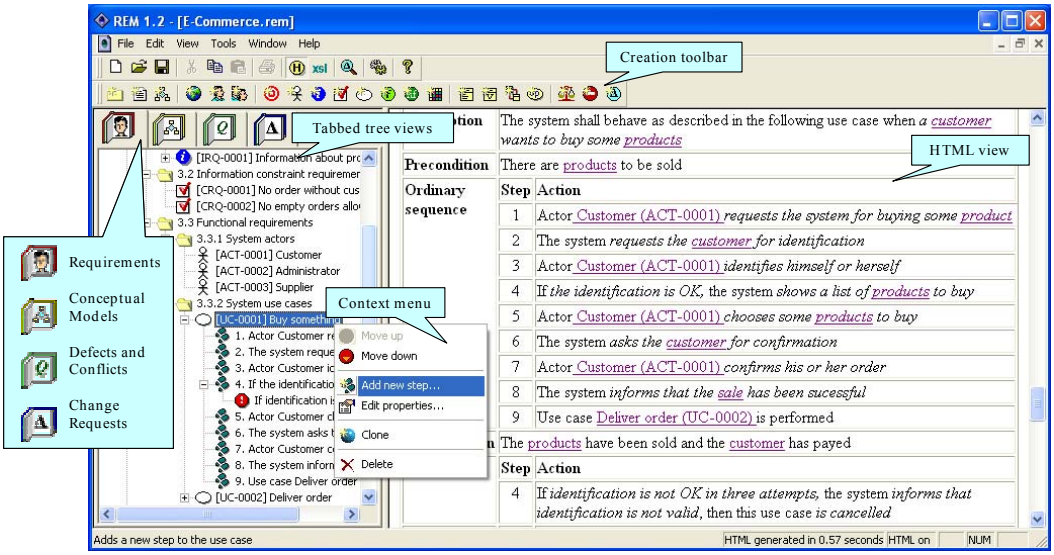


Figura 5. Interfaz de usuario de REM

REM presenta un excelente generador de informes en HTML, para lo cual internamente maneja la información en XML, aplicándole una hoja de estilo XSLT a la hora de generar el correspondiente informe. Las hojas de estilo XSLT son configurables, por lo que permite la visualización de la información de los requisitos de acuerdo a diversos estándares, incluyendo la posibilidad de aplicar heurísticas de verificación automática, tal como se describe en [7]. La Figura 5 presenta la interfaz de REM, que soporta tanto español como inglés.

4.2. GESCAT

GESCAT (http://tejo.usal.es/~fgarcia/) es una herramienta CASE que ofrece un soporte a la realización de las plantillas propias de la MDB.

Esta herramienta no pretende cubrir toda la funcionalidad incluida en REM, sino más bien ser una herramienta de uso docente, que facilite la labor a los alumnos de ITIS en el proceso de gestionar y documentar requisitos.

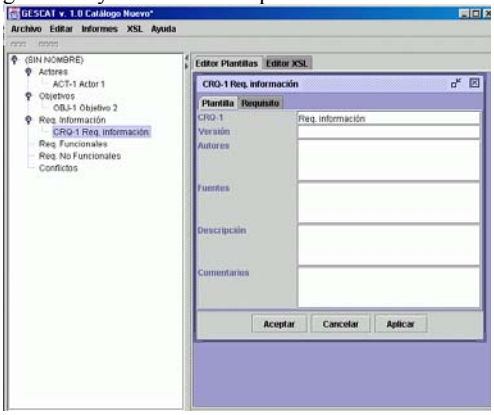


Figura 6. Interfaz de usuario de GESCAT

Esta herramienta se ha desarrollado en Java buscando una independencia de plataforma. Además, y en relación con la posibilidad de generación de informes, GESCAT hace un

profuso empleo de los lenguajes de marcado XML/XSLT, que facilita generar los informes en diferentes formatos como PDF o HTML. La Figura 6 muestra la interfaz de usuario de esta herramienta.

4.3. Left CASE

Left-CASE (http://zarza.usal.es) es una plataforma CASE para entornos de escritorios GNOME 1.4 basada en componentes, que ofrece un amplio soporte a las técnicas de modelado del software que se estudian en la Ingeniería Informática, fundamentalmente en las disciplinas relacionadas con la Ingeniería del Software.

Left CASE está basada en un entorno CASE extensible, donde las técnicas que soporta pueden ampliarse, pero siempre bajo las restricciones de compatibilidad con el entorno.

Para lograr este objetivo se diseña una arquitectura formada por un *framework* base o contenedor y un conjunto de componentes específicos. El contenedor hace las veces de "anfitrión" para los componentes, ofreciéndoles los servicios comunes necesarios, así como el conjunto de interfaces necesarias para que dichos componentes puedan incorporarse al entorno construido, mientras que cada uno de los componentes debe proporcionar todas las características propias de una técnica de modelado concreta [9].

Uno de los componentes desarrollados para integrarse en la plataforma Left CASE es el componente para el modelado de diagramas de casos de uso de acuerdo con UML [1], que para la especificación de cada caso de uso emplea plantillas compatibles con las definidas en la MDB. Dado que Left CASE está desarrollada para entorno GNOME 1.4, la plataforma de ejecución por excelencia será Linux. La interfaz de esta herramienta soporta tanto español como inglés, y en la Figura 7 se aprecia la interfaz de Left CASE con el componente para el modelado de casos de uso cargado.

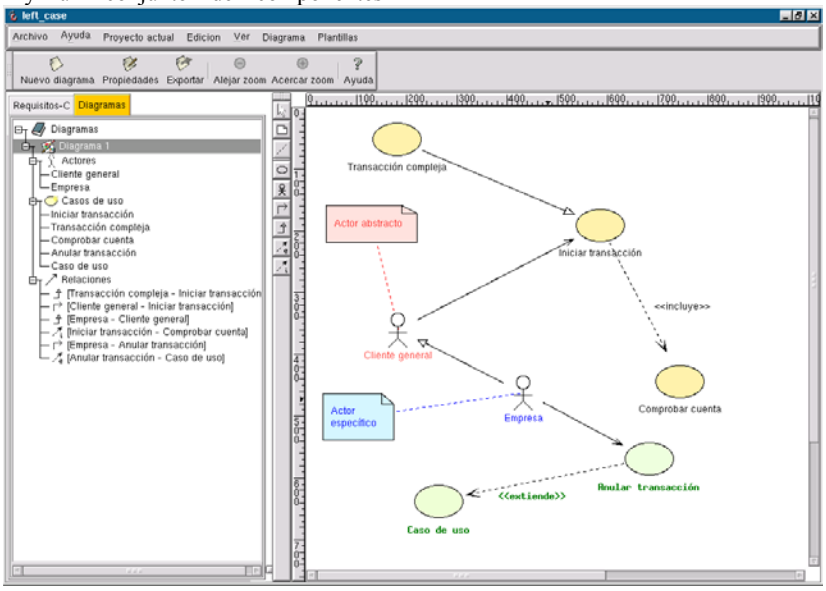


Figura 7. Interfaz de usuario de Left CASE con el Componente CU

5. Conclusiones

Este artículo analiza el uso de una metodología concreta para la obtención y documentación de requisitos en la titulación de Ingeniería Técnica en Informática de Sistemas en la Universidad de Salamanca. Dicha metodología ha sido definida dentro del Departamento de Lenguajes y Sistemas Informáticos de la Universidad de Sevilla.

La metodología en cuestión se basa en una unidad básica de documentación, las plantillas, que se emplean para especificar los diferentes

conceptos que se manejan en la fase de Ingeniería de Requisitos, como por ejemplo objetivos, casos de uso, requisitos funcionales, etc.

Las plantillas utilizan el lenguaje natural como base de su contenido, haciéndolas muy comprensibles tanto a los usuarios de las mismas como a sus destinatarios.

Gracias a su sencillez, este método resulta más familiar y fácil de aplicar y comprender que otras propuestas más complejas como por ejemplo el IEEE Std. 830 [10]. La simplicidad de este método es el punto clave con el que se puede justificar la gran acogida que ha tenido por parte de los alumnos. Esta aceptación queda reflejada en las Figuras 3 y 4 que muestran cómo esta metodología está siendo ampliamente utilizada tanto en los trabajos prácticos de la asignatura Ingeniería del Software como en los PFC.

Su utilización en la asignatura Ingeniería del Software es un resultado que no resulta sorpresivo dado que esta metodología aparece citada en diversas ocasiones a lo largo del programa impartido en la parte teórica de la asignatura (aunque también se menciona y se describe someramente el IEEE Std. 830). Sin embargo, su amplia utilización en los PFC si que representa un claro síntoma de aceptación, porque en dichos proyectos las influencias de los tutores son diversas y el alumno tiene plena libertad para elegir las metodologías que mejor se ajusten a la temática de sus trabajos.

La existencia de herramientas CASE compatibles con esta metodología, disponibles y de libre uso, es sin duda alguna otra ventaja añadida para el éxito y aceptación de esta propuesta metodológica.

Desde el punto de vista de su utilización en docencia, la mayor fuente de errores que comenten los alumnos es su falta de rigor a la hora de rellenar los campos de la plantilla relacionados con el Rendimiento, la Frecuencia, la Importancia y la Urgencia, fundamentalmente debido a que sus proyectos tienen tamaños limitados, donde no les resulta obvio aportar dichos datos.

Agradecimientos

Este trabajo ha sido parcialmente subvencionado por la Consejería de Educación y Cultura de la Junta de Castilla y León mediante el proyecto US23/02.

Referencias

[1] Booch, G., Rumbaugh, J., Jacobson, I. *The Unified Modeling Language User Guide.* Addison-Wesley, 1999

[2] Brackett, J.W. *Software Requirements.* SEI Curriculum Module SEI-CM-19-1.2. Software Engineering Institute. January 1990

[3] Durán, A. *Un Entorno Metodológico de Ingeniería de Requisitos para Sistemas de Información.* Tesis Doctoral, U. de Sevilla, 2000

[4] Durán, A. *REM Web Page.* U. de Sevilla. http://klendathu.lsi.us.es/REM/. 2002

[5] Durán, A., Bernárdez, B. *Metodología para la Elicitación de Requisitos de Sistemas Software (versión 2.3).* Informe Técnico LSI-2000-10, U. de Sevilla. Abril 2002

[6] Durán, A., Bernárdez, B., Ruiz, A., Toro, M. *A Requirements Elicitation Approach Based on Templates and Patterns.* En *Proceedings of WER'99*, Buenos Aires, Argentina. 1999

[7] Durán, A., Ruiz, A., Corchuelo, R., Toro, M.. *Supporting Requirements Verification Using XSLT.* En *Proceedings of the IEEE Joint International Conference on Requirements Engineering*. Essen, Alemania. 2002

[8] European Software Institute. *ESPITI – European User Survey Results*. Informe Técnico ESI–1996–TR95104, 1996

[9] García, F. J., Álvarez, I. *Plataforma CASE Basada en Componentes para la Docencia de Ingeniería del Software.* En las *Actas del IE-2002*. 2002

[10] IEEE. *IEEE Software Engineering Standards Collection 1999 Edition. Volume 4: Resource and Technique Standards*. IEEE Computer Society Press, 1999

[11] Jacobson, I. *Object Oriented Development in an Industrial Environment.* En *Proceedings of the OOPSLA'87*. ACM, 1987

[12] Lethbridge, T.C. *Priorities for the Education and Training of Software Engineers. Journal of Systems and Software,* 53. 2000

[13] Raghavan, S., Zelesnik, G., Ford, G. *Lecture Notes on Requirements Elicitation.* Educational Materials. CMU/SEI-94-EM-10. Software Engineering Institute. March 1994

[14] The Standish Group. TSG. *The CHAOS Report*. 1995

Inteligencia artificial

Un enfoque metodológico para la docencia en Ingeniería del Conocimiento

Amparo Alonso Betanzos, Bertha Guijarro Berdiñas
Departamento de Computación
Universidad de A Coruña
15071 A Coruña
e-mail: ciamparo@udc.es, cibertha@udc.es

Adolfo Lozano Tello
Departamento de Informática
Universidad de Extremadura
10071 Cáceres
e-mail: alozano@unex.es

Resumen

En esta ponencia se presenta una propuesta para la docencia de la Ingeniería del Conocimiento en la Ingeniería en Informática. El planteamiento se orienta al desarrollo de sistemas basados en conocimiento siguiendo la metodología *CommonKADS*. Este enfoque es coincidente en casi su totalidad en las asignaturas de "Ingeniería del Conocimiento" en la Universidad de A Coruña y de "Inteligencia Artificial e Ingeniería del Conocimiento" en la Universidad de Extremadura. En el artículo se expone cómo llevar a cabo esta idea y las ventajas e inconvenientes encontrados.

1. Introducción y motivación

El contenido de los temarios para la enseñanza de la Ingeniería del Conocimiento (IC) para la Ingeniería en Informática en las universidades españolas es muy variado. Algunos proyectos docentes para las asignaturas que imparten esta materia se han elaborado siguiendo fundamentos tradicionales y consolidados en IC, otros complementan a asignaturas cursadas en la titulación relacionadas con la Inteligencia Artificial (IA) y otros apuestan por un enfoque metodológico similar al perseguido en Ingeniería del Software. Es necesario, pues, identificar y debatir los distintos enfoques y las ventajas e inconvenientes de cada uno de los proyectos docentes en esta materia.

Una fuente importante de recomendaciones en la elaboración de un proyecto docente la constituye el *Computing Curricula 2001* [3]. Dentro del área de *Sistemas Inteligentes* todas las unidades de estudio propuestas se catalogan dentro de cursos de nivel intermedio. Como curso avanzado dentro de esta área se propone, entre otros, uno sobre *Sistemas Basados en Conocimiento* donde podría englobarse la asignatura *Ingeniería del Conocimiento*. Sin embargo, hasta el momento, no se han elaborado las descripciones de estos cursos para los que tampoco existen recomendaciones en el *Computing Curricula '91* [1].

Por otro lado, la variabilidad de la asignatura IC en las Universidades españolas se debe, en parte, a la organización de los descriptores de la troncalidad *Inteligencia Artificial e Ingeniería del Conocimiento* (RD 1459/1990. B.O.E. 26/10/90). En ocasiones existe una asignatura de IA genérica que engloba la IC. En otros casos existe una asignatura específica de IC que recibe diferentes nombres en las distintas Universidades: *Ingeniería del Conocimiento, Ingeniería del Conocimiento: Sistemas Expertos*, *Ampliación de Ingeniería del Conocimiento, Ingeniería de los Sistemas Basados en el Conocimiento,* etc. En cuanto al carácter obligatorio u optativo de la asignatura, su carácter cuatrimestral o anual y su número de créditos, también la situación es variable.

Todas estas diferencias dificultan la elaboración de unas directrices comunes en cuanto al temario de la asignatura tal como se evidencia también en los encuentros de docentes de IA [4] y a pesar de los avances que suponen estos trabajos. Podemos encontrar dos enfoques predominantes: un enfoque práctico de la asignatura orientado a la enseñanza de una metodología de IC, y un enfoque más orientado a profundizar en las técnicas empleadas en la IC (adquisición y representación de conocimiento, métodos de resolución de problemas, etc.) Existen, no obstante, una serie de temas básicos comunes a

ambos enfoques, como la adquisición de conocimiento, los procedimientos de evaluación de SSBBC, o la importancia que se concede a la parte práctica de la asignatura dada su naturaleza experimental.

Dada la ausencia de una especificación homogénea, hemos recurrido a un análisis histórico del desarrollo de la disciplina para llegar a una definición de ésta que nos permita sentar las bases de la orientación de la asignatura.

Tal como se explica en [8], después del entusiasmo inicial en el área, surge una crisis a finales de los 80 y el pesimismo en cuanto a las posibilidades de aplicación real de los SSBBC debido a un fracaso generalizado en el campo comercial. Había problemas en cuanto a la estimación del tiempo de desarrollo, el incumplimiento de las expectativas iniciales, y la dificultad del, el mantenimiento y verificación de dichos sistemas. Algunas de las causas de esta crisis fueron la ausencia de una metodología estructurada y específica para el desarrollo de estos sistemas, y el cuello de botella que suponía la adquisición de conocimiento. La situación era análoga a la que había tenido lugar en la llamada crisis del software de finales de los años 60. Así, la misma necesidad de establecer planteamientos más metodológicos que dio lugar al nacimiento de la Ingeniería del Software (IS) en el campo de los sistemas de información, en el campo de la IA hizo surgir una nueva disciplina, llamada IC, que trataba de dar una visión más ingenieril al proceso de construcción de SSBBC. Así, la IC puede definirse como:

"La disciplina tecnológica que se centra en la aplicación de una aproximación sistemática, disciplinada y cuantificable al desarrollo, funcionamiento y mantenimiento de SSBBC. En otras palabras, el objetivo último de la IC es el establecimiento de metodologías que permitan abordar el desarrollo de SSBBC de una forma más sistemática" [10].

Actualmente, se han producido importantes avances en el campo, se han desarrollado diversas metodologías, de las que las más recientes y exitosas están basadas en el *modelado del conocimiento*. Un ejemplo de ello es la metodología *CommonKADS* [9] que se ha convertido en un estándar europeo para el desarrollo de SSBBC.

2. Las asignaturas de Ingeniería del Conocimiento en las Universidades de A Coruña y Extremadura

En la Universidad de A Coruña (UDC) existe una asignatura específica de IC, denominada "Ingeniería del Conocimiento" (ICC), obligatoria y cuatrimestral de 5º curso, y que tiene un total de 4,5 créditos, de los cuales 3 son de teoría y 1,5 de práctica (Suplemento al B.O.E. 23/11/94).

En la Universidad de Extremadura (UEX), la asignatura "Inteligencia Artificial e Ingeniería del Conocimiento" (IAICE) es troncal de 4º curso y se compone de 9 créditos, de los cuales 6 son teóricos y 3 prácticos [5]. En esta asignatura se dedica el primer cuatrimestre a impartir conceptos relacionados con la IC, ya que no existe en la titulación ninguna otra asignatura con descriptores relacionados con este campo.

A pesar de la diferencia en la disposición en los planes de estudio de ambas asignaturas, hemos detectado una gran similitud respecto a los objetivos, contenido de temarios, metodología seguida y propósitos de las prácticas, de ICC y el primer cuatrimestre de IAICE [1]. Este hecho nos ha llevado a confeccionar esta ponencia en la que señalaremos los puntos comunes y comparamos nuestro enfoque con otros seguidos en otras universidades sobre esta materia.

Desde un punto de vista general, las asignaturas indicadas se relacionan con tres grandes bloques temáticos:

- El bloque de asignaturas de Ingeniería del Software, ya que ha sido una fuente de conocimientos en la construcción de metodologías de IC.
- En la UDC, el bloque de asignaturas de Sistemas Inteligentes, ya que la IC es una de sus subáreas. De este bloque, cabría destacar, la relación con la asignatura troncal de IA al ser ICC una asignatura que amplía los créditos asignados a la troncalidad. Otras asignaturas optativas (como sistemas expertos, redes de neuronas artificiales, visión artificial, etc.) completan el escenario básico. En la UEX, en cambio, no existe ninguna asignatura que tenga una relación

[1] A partir de ahora, con las siglas IAICE nos referimos al primer cuatrimestre de la asignatura, que se dedica a la Ingeniería del Conocimiento.

directa con IAICE, por lo que se realiza un enorme esfuerzo por sintetizar los contenidos de esta materia.

- El bloque de asignaturas de *Fundamentos Matemáticos* que incluye, por ejemplo, *Lógica*.

2.1. Formulación de Objetivos

Además de los objetivos generales de toda educación universitaria, nos encontramos con una serie de objetivos específicos de esta materia. El programa de las asignaturas necesariamente va a tener que adaptarse al carácter dinámico de esta disciplina, sin dejar de ofrecer al alumno una base perdurable para que éste pueda adaptarse a los cambios que, sin duda, se producirán en el futuro.

Además, desde el punto de vista de la metodología docente, se realizará un énfasis especial en la visión ingenieril. Según Schreiber [9] el proceso de desarrollo de un SBC "implica elicitar, estructurar, formalizar y operacionalizar la información y el conocimiento implicado en el dominio de un problema intensivo en conocimiento para construir un programa que pueda realizar una tarea compleja de forma adecuada". De este modo, como parte fundamental de la docencia en ICC e IAICE se proporcionará una metodología concreta de desarrollo de SSBBC, y las prácticas de las asignaturas se conducirán siguiendo las fases de un proyecto real de análisis, diseño, implementación y evaluación de una aplicación determinada en un dominio de conocimiento concreto siguiendo la metodología enseñada y con la ayuda de un entorno de desarrollo determinado, específico para este tipo de sistemas.

Así, como objetivos específicos de ambas asignaturas cabe enumerar los siguientes:

1. Definir qué es la IC a partir de las asignaturas básicas con las que se relaciona, que son la IA y la IS, y mostrarla como un campo más, aunque muy actual, para el desarrollo de software dentro de la Informática.
2. Comprender la naturaleza, posibilidades y limitaciones de los SSBBC y mostrar su uso en casos reales de interés.
3. Conocer la problemática particular asociada a la construcción de SSBBC, los roles que participan en un proyecto de conocimientos y las aproximaciones metodológicas aportadas para resolver los problemas comentados.
4. Conocer la aproximación de Modelado de Conocimiento, y aprender a aplicarla a través de alguna metodología basada en este enfoque, para modelar las diferentes perspectivas que tiene un SBC.
5. Definir y establecer el ámbito de aplicación de las diferentes técnicas que se pueden utilizar para la adquisición de conocimiento, tanto manuales como automáticas.
6. Establecer los problemas generales asociados con las diferentes etapas de evaluación de los SSBBC y ser capaz de evaluar su funcionalidad.
7. Proporcionar a los estudiantes experiencia práctica en los objetivos mencionados anteriormente, de forma que sean capaces de desarrollar un proyecto de SSBBC .
8. Conseguir que los alumnos conozcan las áreas de investigación y aplicación de los SSBBC e integren con éxito lo aprendido en su vida profesional.

2.2. Selección de contenidos

En el caso de la IC se dispone de una colección amplia y robusta de técnicas básicas, pero los métodos para aplicarlas de forma sistemática a problemas reales todavía no están lo suficientemente extendidos y estandarizados. En las dos últimas décadas un consorcio industria-universidad bajo el programa Europeo SPRIT, extendido posteriormente en el Programa ESPRIT IT, elabora la metodología *CommonKADS* basada en el modelado de conocimiento y que presenta una clara tendencia convergente con las técnicas empleadas en la IS. En la actualidad, se ha convertido en un estándar de facto para el análisis de conocimiento y el desarrollo de SSBBC.

Los programas de teoría de ambas asignaturas contemplan esta nueva metodología, junto con las técnicas de adquisición de conocimiento y de evaluación de este tipo de software. Por otra parte, los programas de prácticas contemplan, paralelamente al programa teórico, la aplicación secuencial de los conocimientos adquiridos, lo que requiere una planificación temporal estricta.

2.3. Programas de teoría

En concreto, los contenidos de los créditos teóricos de la asignatura ICC se articulan de la forma siguiente:

1. La Ingeniería del conocimiento
   - El conocimiento y su contexto
   - La Ingeniería del Conocimiento
   - Características de los SSBBC
2. Metodologías para la construcción de SSBBC
   - Diferencias entre la IS y la IC
   - Metodologías adaptadas de la IS: Descripción y Problemas.
   - Metodologías basadas en el modelado. Generalidades
3. Análisis de viabilidad e impacto: modelado del contexto en CommonKADS
   - Modelos de organización, tareas y agentes
   - Un caso de estudio
4. Descripción conceptual del conocimiento en CommonKADS
   - Modelos de conocimiento y comunicación
   - Elementos reutilizables
   - Un caso de estudio
5. Del análisis a la implementación en CommonKADS
   - Principio de "Conservación de la Estructura"
   - El modelo de diseño. Fases
6. Técnicas para la adquisición del conocimiento
   - Técnicas manuales
   - Técnicas semiautomáticas
   - Técnicas de adquisición a partir de un grupo de expertos
7. Adquisición automática de conocimiento
   - Aprendizaje inductivo
   - Aprendizaje por analogía
   - Aprendizaje por descubrimiento
8. Evaluación de los sistemas basados en el conocimiento
   - Sistemas de verificación. Tipos
   - Métodos de validación cualitativos y cuantitativos
   - Análisis de utilidad y usabilidad

Y los contenidos de los créditos teóricos de la asignatura IAICE se disponen de la forma siguiente:

1. Introducción a los SSBBC
   - El conocimiento en IA
   - Sistemas basados en conocimientos
   - Ingeniería del conocimiento
   - La metodología CommonKADS
2. Modelado del contexto en CommonKADS
   - Estudio de viabilidad: el modelo de la organización.
   - Análisis de impacto y mejora: el modelo de tareas y el modelo de agentes
3. La conceptualización del conocimiento en CommonKADS: Modelos de conocimientos y comunicación
   - El conocimiento del dominio
   - El conocimiento de inferencia
   - El conocimiento de la tarea. Plantillas de tareas.
4. Diseño e Implementación de SSBBC en CommonKADS
   - Identificación del conocimiento. Adquisición y elicitación del conocimiento.
   - Especificación del conocimiento
   - Validación y refinado del conocimiento
5. Formalismos de representación del conocimiento
   - Lógica de predicados
   - Sistemas de producciones
   - Redes semánticas
   - Marcos y guiones

Como se puede observar, ambas asignaturas siguen, en su parte fundamental, el modelo *CommonKADS* [9] para la construcción de SSBBC. La diferencia entre los temarios sólo comprende un pequeño porcentaje en los últimos temas. Mientras que en ICC se estudia con más detenimiento las técnicas de adquisición de conocimientos y los procesos de evaluación de SSBBC, en IAICE se describen los tipos de formalismos de representación de conocimientos. Esto es debido a que en la UEX, IAICE es la única asignatura de IA en la titulación de la Ingeniería en Informática, mientras que en la UDC existe una asignatura anterior anual de IA donde se estudia la representación de conocimiento.

2.4. Programas de prácticas

Creemos que el dominio de la disciplina incluye no sólo la comprensión de la materia básica, sino también la comprensión de la aplicabilidad de los conceptos a problemas del mundo real. Así, la aproximación ingenieril contemplada en el plano de clases teóricas de la asignatura se ve reforzada en el planteamiento de las prácticas, en las que además de realizar un pequeño proyecto de SSBBC completo, el alumno debe conocer entornos específicos de desarrollo de sistemas basados en el conocimiento. Así, el alumno deberá afrontar un problema real y ser capaz de construir un prototipo que lo resuelva, siguiendo los pasos de la metodología *CommonKADS* y utilizando una herramienta específica para el desarrollo de un SBC. Esta visión del programa práctico es coincidente en las dos asignaturas (ICC e IAICE) de esta ponencia.

Una vez elegido el dominio de aplicación, se realiza el análisis del sistema que, en la metodología *CommonKADS* equivale a elaborar

los modelos de contexto y los modelos conceptuales. Durante estas semanas en las que se realiza la labor de análisis, las clases prácticas se dedican también a enseñar a los alumnos de forma tutorada las características de la herramienta que utilizarán. El contenido de estas clases es el siguiente:

1. Edición y visualización de conocimiento
2. Métodos de razonamiento
3. Depuración del conocimiento
4. Implementación de un pequeño SBC a partir de un dominio juguete (sólo en ICC)
5. Integración con bases de datos y programas externos
6. Documentación del conocimiento
7. Diseño avanzado en la herramienta
8. Implementación del SBC elegido utilizando la herramienta (sólo en ICC)

Como se puede observar, en ICC los alumnos dedicarán ya las últimas semanas a la implementación del SBC elegido que seguirá las especificaciones establecidas en los modelos de análisis. En IAICE, en cambio, el objetivo final es la creación de la documentación que describe al SBC.

2.5. Método y Material Didáctico

Las clases de teoría se imparten siguiendo el método de clase magistral, utilizando diversos medios. No obstante, dada la orientación eminentemente práctica de la asignatura tiene especial importancia ilustrar frecuentemente con ejemplos los conceptos explicados. Por un lado, a lo largo de la exposición teórica se utilizarán ejemplos aislados para favorecer la comprensión de aspectos concretos. Por otro lado, y como apoyo importante a las prácticas, la explicación de la metodología de desarrollo *CommonKADS* se ilustrará además sobre un ejemplo completo de desarrollo de un SBC que se irá construyendo a lo largo del curso y del que se proporcionará a los alumnos las transparencias y las hojas de trabajo (usadas en *CommonKADS*) resultantes.

En cuanto a las prácticas de laboratorio, creemos que hacer que el alumno se enfrente a un problema e intentar resolverlo por sí mismo es la mejor y, en ocasiones, la única manera de asimilar los conceptos expuestos durante las clases teóricas.

En la **asignatura de IAICE**, se proporciona un enunciado sobre algún dominio adecuado para la aplicación de un SBC más o menos complejo. Se forman grupos de trabajo de hasta un total de 4 alumnos.

Las primeras semanas se dedican al aprendizaje de la herramienta Kads22 para la construcción de modelos de conocimientos. Se intenta que cuando se describan las correspondientes hojas de trabajo de *CommonKADS* en las clases de teoría, el alumno tenga ya los conocimientos suficientes en la herramienta para formar estos modelos.

Una vez dominado el entorno de las herramientas, en las sucesivas sesiones los alumnos confeccionan las hojas de trabajo que han interpretado desde las siguientes fuentes:

- El enunciado de la práctica dado en las primeras semanas, donde se explican los objetivos generales, las prioridades y se identifican las tareas y entidades principales.
- Documentos adicionales que se van proporcionado para complementar y detallar el sistema descrito en el enunciado. Estos documentos consistirán tanto en la descripción de los aspectos de la entidad, los modelos de trabajo, las políticas de la empresa, etc., como en documentos concretos sobre el dominio que se intenta implementar. Se facilitan estas especificaciones de forma que coincidan con el contenido explicado con anterioridad en las clases de teoría.
- La adquisición de conocimientos a partir de expertos y directivos. En este caso, se simula que el profesor es en algunos casos el director de la organización relacionada con el sistema y en otros el experto en la materia. Así, de forma periódica los alumnos someten al profesor en público a una batería de preguntas para delimitar cada uno de los aspectos conceptuales y procedimentales del SBC. Los alumnos deberán plasmar en las descripciones de sus modelos los conocimientos extraídos de estas "entrevistas".

En la **asignatura de ICC**, son los propios alumnos los que deben elegir el dominio de aplicación del SBC que deberán construir a lo largo del curso. El sistema debe resolver un problema real, que debe ser de los habituales en IC (clasificación, asesoramiento, diagnóstico, síntesis, planificación, monitorización, asignación,

diseño de configuraciones, ordenación cronológica, etc.) Con el fin de orientarles en la elección, durante la presentación de la asignatura, se le suministran a los alumnos una serie de temas genéricos y de aplicaciones típicas de los SSBBC, así como una serie de direcciones http donde podrán ver ejemplos de SSBBC reales. Los alumnos podrán elegir un tema de práctica directamente escogido de entre los presentados, o bien aportar ellos sus propias ideas al respecto. En cualquier caso, la elección deberá discutirse previamente con los profesores durante las tutorías con el fin de asegurar la viabilidad del proyecto y la disponibilidad de las fuentes de conocimiento. El objetivo de esta fase es que el alumno aprenda a distinguir el tipo de problemas para los que los SSBBC constituyen una aproximación efectiva.

Durante el curso, los alumnos deberán entregar escalonadamente los documentos relativos a las distintas fases del desarrollo de su sistema, según el calendario que se les suministra al principio de curso coordinado de acuerdo a la disposición temporal de los temas teóricos, y que consta de cuatro pasos:

1. Entrega del tema de la práctica, y de los componentes del grupo
2. Entrega de los modelos de organización, tareas y agentes. Los objetivos de esta segunda fase serán la constatación de la viabilidad del sistema propuesto, así como perfilar los objetivos del mismo
3. Entrega de los modelos de conocimiento y comunicación del sistema junto a su validación sobre un juego de ensayo, y
4. Enguirá de sistema ya implementado, que seguirá las especificaciones establecidas en los modelos anteriores, junto con una memoria final en la que se detallen todos aquellos aspectos importantes que no se hayan reflejado en los documentos anteriores: refinamiento de los modelos de conocimiento y comunicación, problemas y cuestiones relativas a la implementación, resultados de validación, etc.

La coordinación entre los temas teóricos y los laboratorios puede observarse en la figura 1 en la que se han encajado los temas en el periodo lectivo ideal de 15 semanas que componen un cuatrimestre.

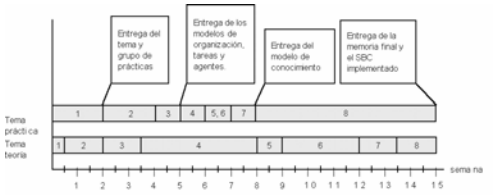


Figura 1. Planificación temporal propuesta para la asignatura ICC.

Como se puede observar en este cronograma, durante las primeras clases prácticas se imparten, de forma muy interactiva, diversos temas orientados a que el alumno se familiarice con la herramienta en la que deberá finalmente implementar el SBC propuesto (en el caso de la UDC, Nexpert.Object). Por esta razón, la impartición de estos temas se estructura de forma que finalice en el momento en que el alumno entrega el modelo de conocimiento y su proyecto está listo para comenzar la implementación.
Para la realización de los modelos de análisis se suministrará a los alumnos herramientas automáticas para el modelado de conocimiento como AION 8.1 o Kads22.

Las prácticas se realizarán por parejas. Consideramos que los grupos formados por dos personas son óptimos para estos objetivos ya que aporta el potencial educativo, formativo e informativo del trabajo en equipo pero intentando mantener un aprovechamiento amplio de los conocimientos de la asignatura que podría quedar diluido en un grupo más amplio.

Es necesario destacar en el modelo de planteamiento de ambas asignaturas, el importante papel que juegan las tutorías que los alumnos utilizarán para resolver dudas en la elaboración de los modelos y servirán al profesor para un seguimiento más cercano del trabajo de los alumnos.

2.6. Evaluación del alumno

Lo ideal es evaluar al alumno siguiendo un procedimiento continuo, basado además en una relación constante profesor-alumno durante el curso. En estas asignaturas, este tipo de procedimiento resulta, inicialmente, muy difícil, debido al número de alumnos matriculados (147 en IAICE y 102 en ICC) y al carácter cuatrimestral de la docencia de esta materia, que

reduce considerablemente el número de pruebas parciales que se pueden realizar, ya que al ser necesario hacerlo durante las horas de clase, se reduciría considerablemente el temario.

En **ICC** se realiza una evaluación continua, y prácticamente individualizada, con el esquema planteado para las prácticas. Además, del seguimiento semanal durante las horas asignadas al laboratorio, la entrega periódica de documentación previa a la implementación nos permitirá valorar la asimilación del alumno de los aspectos teóricos de la asignatura, además de subsanar los errores que se pudieran producir en su concepción.

La evaluación final del alumno se hará basándose en dos notas: la correspondiente a un examen escrito sobre los conocimientos teóricos de la asignatura, y la correspondiente a la revisión de las prácticas desarrolladas por los alumnos.

En **IAICE** sólo se pide al grupo la entrega de la documentación final descriptiva del SBC al final del primer cuatrimestre, pero se dedica parte de algunas sesiones prácticas para evaluar (sin calificación) el estado de las correspondientes hojas de trabajo descriptivas del sistema.

En esta asignatura, la calificación final del alumno se obtiene ponderando con un setenta por ciento el examen sobre los conocimientos teóricos y el treinta por ciento restante se obtiene desde las prácticas obligatorias de la asignatura.

En el apartado de prácticas para ambas asignaturas se valorarán: a) la entrega de las mismas en el momento y la forma adecuadas, b) la adecuación del material presentado a los requisitos de funcionamiento del mismo, c) el rigor seguido, tanto en la documentación como en la implementación software, y d) el grado de coordinación entre los miembros del grupo.

3. Ventajas e inconvenientes de la enseñanza de la IC orientada a una metodología

En la sección anterior hemos descrito las asignaturas ICC e IAICE que, como se puede observar, tienen un planteamiento común en cuanto a objetivos, contenido del temario y desarrollo de las prácticas. Pero esta visión no coincide con las de otras Universidades españolas, como se mencionó en la sección 1.

Como se ha señalado, muchos de los programas docentes en IC contienen temas de adquisición de conocimientos, representación de conocimientos, evaluación de SSBBC, algunos incorporan temas de tratamiento de incertidumbre, otros se orientan a la enseñanza de algún lenguaje de programación de IA y algunos temarios realizan un enfoque únicamente orientado a la lógica formal [7]. Se puede decir que estos temarios no siguen una metodología concreta de desarrollo de SSBBC, sino que presentan distintos temas relacionados con la IC, y procesos y técnicas relacionados con los SSBBC pero usando diferentes puntos de vista y terminologías.

Así, a diferencia de los enfoques anteriores, la propuesta que hacemos en esta ponencia pensamos que aporta los siguientes beneficios:

- Aunque el aprendizaje de una metodología conlleva un considerable esfuerzo y gasto de tiempo, un vocabulario y nomenclatura de representación homogéneo (como sigue *CommonKADS*) hace más fácil la comprensión de los distintas fases de construcción del SBC y las técnicas empleadas en cada proceso.
- Emplear una metodología completa para la confección de SSBBC hace que el alumno pueda percibir el ciclo de vida completo de desarrollo del sistema. El hecho de no centrarnos sólo en algunos aspectos del desarrollo (adquisición, evaluación, etc.) y la generación de documentación descriptiva del sistema facilita que los alumnos no caigan posteriormente en algunos de los errores que se produjeron históricamente en la disciplina.
- El acercamiento entre las metodologías de desarrollo de SSBBC y de sistemas software tradicionales es una tendencia que viene consolidándose desde hace varios años [6]. *CommonKADS* sigue esta filosofía y permite entender las ideas de desarrollo de sistemas que se estudian en otras asignaturas relacionadas con la Ingeniería del Software. Se debe indicar, además, que *CommonKADS* utiliza una extensión de los diagramas UML [2] para representar los conocimientos y tareas, por lo que consolidará el aprendizaje de esta notación si afortunadamente se sigue en otras asignaturas de la titulación.
- El empleo de una metodología que posee herramientas software de apoyo al desarrollo

facilita mucho la compresión del alumno, ya que los contenidos teóricos son probados correspondientemente en las sesiones prácticas. Además, se puede evaluar de forma continua los progresos y errores de los alumnos al aplicar estos conceptos.

- Además, creemos que estamos ofreciendo al alumno unos conocimientos que podrán utilizar directamente si logran trabajar en empresas relacionadas con la gestión del conocimiento o el desarrollo de SSBBC, ya que la metodología *CommonKADS* es una de las más empleadas hoy en día en el ámbito de la Comunidad Europea en estas materias.

Por otra parte, aunque pensamos que las ventajas apuntadas superan ampliamente a los inconvenientes, debemos indicar que nuestra propuesta tiene algunas deficiencias como son: la inexistencia de material bibliográfico en castellano o que la apuesta por una única metodología puede hacer que el alumno asocie la construcción de SSBBC exclusivamente con esta metodología concreta. Se debe indicar, además, que en los casos en los que no existan otras asignaturas relacionadas con la IA, se debe hacer un esfuerzo adicional en ajustar el temario para incluir también temas básicos de la IC, como los relacionados con los formalismos de representación del conocimiento, razonamiento con incertidumbre, etc.

4. Conclusiones

En este artículo se ha presentado y justificado un enfoque metodológico para la docencia de la Ingeniería del Conocimiento para la Ingeniería en Informática. Esta enseñanza orientada a una metodología coincide en las asignaturas de "Ingeniería del Conocimiento" de la Universidad de A Coruña y del primer cuatrimestre de la asignatura "Inteligencia Artificial e Ingeniería del Conocimiento" de la Universidad de Extremadura.

Podemos concluir que, frente a una enseñanza empleando varios puntos de vista de la Ingeniería del Conocimiento o del estudio de ésta usando modelos aislados, el empleo de una metodología (como *CommonKADS*) para estudiar los procesos y técnicas de desarrollo de SSBBC ofrece al alumno un enfoque integral del problema y le permite equiparar su desarrollo con procesos similares realizados en Ingeniería del Software. Esperamos que otros profesores que impartan (o vayan a hacerlo) asignaturas relacionadas con la Ingeniería del Conocimiento tengan en cuenta esta propuesta ya que, a nuestro juicio, los resultados que ofrece son muy satisfactorios.

Referencias

[1] ACM/IEEE-CS Joint Curriculum Task Force. Computing curricula 1991. IEEE Computer Society Press, 1991.

[2] Booch, G., Rumbaugh, J., Jacobson, I. *The Unified Modelling Language User Guide*. Addison-Wesley, 1998.

[3] CORPORATE The Joint Task Force on Computing Curricula. *Computing Curricula 2001*. Journal of Educational Resources in Computing, vol. 3, núm. 1, 2001.

[4] Gómez, A., Montes, C. *Enseñanza de Inteligencia Artificial e Ingeniería del Conocimiento*. Inteligencia Artificial, vol. 3, páginas 2-9, 1997.

[5] Guía de la Escuela Politécnica de Cáceres de la Universidad de Extremadura, 2002.

[6] Juristo, N., Pazos, J. *Towards a Joint Life Cycle for Software and Knowledge Engineering*. En: Cuena, J. (Ed.) *Knowledge Oriented Software Design*. Elsevier, 1993.

[7] Millán, E., Pérez de la Cruz, J.L. *Ingeniería del Conocimiento: Un enfoque formal.* CAEPIA, 2001.

[8] Palma, J.T., Paniagua, E., Martín, F., Marín, R. *Ingeniería del Conocimiento. De la extracción al modelado de Conocimiento*. Revista Iberoamericana de Inteligencia Artificial, vol. 11, pp. 46-72, 2000.

[9] Schreiber, G., Akkermans, H., Anjewierden, A., de Hoog, R., Shadbolt, N., van de Velde, W., Wielinga, B. *Knowledge engineering and management. The CommonKADS methodology.* MIT Press, 2000.

[10] Shaw, M.L.G., Gaines, B.R. *The synthesis of knowledge engineering and software engineering*. E: Loucopulos, P. (Ed.), *Advanced Information Systems Engineering*, LNCS, vol. 593, 1992.

[11] Urretavizcaya, Ma., Onaindía, E. *Docencia Universitaria de Inteligencia Artificial*. Inteligencia Artificial, vol. 17, pp. 23-30, 2000.

Conceptos básicos en la enseñanza de la Inteligencia Artificial: datos, información y conocimiento

Margaret Miró-Julià
Departament de Ciències Matemàtiques i Informàtica
Universitat de les Illes Balears
07122 Palma de Mallorca
margaret.miro@uib.es

Resumen

En este trabajo se presentan aspectos docentes de la Inteligencia Artificial en la Universidad de las Islas Baleares. La ponencia está dividida en tres partes.

En la primera se enmarca la problemática asociada a la docencia universitaria de la Inteligencia Artificial centrada en torno al qué debe enseñarse.

En la segunda parte de la ponencia se analiza el cómo debe enseñarse, haciendo hincapié en la inexistencia de una definición consensuada de lo qué es el conocimiento.

La tercera parte propone, basándose en lo anterior, una metodología docente para la asignatura *"Inteligencia Artificial"* de los estudios de Ingeniería Superior en Informática.

1. La asignatura "Inteligencia Artificial"

1.1. Objetivos y contenidos

La rápida evolución de las tecnologías obliga a una revisión continuada de los planes de estudio de Informática y al papel que en ellos debe desempeñar la Inteligencia Artificial. El ``Computing Curricula 2001" (CC2001) [1] es una reciente propuesta docente revisada conjuntamente por la *Association for Computing Machinery* (ACM) y el *Institute of Electrical and Electronics Engineers* (IEEE) que incorpora al curriculo los recientes cambios tecnológicos. En CC2001 se identifican diversas áreas de conocimiento que a su vez están divididos en temas.

La Inteligencia Artificial se consolida dentro del área de conocimiento *Sistemas Inteligentes*. El CC2001 señala la importancia de los conocimientos en el área de *Sistemas Inteligentes*: "Un sistema inteligente tiene que percibir su entorno, actuar racionalmente para completar sus tareas e interactuar con otros sistemas y con los seres humanos." Estas capacidades se consiguen mediante visión por ordenador, lenguaje natural, robótica, ….. Todos estos temas dependen de un conjunto amplio de representaciones del conocimiento (general y especializado), de mecanismos de razonamiento, de algoritmos de búsqueda y de técnicas de aprendizaje.

La Inteligencia Artificial proporciona una serie de herramientas para la resolución de problemas que resultan difíciles o impracticables de resolver por otros métodos.

El informe CC2001 establece materias esenciales y materias optativas dentro del área de conocimiento de *Sistemas Inteligentes*:

1. Materias esenciales:
- Conceptos básicos en Sistemas Inteligentes
- Búsqueda y satisfacción de restricciones
- Representación del Conocimiento y Razonamiento

2. Materias optativas
- Búsqueda avanzada
- Representación del Conocimiento y Razonamiento avanzados
- Agentes
- Lenguaje natural
- Aprendizaje y redes neuronales
- Planificación
- Robótica

En España, en el marco de las III Jornadas sobre la Enseñanza Universitaria de Informática

(JENUI'97), se celebró el primer panel sobre *Enseñanza de Inteligencia Artificial e Ingeniería del Conocimiento*. Uno de los objetivos de este panel consistió en construir un programa de referencia consensuado entre todos los asistentes para la asignatura ``ideal'' de Inteligencia Artificial [2]. Los asistentes, por unanimidad, acordaron que el núcleo de la asignatura debería estar formado por dos temas fundamentales: búsqueda y representación de conocimientos. Surgieron discrepancias en cuanto a la extensión de los contenidos, el orden en que aparecían en el programa y el resto de contenidos. Se consensuaron unos contenidos mínimos:

1. Introducción a la IA
   1.1. Definiciones de IA
   1.2. Revisión histórica
   1.3. Breve panorámica de áreas de aplicación
2. Caracterización de problemas en espacios de estados
3. Búsqueda
   3.1. Caracterización
   3.2. Búsqueda no informada
   3.3. Búsqueda informada:
      3.3.1. métodos genéricos de resolución
      3.3.2. búsqueda sin adversarios
      3.3.3. búsqueda con adversarios
4. Representación de Conocimientos
   4.1. Caracterización
   4.2. Lógica
   4.3. Sistemas de Producción
   4.4. Representaciones Taxonómicas
      4.4.1. redes semánticas
      4.4.2. marcos

Más recientemente, en el marco de la Conferencia de la Asociación Española para la Inteligencia Artificial (CAEPIA-TTIA'2001), se celebró el encuentro de *Docencia de la Inteligencia Artificial*. Muchas de las ponencias allí presentadas ajustan sus programas a los contenidos mínimos arriba mencionados. Resaltamos los programas propuestos por Mata [9], Moreno Pérez et al. [10] o Mandow et al. [4],

Por otro lado, se aseguró la necesidad de que la docencia impartida se ajuste a los requerimientos y evolución de los problemas que la Inteligencia Artificial deberá abordar para consolidarse como una tecnología útil a la sociedad.

Desde el punto de vista curricular es necesario actualizar los contenidos teniendo en cuenta tanto el marco general de la disciplina como el entorno interno de cada centro docente.

1.2. La Inteligencia Artificial en la UIB

La docencia de la Inteligencia Artificial en la Universidad de las Islas Baleares (UIB) se localiza en los estudios de Informática que comprende las titulaciones de *Ingeniería Técnica en Informática de Gestión* (ITIG), *Ingeniería Técnica en Informática de Sistemas* (ITIS) e *Ingeniería Superior en Informática* (ISI). Ambas ingenierías técnicas se organizan en tres cursos académicos y tienen acceso directo a la titulación de segundo ciclo de Ingeniería Superior en Informática, organizada en dos años.

Actualmente, la UIB ofrece dos asignaturas de Inteligencia Artificial: *"Introducción a la Inteligencia Artificial"* de 6 créditos que es asignatura optativa en ITIG y asignatura de libre configuración en ITIS y la asignatura troncal de 9 créditos *"Inteligencia Artificial"* de la ISI.

El trabajo realizado por CC2001, JENUI '97 y CAEPIA-TTIA'2001 facilita bastante la elección de contenidos para la elaboración del programa de las asignaturas de Inteligencia Artificial.

El objetivo central de la asignatura *"Introducción a la Inteligencia Artificial"* es capacitar al alumno para reconocer problemas susceptibles para ser abordados con la metodología de la Inteligencia Artificial y a partir de las nociones básicas dadas elegir la más adecuada.

La asignatura troncal *"Inteligencia Artificial"* es la más importante de las asignaturas de Inteligencia Artificial que se imparte actualmente en la UIB. De acuerdo con las directrices mencionadas arriba, su programa:

- incorpora una introducción que incluye una perspectiva histórica, definición y ejemplos de aplicaciones relevantes;
- imparte los fundamentos de la búsqueda analizando las técnicas heurísticas fundamentales;
- estudia la representación del conocimiento y la construcción de sistemas basados en el conocimiento.

El programa actual de la asignatura consta de 7 temas, con contenidos que se ajustan a los criterios arriba mencionados:

1.- Introducción
2.- Resolución de Problemas en Inteligencia Artificial.
3.- Búsqueda en el Espacio de Estados.
4.- Representación del Conocimiento
5.- Razonamiento con Incertidumbre.
6.- Sistemas basados en el Conocimiento.
7.- Planificación.

2. Datos, información o conocimiento

Una vez establecido el qué debe enseñarse, hay que considerar el cómo. Según Rich [6], "Uno de los más sólidos y rápidos resultados que surgieron en las primeras tres décadas de las investigaciones en Inteligencia Artificial fue que la inteligencia necesita conocimiento". Todos coincidimos en la necesidad del conocimiento para desarrollar técnicas, incluso conocemos sus propiedades. Todos pensamos que sabemos lo que es el conocimiento. Las diferentes formas de vida (humana o de otro tipo) no podrían funcionar si no estuvieran recibiendo y procesando conocimiento constantemente, siempre ha sido así. Aunque el conocimiento sea esencial para el funcionamiento de cualquier organismo, desconcierta saber que no exista una definición consensuada del concepto de conocimiento aunque, esto sí, exista una confusión generalizada al respecto.

Parte de esta confusión radica en el hecho de que conocimiento no es **una** cosa, conocimiento son **muchas** cosas. Aunque nos referimos a conocimiento utilizando un nombre común individual, realmente debería ser interpretado como un nombre **colectivo**.

La noción de conocimiento puede ser explorado desde una variedad de perspectivas. Desde la perspectiva de la lógica matemática donde el conocimiento queda claramente estructurado mediante una descripción precisa y concisa: la lógica de primer orden, hasta un punto de vista psicológico donde el conocimiento es un fenómeno empírico del mundo natural, pasando por un punto de vista social donde el conocimiento es el producto de la búsqueda a ciegas que es la evolución. El resultado de todas estas observaciones es que el conocimiento son muchas y variadas cosas que de momento no tienen una representación general común.

La representación del conocimiento es un bloque importantísimo dentro de la docencia de la Inteligencia Artificial. Los libros más recomendados en las bibliografías de las diversas asignaturas de Inteligencia Artificial en las universidades españolas dedican varios capítulos al tema. Pero, resulta interesante y sorprendente advertir que los autores no definen el conocimiento, suponen que todos sabemos lo que es. Ya lo dice claramente Nilsson en su libro [5], "El gran problema de la Inteligencia Artificial es qué expresar, no como expresarlo", sé representar el conocimiento aún no sabiendo lo qué es. La definición de lo qué es el conocimiento no preocupa, tan sólo interesa representar el conocimiento para poder resolver el problema concreto que se está estudiando.

Según Newell [7], "la gente … induce a partir de su práctica una cierta noción de lo que significa que un sistema tenga conocimiento y que significa proporcionar los mecanismos del nivel simbólico que codifican dicho conocimiento y lo extraen para que el programa se comporte del modo deseado". Sin embargo, ¿podemos basar la enseñanza de la Inteligencia Artificial en la esperanza de que los estudiantes induzcan por si mismos la noción adecuada de conocimiento? ¿Sería necesario abordar esta cuestión de forma explícita?

2.1. El conocimiento: ente abstracto

En la asignatura *"Introducción a la Inteligencia Artificial"*, la inclusión del concepto de agente esconde la necesidad de definir el concepto de conocimiento.

El concepto agente se refiere a cualquier sistema que se desenvuelve en un determinado medio en el que puede realizar acciones y del cuál puede recibir percepciones. Además, se supone que un agente posee una serie de objetivos y que intenta seleccionar sus acciones de modo que pueda conseguirlos. La introducción del término de agente da una definición operativa al concepto de conocimiento, ya que, una característica importante de un agente es que posee conocimiento.

El conocimiento de un agente se define como aquello que le hace competente para la

consecución de sus objetivos, es lo que le permite seleccionar las acciones adecuadas. El conocimiento es algo abstracto que nosotros, como observadores, atribuimos a un agente para explicar su comportamiento sin necesidad de entrar en los detalles de su estructura física.

La introducción del agente parece oportuno en un curso introductorio de Inteligencia Artificial. Sin embargo, en la asignatura *"Inteligencia Artificial"* de la ISI en dónde debe profundizarse más, es conveniente introducir el concepto de conocimiento a un nivel adecuado a un segundo ciclo universitario.

2.2. El conocimiento: ente concreto

Para conseguir una definición concreta del conocimiento se empieza intentando aislar el conocimiento más básico, el más rudimentario, el más puntual. Desde un punto de vista informático el dato es un conocimiento puntual en cuanto a su contenido.

Los datos son observaciones obtenidas a partir de investigaciones, los datos hacen referencia a valores que pueden ser cuantitativos (numéricos) o cualitativos (cualidades). Generalmente, un único dato no tiene mucho sentido, a no ser que se incluya dentro de un contexto. El contexto es necesario para darle un significado a los datos, incluir un contexto equivale a realizar afirmaciones acerca de un objeto (elemento) y sus propiedades o atributos.

A partir de las relaciones entre los datos y de los patrones encontrados se consigue información. La información consiste en valores (datos) que se poseen de ciertos objetos. Esta información tiene un significado para la persona que lo analiza y depende del contexto utilizado.

Las palabras dato e información son de uso cotidiano y en muchas ocasiones se utilizan indistintamente. Es más, en el lenguaje coloquial, dato, información y conocimiento son sinónimos. Si se consulta un diccionario se aprecia que los términos dato, información y conocimiento están definidos unos en términos de los otros. Sin embargo, todos somos conscientes de unos pequeños matices que los diferencian. Todos coincidimos en que conocimiento es algo más que información. Al intentar comprender mejor la información se obtiene conocimiento.

La información consiste básicamente en datos almacenados, es un conocimiento estático. Pero el conocimiento tiene además una componente funcional: la interpretación de los datos. Esta interpretación está íntimamente ligada con el uso que se hace de los datos. Es necesaria una componente dinámica, unos procesos que manipulen e interpreten los datos con el fin de transformarla en conocimiento dinámico. Según Freksa [3], conocimiento es la interacción entre los datos y la interpretación:

Conocimiento = Datos + Interpretación

donde *datos* refleja el aspecto estático e *interpretación* el aspecto dinámico.

3. Método propuesto

En la enseñanza universitaria está presente el eterno dilema entre la educación y la profesionalidad; entre el saber y el hacer. En las enseñanzas de carácter tecnológico surge el peligro del predominio de la concepción del saber como mero almacenamiento de definiciones y teoremas o como familiaridad con las técnicas usuales frente a la verdadera dimensión formativa que debe presidir la educación superior.

La rápida evolución de las técnicas y conocimientos aplicados y la adaptación a las necesidades sociales cambiantes inciden directamente en la docencia de la Inteligencia Artificial.

En la enseñanza de la Inteligencia Artificial en la UIB no basta con conseguir que el alumno asimile un cúmulo de conocimientos determinados, tanto o más importante es inculcarle la aptitud y la preparación necesarias para completar, por sí sólo, su formación informática en las líneas y temas que le resulten indispensables. Hay que dar más cabida a la valoración de la capacidad creativa que a los conocimientos de herramientas específicas.

Es necesario formar al futuro profesional informático, no basta con sólo enseñarle algoritmos y métodos específicos, hay que exponer, desarrollar e interpretar el meta-problema que subyace debajo de los problemas estudiados mediante métodos y técnicas propios de la Inteligencia Artificial. Además de enseñar al alumno las distintas representaciones del conocimiento según el problema a estudiar, es necesario inculcarle una sensibilidad e inquietud

acerca de lo que es el conocimiento., y ofrecerle una primera aproximación de la representación del conocimiento, una representación válida para cualquier tipo de conocimiento.

El método propuesto no modifica el contenido, el "qué", del programa de la asignatura, tan sólo retoca la metodología, el "cómo", mediante la introducción, de manera pausada, de la definición del conocimiento.

Existen dos temas del programa en los cuales es necesario tratar el conocimiento y que permiten la introducción de la noción de conocimiento de forma paulatina: Búsqueda en el Espacio de Estados (conocimiento estático) y Representación del Conocimiento (conocimiento dinámico).

3.1. Búsqueda en el Espacio de Estados

Los primeros trabajos desarrollados en el campo de la inteligencia artificial (1955-1965) abordan problemas que eran simplificaciones del mundo real: demostración de teoremas, problemas de juegos, etc. El primer formalismo de representación fue el espacio de estados que representa la estructura de un problema en término de las alternativas disponibles.

La metodología utilizada consistía en la realización de procesos de búsqueda en espacios de estados, desde un estado inicial a un estado solución, sin utilizar ningún conocimiento sobre el dominio del problema. La búsqueda a ciegas del dominio pretende realizar una exploración exhaustiva del espacio de estados de forma progresiva y sistemática que, en principio, no deje ningún estado sin explorar.

Este método no resulta eficaz debido a la explosión combinatoria. Información adicional sobre el dominio puede ayudar a dirigir el proceso de búsqueda, de tal manera que son explorados en primer lugar aquellos estados que resultan más prometedores a la hora de conducir a un estado solución.

La principal diferencia entre la búsqueda informada y la búsqueda a ciegas es que a cada estado se le asocia un valor de lo cerca que se encuentra de un estado solución. Existe información en forma de una tabla de datos dentro de un contexto, existe un primer nivel de conocimiento que ayuda a guiar el proceso de búsqueda. Al disponer de esta información adicional, la búsqueda deja de ser rígida, resultando más inteligente. El desarrollo de la búsqueda heurística permite la introducción, de manera simple, de un primer nivel de conocimiento estático, que de alguna manera indica una estrategia para la resolución del problema.

Un problema que resulta ilustrativo de la búsqueda heurística y el primer nivel de conocimiento es el típico problema que consiste en varias ciudades comunicadas por un servicio de autobuses, se pretende ir de la ciudad A a la ciudad B. Este problema puede resolverse utilizando las distintas estrategias de búsqueda no informada, pero si se dispone de una tabla que proporciona las distancias entre cada ciudad y la ciudad B, si se dispone de conocimiento llegamos más fácilmente al estado solución.

3.2. Representación del Conocimiento

En la década de los setenta y ochenta se empezaron a aplicar técnicas propias de la Inteligencia Artificial en la resolución de problemas complejos del mundo real. Una importante conclusión a la que se llegó fue la importancia de saber modelar la gran cantidad de conocimiento específico del dominio de cada problema. Hay que diseñar estructuras capaces de reproducir, de manera útil, ciertos aspectos de la realidad.

El conocimiento de primer nivel introducido en la búsqueda heurística consiste básicamente en datos almacenado. Es necesaria una componente dinámica, unos procesos que manipulen e interpreten los datos con el fin de transformarla en conocimiento dinámico.

Lo que realmente distingue la información del conocimiento es el carácter objetivo de la información frente al carácter subjetivo del conocimiento. La información, por sí misma, es inerte. Es el usuario al utilizarla, quien la transforma en algo activo: en conocimiento.

Los datos junto con su interpretación forman la base de conocimientos sobre la cual se aplican diferentes tipos de razonamientos para extraer más información. Esta representación de los datos suele realizarse mediante proposiciones válidas representadas a través de la lógica o reglas dando lugar a estructuras declarativas o de procedimientos.

Pero, existe otro tipo de conocimiento, el generado a partir de una colección de datos pertenecientes a un mismo contexto. Este prototipo de conocimiento considera un conjunto de objetos y un conjunto de atributos, los datos corresponden a los valores de los atributos que corresponden a los diversos objetos.

Dado un conjunto de objetos $D = \{d_1, d_2, ..., d_m\}$ y un conjunto de atributos $R = \{r_1, r_2, \ldots, r_n\}$, definimos la tabla de objetos y atributos (OAT) como aquella tabla cuyas filas representan los objetos y cuyas columnas representan los atributos. Cada elemento t_{ij} de la tabla representa el valor del atributo r_j que corresponde al objeto d_i tal como indica la Tabla 1.

	$\mathbf{r_1}$	$\mathbf{r_2}$	…	$\mathbf{r_j}$	…	$\mathbf{r_n}$
$\mathbf{d_1}$	t_{11}	t_{12}	. . .	t_{1ij}	.....	t_{1n}
$\mathbf{d_2}$	t_{21}	t_{22}	. . .	t_{2ji}	. . .	t_{2n}
	.....					
$\mathbf{d_i}$	t_{i1}	t_{i2}	. . .	t_{ij}	. . .	t_{in}
	.....					
$\mathbf{d_m}$	t_{m1}	t_{m2}	. . .	t_{mi}	. . .	t_{mn}

Tabla 1. Tabla de objetos y atributos

La OAT es una correspondencia C entre el conjunto D de objetos y el conjunto R de atributos. Una OAT es la descripción del conocimiento que se posee sobre los objetos dados a partir de los atributos dados.

La información (los datos) presentada y organizada apropiadamente origina conocimiento, es decir, genera un conjunto de nociones o ideas, (valores de atributos) sobre un subconjunto de objetos D_i. Llamamos conocimiento a toda pareja de la forma $< D_i, C(D_i) = R_i >$. En general, todo subconjunto D_i genera conocimiento.

Previamente se ha expuesto el carácter colectivo del conocimiento. Esta idea queda claramente reflejada al considerar una OAT, ya que se generan múltiples conocimientos del tipo $< D_i, R_i >$.

Este tipo de conocimiento definido a partir de una pareja de subconjuntos $< D_i, R_i >$ puede ser estudiado de manera intuitiva utilizando el álgebra de Boole (teoría de conjuntos)., ya que los subconjuntos D_i y R_i son elementos del conjunto partes de D y partes de R respectivamente.

Por desgracia, el conocimiento es más complejo que las simples ideas expuestas aquí. Según J. Miró [8], quien ha profundizado en el tema, existen dos tipos de conocimiento:

- el conocimiento extensional o de modo *R*, que corresponde a un modelo detallado de conocimiento donde se enumera a todos los objetos que posean los valores considerados de los atributos;
- el conocimiento declarativo o de modo *L* que corresponde a un modelo genérico de conocimiento donde se realizan afirmaciones de un subconjunto de objetos, sin enumerarlos, utilizando tan sólo los valores de sus atributos.

En otras palabras, a partir de una OAT se pueden realizar afirmaciones detalladas enumerando los objetos, o bien pueden realizarse afirmaciones genéricas, sin nombrar a los objetos, a partir de los valores de los atributos. Por ejemplo, las matemáticas utilizan en sus definiciones conocimiento modo *L*, se realizan afirmaciones sobre el conjunto de los números reales sin especificar que elementos la componen.

Debe quedar claro que el mismo conocimiento puede ser descrito en modo *R* o modo *L*. La generación del conocimiento en modo *L* a partir del correspondiente conocimiento en modo *R* equivale a un proceso de inducción. La obtención del conocimiento en modo *R* a partir del mismo conocimiento en modo *L* equivale a un proceso de instanciación (ejemplo). La adquisición de conocimiento en modo *L* a partir de conocimiento modo *L* constituye un proceso de deducción.

4. Conclusión

La metodología y los objetivos del programa de la asignatura "Inteligencia Artificial" de los estudios de ISI debe reflejar la misión del profesor: formar a futuros profesionales informáticos. Esta formación consiste en proporcionar a los alumnos conocimientos específicos y, sobre todo, en inculcarles una visión conjunta de la complejidad presente en el planteamiento de problemas y de la simplicidad de su resolución mediante métodos y técnicas de la Inteligencia Artificial.

Los métodos propios de la Inteligencia Artificial mejoran su eficiencia si se utiliza el conocimiento disponible. La introducción formal del concepto de conocimiento permite introducir, de manera natural, la complejidad (no lo sé definir) y la simplicidad (sé lo que es) presente en la Inteligencia Artificial.

La metodología propuesta introduce el conocimiento de primer nivel o conocimiento estático como aquella información adicional necesaria para realizar una búsqueda heurística y concluye definiendo el conocimiento como una correspondencia entre el conjunto de objetos D y el conjunto de atributos R.

Los métodos de búsqueda sin información son ineficaces y no tienen aplicación práctica dentro de la Inteligencia Artificial. Los métodos heurísticos sacrifican la completitud al incrementar la eficiencia y son preferibles a la hora de resolver un problema. Una heurística es una estrategia basada en una suposición bien fundamentada que puede resolver un problema dado, pero que no ofrece ninguna garantía. Las heurísticas son estrategias que necesitan información adicional para poder realizar la suposición. Es necesario conocimiento en forma de dato (valor numérico en un contexto) que evalúe de algún modo lo prometedor que resulta el nodo. De una manera natural, se introduce el conocimiento estático. Este primer nivel de conocimiento permite explorar en primer lugar los caminos más prometedores, acotando el espacio de búsqueda.

Para poder realizar una representación coherente del conocimiento se incluye una presentación formal de conocimiento. El conocimiento dinámico formado por los datos almacenados y los procesos que manipulan e interpretan los datos con el fin de transformarla en conocimiento dinámico. Los datos junto con su interpretación forman la base de conocimientos sobre la cual se aplican diferentes tipos de razonamientos para extraer más información. Esta representación de los datos suele realizarse mediante proposiciones válidas representadas a través de la lógica o reglas dando lugar a estructuras declarativas o de procedimientos. Estas estructuras constituyen el núcleo clásico de la representación del conocimiento.

Sin embargo, existe otro tipo de conocimiento, el generado a partir de una tabla de objetos y atributos. Este tipo de conocimiento se genera a partir de una correspondencia entre el conjunto D de objetos y el conjunto R de atributos y permite la representación del conocimiento utilizando parejas de subconjuntos de D y de R de la forma $< D_i, R_i >$.

Como reflexión final, es conveniente reconocer que el método propuesto admite muchas mejoras que esperamos identificar e incluir oportunamente en un futuro.

Agradecimientos

Deseo expresar mi agradecimiento a mis compañeros del grupo de investigación "Modelos formales en Inteligencia Artificial" que directa o indirectamente me han ayudado a preparar esta ponencia que pretende reflejar nuestras inquietudes en la docencia de la Inteligencia Artificial.

Quiero agradecer al Dr. Gabriel Fiol sus valiosos comentarios sobre lo "qué" es la Inteligencia Artificial y sus sugerencias y el continuo intercambio de ideas sobre "cómo" debe enseñarse la Inteligencia Artificial.

Finalmente, quiero expresar mi respeto, admiración y agradecimiento al Dr. José Miró por sus largas discusiones sobre los pilares de la Inteligencia Artificial; su disponibilidad a la hora de implementar esta metodología; y sobre todo por sus ideas, perspicacias e intuiciones.

Referencias

[1] IEEE/ACM Computer Science Joint Task Force, *Computing Curricula 2001*, Communications of the ACM, 2001. http://www.amc.org/sigcse/cc2001

[2] A. Gómez-Pérez y C. Montes. Enseñanza de Inteligencia Artificial e Ingeniería del Conocimiento. *Inteligencia Artificial*, 3: 2-9, 1997. http://aepia.dsic.upv.es/numeros/3/gomez

[3] Ch. Freksa, U. Furbach y G. Dirlich, "Cognition and Representation", ATP-34-X-84. Technische Universität München, 1984.

[4] L. Mandow y J. L. Pérez de la Cruz, *"Qué" y "cómo" enseñar en Inteligencia Artificial.* En *CAEPIA-TTIA'2001, Conferencia de la*

Asociación Española para la IA, pp. 1-10, Gijón, 2001.
http://ftp.aic.uniovi.es/Docencia_IA/Ponencias

[5] N. Nilsson, *Inteligencia Artificial. Una nueva síntesis*, McGraw Hill, 2000.

[6] E. Rich y K. Knight, Inteligencia Artificial, McGraw Hill, 1994.

[7] A. Newell, *Reflections on the knowledge level*, Artificial Intelligence. 59: 31-38, 1983.

[8] J. Miró, *Gathering Knowledge,. Letters to a Friend* . Ed. UIB, 1992.

[9] E. J. Mata, *Aspectos docentes de la asignatura ``Inteligencia Artificial'' del plan de estudios de la licenciatura de Matemáticas en la Universidad de La Rioja.* En *CAEPIA-TTIA'2001, Conferencia de la Asociación Española para la IA*, pp. 139-146, Gijón, 2001.
http://ftp.aic.uniovi.es/Docencia_IA/Ponencias

[10] J. A. Moreno Pérez, J. M. Moreno Vega, P. Caballero, P. García, F. Pérez y E. Sánchez, *Docencia de IA en la Universidad de La Laguna.* En *CAEPIA-TTIA'2001, Conferencia de la Asociación Española para la IA*, pp. 65-74, Gijón, 2001.
http://ftp.aic.uniovi.es/Docencia_IA/Ponencias

Métodos pedagógicos innovadores

Debate y foro en el aula como metodología docente: estudio comparativo de su aplicación en la asignatura Sistemas de Transmisión de Datos

José L. Poza, Alberto Bonastre, José Oliver
Dpto. de Informática de Sistemas y Computadoras
Escuela Técnica Superior de Informática Aplicada
Universidad Politécnica de Valencia. 46022 Valencia
e-mail: {jopolu, bonastre, joliver }@disca.upv.es

Resumen

La docencia universitaria está obligada a dar una doble atención a los sistemas pedagógicos. Una primera aplicación está en el trabajo de dar clase y la otra aplicación está en la investigación de nuevas técnicas docentes.

En este trabajo se presenta la comparación de la experiencia en la utilización de dos métodos docentes innovadores en el aula, en concreto el debate y el foro.

El uso de estos métodos se ha probado en la asignatura Sistemas de Transmisión de Datos, dentro del programa EUROPA (Una Enseñanza ORientada al APrendizaje) [1] de la Universidad Politécnica de Valencia.

Ambos métodos se emplearon para fomentar hábitos de trabajo en grupo y de capacidad de comunicación. El aula fue el escenario donde se desarrollaron estas actividades. Los temas de las actividades fueron la banda ancha para el debate y las nuevas generaciones de móviles por lo que respecta al foro.

En el presente trabajo se ofrece la estructuración y materiales empleados para ambas actividades y la comparación de los resultados de las opiniones de los alumnos obtenidas a partir de unas encuestas de opinión.

1. Introducción

La asignatura de Sistemas de Transmisión de Datos [2] se centra en los primeros niveles del modelo ISO/OSI (nivel físico y enlace de datos), lo que tradicionalmente se ha venido en llamar "data communication" [3] profundizando en las técnicas de transmisión existentes y describiendo diferentes opciones de implementación. Es una asignatura optativa, correspondiente a los planes de estudios de 1.996, para la obtención de los títulos de Ingeniero en Informática (5º curso, con 3 créditos de teoría y 3 créditos de prácticas) e Ingeniero Técnico en Informática de Sistemas y de Gestión (3º curso, con 3 créditos de teoría y 1.5 créditos de prácticas).

El proyecto EUROPA nace como sucesor del proyecto PIE (Proyecto de Innovación Educativa) de la Universidad Politécnica de Valencia. El proyecto EUROPA trata de lograr el cuádruple objetivo de que cada alumno "aprenda a conocer y saber, aprenda a aprender, aprenda a ser y aprenda a convivir con otros" [1]

Debido a que la asignatura es final en las titulaciones en las que se imparte, se hace necesario fomentar los hábitos de conducta necesarios en el desarrollo de la carrera profesional de los alumnos.

Estos hábitos son difíciles de lograr por medio de una metodología docente basada en la clase magistral, en la realización de problemas técnicos y un poco menos en la realización de prácticas de laboratorio. Por ello, se decidió emplear técnicas de trabajo en grupo y de comunicación. En concreto, se escogió el debate para mejorar la capacidad de argumentación y de crítica, mientras que el foro se eligió para mejorar la capacidad de intercambio de comunicación y síntesis. Ambas técnicas se conocen, desde el punto de vista del programa EUROPA, como seminarios. No fueron los únicos que se realizaron durante el curso, ya que también se efectuaron dos estudios de caso similares a los realizados en cursos anteriores, ya comentados en pasadas ponencias [4].

La experiencia se ha realizado con los alumnos de 5º curso de la titulación de Ingeniero en Informática. El grupo se componía de 21 alumnos, de los cuales se constató la asistencia regular de 16 de ellos.

Sobre la documentación necesaria para realizar las sesiones, se dividió en dos tipos, previa y de seminario. La documentación previa estuvo disponible una semana antes de la realización de las sesiones. Esta documentación se insertó en la página Web de la asignatura [5] y sigue accesible. La documentación necesaria para el desarrollo del seminario, tanto del debate como del foro, se entregó en la misma sesión. Se planteó la posibilidad de proporcionarla previamente, pero el hecho de que estudiaran sólo aquellos aspectos necesarios para el desarrollo del seminario, hizo que se desestimara esta posibilidad.

La estructura del artículo es la siguiente: en el apartado 2 sobre la metodología se explican los conceptos teóricos aplicados en la experiencia. El desarrollo y materiales empleados tanto en el debate como en el foro se explican detalladamente en el apartado 3. Los resultados se analizan en el apartado 4. Finalmente el apartado 5 recoge las impresiones y reflexiones de los profesores sobre ambas experiencias.

2. Metodología

Como se ha comentado, en la asignatura se fomentan determinados hábitos y actitudes que son importantes en la labor profesional que los alumnos van a desarrollar [6]. Principalmente se centraron los esfuerzos en promover aquellos hábitos que fomentaran el trabajo en grupo y el pensamiento crítico. Se pueden apreciar en la figura .1

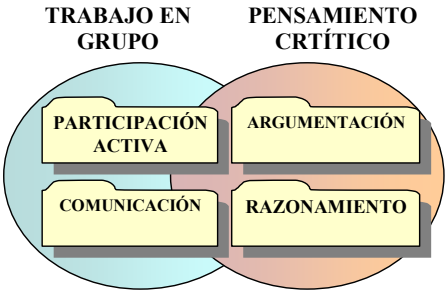


Figura 1. Hábitos de trabajo recomendables.

Éstos hábitos y actitudes se pueden promover a través de varios de los métodos docentes que emplean dinámicas de grupo [6]. Como primer paso en la aplicación de estas metodologías docentes resulta indispensable una preparación previa del alumnado en grupo, lo cual redunda en una estimulación en la capacidad de análisis de la documentación entregada y de la conveniencia del trabajo en grupo.

Las metodologías seleccionadas, adecuadas tanto a la materia como a las condiciones existentes (nº de alumnos, posibilidad de realización en el aula de teoría, etc.) fueron el debate y el foro [7].

Habitualmente se entiende como debate la controversia o discusión sobre una cosa entre dos o más personas, en este caso se pretendió agregar el trabajo en grupo tanto en la preparación del debate como en la actuación en el mismo. Foro se entiende como una reunión para discutir asuntos de interés actual ante un auditorio que a veces interviene en la audición, en este caso se secuenció el foro, primero una reunión de grupo y seguidamente la exposición al público para hacer una puesta en común.

2.1. Aplicación como método docente

Una de las diferencias fundamentales entre un debate y un foro, es que en el debate se tiene una opinión prefijada, mientras que el foro sirve para obtener una opinión, por lo que no se parte de una postura determinada.

La otra diferencia es la ya expuesta en las figuras 2 y 3, donde se puede apreciar las diferencias de contenidos a tratar por parte de los grupos en el caso del debate, y la coincidencia de los mismos en el caso del foro.

2.2. Debate

El primer método probado fue el debate. Se dividió a los alumnos en dos grupos, uno de los grupos debía dar a conocer la tecnología ADSL y el otro la del cable óptico. Esta división se hizo por azar y en el inicio de la sesión, de forma que un alumno no conociese su postura previamente. Esto último es necesario, ya que de conocer la postura previamente, se podría dar el caso de que el alumno sólo prepare el tema que le toque defender.

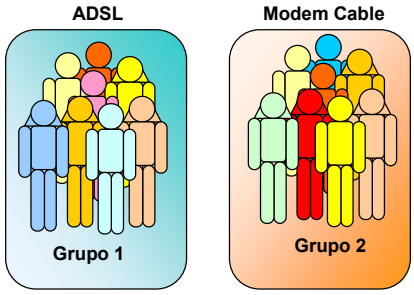


Figura 2. Esquema de organización del debate.

Como se puede apreciar en la figura 2, en el debate, los grupos no comparten la temática, lo que implica que no hay un punto en común, y consecuentemente no existen objetivos compartidos.

2.3. Foro

El segundo de los métodos que se planteó emplear fue el foro. Dentro del ambiente de la informática, se entiende los foros como una secuencia de comentarios que se van respondiendo, esto se podría entender como una mezcla del debate y el foro, sin embargo esto no es lo que se pretendió con la experiencia realizada.

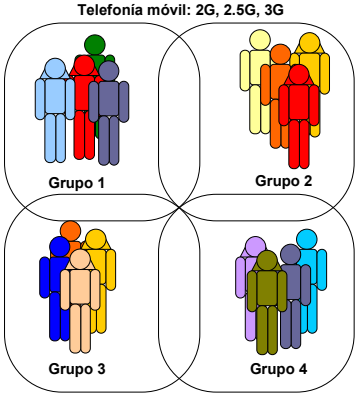


Figura 3. Esquema de organización del foro.

Como se puede apreciar en la figura 3. En el foro todos los grupos comparten la misma temática. Esto hace que tengan un objetivo común y compartido, lo que facilita la comunicación y la participación.

3. Experiencia: Debate y Foro

Para la secuenciación de ambas metodologías, se procuró seguir un esquema lo más parecido posible. Ya que el uso de métodos innovadores es de por sí confuso para los alumnos, se debe procurar homogeneizar lo más posible su estructura. Esta estructura tenía tres partes organizadas temporalmente:

1. Preparación.
2. Desarrollo.
3. Evaluación.

Esta secuenciación se puede ver con mucho más detalle en la figura 6 junto al papel que desempeñan en cada fase tanto el profesor como el alumno.

El tema del debate fue “ADSL vs Cable”, el hecho de que los medios de transmisión por banda ancha tuviesen dos alternativas, hizo que este tema fuese elegido para el debate.

La documentación sobre el tema es variada, y las comparaciones que se hacen de los sistemas son fáciles de localizar en Internet, estas referencias se encuentran en la documentación correspondiente [5].

El tema del foro fue el futuro de las comunicaciones. En el foro se pretendía conseguir que los alumnos conociesen el futuro de las comunicaciones haciendo un análisis de las generaciones de comunicaciones de telefonía móvil (GSM, GPRS y UMTS) y la influencia que la legislación tiene en los cambios de generación.

3.1. Preparación de las sesiones

La documentación previa para el debate fue la siguiente:

- Documento sobre las características deseables de un debate.
- Documento sobre las características técnicas de ADSL.
- Documento sobre las características técnicas del cable óptico.
- Documento con una recopilación de 15 artículos de prensa sobre el desarrollo de la banda ancha en España.

Esta documentación puede consultarse en [5]. A partir de esta documentación, los alumnos debían obtener las características técnicas de ambos sistemas.

El foro se planteó como una experiencia de colaboración entre los grupos, para la preparación se proporcionaron diversos documentos:

- Trabajos técnicos sobre GSM
- Trabajos técnicos sobre UMTS
- Especial cronológico sobre la liberalización de las comunicaciones.

En ambas preparaciones no existían ejercicios ni boletines que rellenar, ya que se pretendía que fuese en el aula donde hiciesen el trabajo de síntesis. Esto implica que la preparación previa se redujese a la lectura de la documentación.

3.2. Desarrollo del debate

Para el desarrollo del debate se proporcionó como documentación un enunciado. Esta documentación puede consultarse en [5]. El enunciado estaba dividido en cuatro puntos, estos eran:

1. Descripción del debate: se describía la documentación que debían tener y el "guión" de tiempo de la sesión.
2. Qué se pide: donde se explicaba lo que se esperaba que fuese el desarrollo del debate.
3. Qué se evalúa (y qué no): donde se explicaban los ítems que se esperaba evaluar y qué aspectos no eran evaluables.
4. Condiciones: normas mínimas que se deberían cumplir en el desarrollo del debate.

Sobre la secuenciación del desarrollo del debate, éste se dividió en varias fases.

1. Reparto de material, elección de posturas, explicación de la sesión, dudas y similares.
2. Preparación de los argumentos por parte del grupo.
3. Exposición de los argumentos por parte de los portavoces de los grupos.
4. Desarrollo del debate, defensa de posturas, exposición de datos técnicos.
5. Obtención de conclusiones.

Durante el desarrollo del debate, dos alumnos voluntarios ejercieron de moderador y de secretario (para ayudar al moderador en la toma de tiempos, turnos de palabra y afines). A continuación se detalla el desarrollo de las fases.

En le fase de reparto de material y preparación de grupos, se tuvo mucho cuidado de realizar un reparto de alumnos al azar, procurando evitar los grupos formados. En el debate se trataba de intercambiar opiniones, por lo que se intentó limitar la afinidad, casi segura, entre amigos. En esta parte también se escogieron los voluntarios para hacer de moderador y secretario.

Durante esta fase se explicaron las normas del debate, estas fueron:

- Los turnos de intervención se piden al secretario, diciendo el nombre, sin molestar a los que en ese momento estén interviniendo.
- El tiempo máximo de turno es de tres minutos. Se avisará cuando sólo quede un minuto para agotar ese tiempo.
- Se puede tomar la palabra dos veces seguidas por parte del grupo, pero siempre pidiendo el turno al secretario.
- Si se toma la palabra sin haber pedido el turno, o se excede el tiempo, se penaliza con la pérdida de un turno de réplica.

En la segunda fase, se dejó a los grupos que prepararan los argumentos con los que iban a tratar de razonar sus opiniones. Estos argumentos se anotaban en la correspondiente hoja de evaluación.

En la tercera fase se exponían los argumentos, esta fase no es el debate en sí, pero es necesaria ya que se trata de intercambiar información. En esta fase se debía de dar a conocer las ventajas de la tecnología de banda ancha empleada y los datos técnicos (en esta cuestión se incidió mucho) que avalaran dichas ventajas.

Una vez expuestos los argumentos, se entraba en el debate propiamente dicho. Cabe destacar que se realizaron muchas intervenciones, en un tono muy adecuado, pero con baja argumentación técnica. El moderador y el secretario tuvieron que solicitar que se fundamentaran con datos técnicos veraces y citando las fuentes de procedencia las opiniones presentadas en varias ocasiones. Esto hizo que en un par de ocasiones el debate se parase para dar tiempo a buscar dichos datos.

3.3. Desarrollo del foro

Para el desarrollo, se les marcó claramente a los alumnos cuáles eran los puntos importantes de desarrollo del foro:

1. El objetivo del foro de discusión es conocer en qué consiste el paso de los sistemas de segunda generación (2G) a los sistemas de tercera generación (3G).
2. Los aspectos que se deberán tener en cuenta para el estudio serán los siguientes:
   2.1 Legislación sobre las telecomunicaciones.
   2.2 Liberalización de las comunicaciones en Europa.
   2.3 Aspectos técnicos de las generaciones.
3. Finalmente se deberán obtener unas conclusiones sobre las tendencias futuras.

Una visión más gráfica del desarrollo del foro se puede ver en la figura.

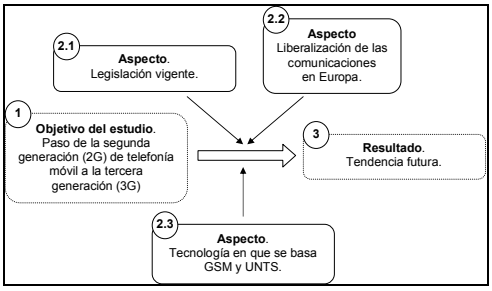


Figura 4. Esquema de desarrollo del foro.

Las fases de desarrollo implicaban un doble cambio de información, el primero dentro del propio grupo, para preparar los resultados que se los solicitaba en las hojas de evaluación. El segundo intercambio de información se producía en la segunda fase, donde se obtenían las conclusiones a partir de la información que suministraban los otros grupos. Esta fase no se puede realizar tan fácilmente en el debate debido a que los intereses de los grupos son tecnologías distintas.

3.4. Evaluación

La evaluación de una sesión de un debate es difícil de realizar ya que se debe aislar la posible visión de ganadores y perdedores en ella. Se buscó evaluar los conocimientos adquiridos por medio de la sesión, para ello se decidió dividir en ítems las hojas de respuestas, de forma que extrajeran las ventajas y los inconvenientes, en diferentes aspectos, de las dos tecnologías sobre las que se había debatido.

Para la evaluación se proporcionó un boletín individual. Esta documentación puede consultarse en [5]. Una captura sobre el tipo y formato se aprecia en la figura 1.

Figura 5. Boletín de evaluación del debate.

.

La evaluación del foro se hizo en los tres aspectos de desarrollo que éste cubría, la del trabajo individual, trabajo en grupo y trabajo global del aula después de la ronda de intercambio de opiniones.

En este caso, a diferencia del debate, donde la evaluación individual evitaba la confrontación entre grupos, se hizo una evaluación en grupo. De una forma similar a la evaluación del debate, la evaluación del foro también se hizo por medio de un boletín en el que los distintos apartados, dirigían la actividad del alumno. Esta documentación puede consultarse en [5].

Una cuestión a resaltar es la necesidad de un boletín de evaluación en el que el alumno encuentre la secuencia de acciones del debate ordenadas, ya que la falta de costumbre del alumno a realizar este tipo de experiencias hace que se le deba orientar un poco por el camino de la actividad. Los recuadros reservados para responder ejercen en este caso de guión de la sesión. Tanto en el debate como en el foro, la documentación seguía la misma filosofía

	 PUNTO DE VISTA DEL ALUMNO	 PUNTO DE VISTA DEL PROFESOR
 Preparación	Se debe preparar la documentación técnica para poder desarrollar las sesiones con suficiente fundamento.	
	INDIVIDUAL: Realizar los resúmenes o esquemas necesarios.	Preparar los puntos principales que se deberán tratar.
 Desarrollo	GRUPO: Selección y preparación de los argumentos.	Resolver las dudas de los grupos en la preparación de argumentos.
	Exposición y argumentación de argumentos.	Colaborar en el desarrollo de la sesión.
 Evaluación	INDIVIDUAL (debate), GRUPO (foro) Obtención de las conclusiones.	Resolver las dudas de los grupos en la preparación de conclusiones.
	Rellenar las hojas de respuesta y el "acta" de la sesión del grupo.	Evaluar las hojas de respuesta y el trabajo en grupo desarrollado.

Figura 6. Tabla resumen de la estructura del debate y el foro.

4. Resultados (evaluación del impacto)

Se realizaron dos tipos de encuestas. El primer tipo fueron las concretas sobre aspectos del desarrollo de cada sesión. El segundo tipo fueron generales para poder realizar una comparativa de los seminarios empleados.

4.1. Encuestas del desarrollo de las sesiones

Para comprobar el desarrollo de la sesión de debate, donde se fomentaba la capacidad de comunicación del alumno, se realizó una encuesta a los alumnos, sobre el desarrollo de la sesión.

En esta encuesta se les preguntó sobre los hábitos de diálogo. Las preguntas fueron las siguientes.

- El vocabulario que se ha empleado ha sido el adecuado.
- Los argumentos que se han dado, han sido claros.
- Las opiniones expresadas han sido respetadas.
- El nivel de comprensión de los interlocutores ha sido bueno.
- El tono de voz empleado ha sido el adecuado.
- Se ha recurrido en exceso al uso de muletillas.
- Se ha recurrido en exceso a la gesticulación.
- El debate/foro ha despertado el interés de los interlocutores.
- El grado de participación ha sido aceptable.
- Se ha mantenido el orden en cuanto al turno de opinión.

Estas preguntas, inciden sobre los aspectos fundamentales que se consideran positivos en un debate y en un foro :

- Compartir una comunicación intelectual.
- Dialogar con unas normas mínimas de educación: escuchar, no cortar la palabra, no acalorarse, emplear un lenguaje adecuado.

- Aportar argumentos apropiados.

Las preguntas fueron las mismas para el debate y para el foro debido a que era necesario poder hacer una comparación en el mismo aspecto de ambas metodologías.

De los resultados, se ha podido deducir que el intercambio de información entre grupos que se dio en el foro no se dio en el debate. Este aspecto era previsible ya que en el debate cualquier intercambio de información puede dar lugar a tener que reconocer una deficiencia en la tecnología, el hecho de que se repitiera que el objetivo era aprender y no "ganar" al otro grupo no cambió esa postura.

4.2. Encuestas del método docente

Estas encuestas se realizaron al final del curso, se preguntó sobre los seminarios realizados. Al debate hay que añadirle dos sesiones de "estudios de caso", ya probadas el curso anterior.

A los alumnos se les pidió que hiciesen una valoración global de las tres metodologías en tres aspectos concretos:

- Facilita el aprendizaje de los contenidos.
- Ayuda al desarrollo de aspectos útiles para la futura carrera profesional.
- Fomenta el trabajo en grupo.

Se les pidió una valoración de 0 a 10 sobre la actividad en general. Los resultados de los tres tipos de seminarios se verán y analizarán a continuación.

En lo que a facilidad para aprender respecta, es decir la sensación de que la metodología empleada ha facilitado el aprendizaje, se pueden ver los resultados en la figura 7.

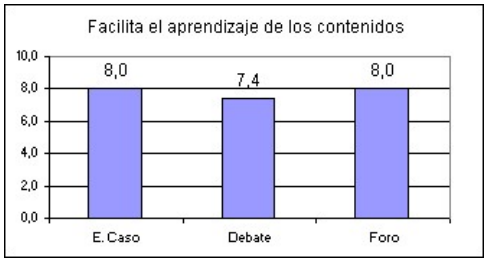


Figura 7. Gráfica comparativa sobre el aprendizaje en estudios de caso, debate y foro.

En el gráfico se puede ver cómo el debate obtuvo peor puntuación que los otros dos métodos empleados de trabajo en grupo en el aula. La impresión de los profesores después de la sesión, fue que no habían aprendido debido a la necesidad de encontrar argumentos en contra de otras posturas, *la confrontación de ideas, aunque ambas sean legítimas, no ayuda al aprendizaje de contenidos técnicos*.

En cuanto a lo que desarrollo de hábitos importantes para la carrera profesional futura (y deseable) de los alumnos, los resultados se pueden observar en la figura 8.

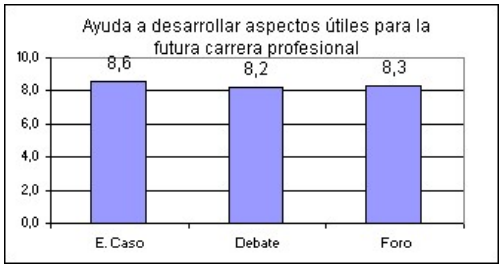


Figura 8. Gráfica comparativa sobre el desarrollo de hábitos profesionales.

En la figura se puede ver cómo el estudio de caso, actividad meramente práctica, pero sin apenas diálogo entre grupos (sí dentro del grupo) y sin intercambio de opiniones, obtiene mayor sensación de utilidad para la vida profesional.

Se puede hacer una analogía entre la conducta que pueda adoptar el alumno en cada tipo de seminario y determinados perfiles profesionales.

El hecho de que un alumno adopte una mejor actitud ante el estudio de caso implica que podrá llevar a cabo un puesto técnico con mayores aptitudes para ello; de la misma forma, del alumno que en el debate se muestre más participativo y demuestre mayores dotes de argumentación, puede deducirse que dispone de mayores aptitudes para ejecutar tareas comerciales; y finalmente, en el foro destacarán los alumnos cuyas características personales se asemejan más a las propias del trabajo ejecutivo, principalmente por el intercambio de información que conlleva, la capacidad de organización e incluso liderazgo.

Por último, en lo que al fomento del trabajo en grupo se trata, pues esa era la finalidad principal de las actividades planteadas, se les preguntó

sobre este aspecto. Los resultados se pueden ver en la gráfica.

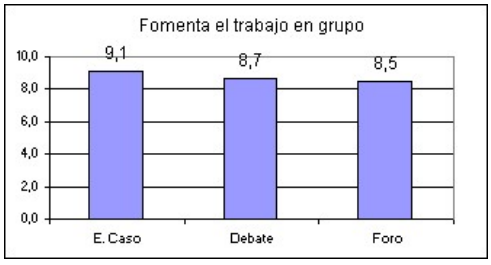


Figura 9. Gráfica comparativa sobre el fomento del trabajo en grupo.

En su origen se esperaba de los resultados de los seminarios, que fuese el foro el que destacase como la técnica donde se produjera una mayor cohesión del grupo y se fomentara mayormente el trabajo en equipo. De las encuestas se ha deducido que no es así. *A la hora de trabajar en equipo, es más fácil compartir información técnica que opiniones.*

5. Conclusiones

Se ha expuesto una metodología que emplea tanto los debates como los foros. Esta metodología se puede emplear para que los alumnos amplíen el tipo de hábitos a desarrollar. Este ha sido el primer curso en el que se han probado. Estas metodologías. Los métodos usados fueron recibidos al principio de forma escéptica. No es habitual realizar un debate, y mucho menos un foro, en el aula universitaria.

El uso del debate fomenta las técnicas de argumentación. Sin embargo, la discusión diluye el tema a tratar, lo que no produce un menor aprendizaje de los elementos técnicos del temario.

El foro fomenta la capacidad de comunicación, por lo que los aspectos técnicos surgen de una forma más natural. Es posible que debido a que un argumento puede no ser técnico el debate deje tiempo a este tipo de explicaciones, mientras que el foro, al no tener que argumentar o defender posturas, permite una mejor apreciación técnica del contenido.

El debate no logró apenas lograr una formación técnica apropiada. Esto se puede deber a varios factores, pero principalmente los alumnos comentaron que les resultaba difícil identificarse con una postura tecnológica, que necesariamente no compartían. El hecho de tener una idea preconcebida que defender hizo que no se volcaran en aprender conceptos sobre la opción contraria.

El foro, en cambio, logró un clima de trabajo y de objetivo común muy agradable, con un intercambio de información muy fluido, posiblemente debido al objetivo compartido.

La predisposición de los alumnos es mucho más favorable hacia el foro en comparación con el debate, es decir, *cuando se trata de aprender es mejor la metodología que no implica una opinión previa, ni una confrontación aunque esta sea amistosa*.

Agradecimientos

A Elisa Valls por su visión e interpretación de los seminarios desde la óptica del campo laboral.

Referencias

[1] Vicerrectorado de coordinación académica y alumnado. Una enseñanza orientada al aprendizaje: proyecto EUROPA. UPV. 2002.

[2] Bonastre A., Buendía F., Pérez M. Equipos y sistemas de transmisión de datos. Servicio de publicaciones de la U.P.V., 1996.

[3] Tanenbaum, Andrew S. Redes de computadoras. Prentice-Hall Hispanoamericana. 1996.

[4] Poza, J.L., Bonastre, A. Oliver, J. Aplicación de las directivas EUROPA en la asignatura de Sistemas de Transmisión de Datos. XXIII Jornadas de la Enseñanza Universitaria de la Informática. 2002.

[5] Página Web de la asignatura Sistemas de Transmisión de Datos. http://www.redes.upv.es/stdfi/.

[6] Newstrom, John W. Edward E. Scannell. 100 ejercicios para dinámica de grupos: una estrategia de aprendizaje y enseñanza. McGraw-Hill, cop. 1989.

[7] RAE. Diccionario de la Real Academia Española. http://www.rae.es/.

[8] Equipo Editorial de Larousse Planeta. Manual Práctico de Expresión Oral. Larousse Planeta. 1995.

La mensajería instantánea al servicio de la docencia

Vicente Galiano Ibarra , Katja Gilly de la Sierra, Alejandro Pomares Padilla
Dept. Física y Arq. Computadores
Universidad Miguel Hernández
03202 Elche (Alicante)
e-mail: {vgaliano,katya,pomares}@umh.es

Resumen

Se presenta la tecnología conocida como mensajería instantánea enfocada a fines docentes. El correo electrónico y la disposición de un sitio *web* no garantiza la entrega al alumno de material docente ni un conocimiento adecuado de la evolución del curso. Frente a las limitaciones de otros sistemas de información, la mensajería instantánea ofrece herramientas que mejoran la comunicación entre el profesor y el alumnado.

El objetivo de esta ponencia se centra en comprobar si la mensajería instantánea ofrece nuevas funcionalidades y mejoras en la docencia universitaria. Para ello, se ha implementado un sistema propio y realizado una serie de pruebas con unos resultados que invitan a seguir utilizando el sistema y ampliar el conjunto de sus funciones.

1. Introducción

En los últimos años, el acceso de los alumnos a las tecnologías de la información ha ido incrementándose a medida que les permitían mayores posibilidades y servicios.

El correo electrónico ofrece una forma rápida y sencilla de comunicación entre alumnado y docente.

Por otro lado, la creación de un sitio *web* de la asignatura, permite a los estudiantes obtener información útil como apuntes, referencias, bibliografía, listados y noticias de la evolución de la asignatura en el curso académico. A pesar de ello, no se garantiza que el estudiante acceda a esos documentos o se le haya informado de un hecho relevante.

Además de las herramientas descritas, venimos observando como el propio alumno utiliza nuevas tecnologías. Una de ellas le permite encontrarse con sus amigos de una manera virtual y hablar con ellos en conversaciones bidireccionales o mutidireccionales; saber quien está conectado en cada momento y poder enviar archivos de una forma sencilla. Este servicio se conoce como mensajería instantánea (M.I.)[1] y existen distintas aplicaciones que implementan estas funcionalidades; las más conocidas son: *MSN Messenger*[2], *ICQ*[3], *AIM*[4] *y Yahoo Messenger*[5].

En las aulas de informática disponibles en el Campus de Elche de la Universidad Miguel Hernández, los estudiantes realizan las prácticas de las asignaturas, buscan información en *internet* e incluso pasan su tiempo de ocio. En la mayoría de ordenadores de estas aulas, la aplicación *MSN Messenger* se encuentra instalada y los alumnos se registran con sus propias cuentas personales para hablar con amigos que estan localizados en otros Campus o bien, en sus casas.

Este hábito representa una potente oportunidad, puesto que el estudiante conoce el funcionamiento y las características de la herramienta. Al utilizar un sistema de mensajería instantánea con fines docentes se abre un conjunto nuevo de posibilidades para una comunicación docente-alumno más directa. Este sistema de mensajería recibe el nombre de Mensajería Instantánea Docente (M.I.D.).

2. Requerimientos

Los estudiantes conocen el funcionamiento de un sistema de mensajería instantánea. Nuestra oportunidad se basa en aprovechar la inercia de este conocimiento. Para ello se ha de diseñar un sistema o herramienta que cumpla con las siguientes características:

1. Funciones y aspecto similar a las aplicaciones conocidas.
2. Comunicación con el sistema administrativo de la universidad: posibilidad de obtener información útil para el estudiante como asignaturas matriculadas, profesores de las asignaturas, etc.
3. Incorporación de aplicaciones docentes que faciliten el aprendizaje del estudiante, la evaluación por parte del profesor y mejoren la comunicación entre todos los elementos que intervienen en el proceso.

3. Aplicaciones docentes

Las aplicaciones que hemos creído interesantes incorporar a nuestro sistema docente de mensajería instantánea son las siguientes:

- Control de asistencia de estudiantes a prácticas.
- Control de actividad del estudiante en el terminal durante las prácticas.
- Publicación de noticias (ejemplos: "Las practicas se entregan el día...", "los apuntes están disponibles en …")
- Publicación de notas con mensajes *online/offline* indicándole a cada estudiante su nota de forma personal.
- Clases multimedia a distancia mediante el uso incorporado de una *webcam* y un micrófono.
- Tutorías a distancia en las asignaturas que se encuentre matriculado el estudiante.
- Envío de dudas *offline*, almacenándolas en el servidor, que serán respondidas cuando el profesor inicie la aplicación.
- Creación de grupos por asignaturas y cursos.
- Practicas evaluadas: serie de mensajes que van saliendo desde el cliente de mensajería del profesor y que el estudiante ha de responder a medida que le aparecen en el desarrollo de la práctica.
- Publicación de archivos para descarga automática (por ejemplo, un profesor configura el sistema para que cuando los usuarios se conecten se descarguen automáticamente un archivo de practicas, de apuntes...).

Además de las aplicaciones citadas, el sistema ha de permitir incorporar nuevas funcionalidades siempre que éstas sean valoradas como positivas e interesantes en la docencia.

Del conjunto de aplicaciones planteadas se han desarrollado algunas de ellas, que explicaremos en profundidad. Éstas han sido probadas experimentalmente en las prácticas de una de las asignaturas que se imparten en el Área de Arquitectura y Tecnología de Computadores.

3.1. Perfiles de Usuarios

En este punto hemos de destacar que el profesor o responsable de las prácticas ha de tener ciertos permisos que le permitan realizar funciones no permitidas a los estudiantes. De este modo, se observa la necesidad de establecer distintos perfiles de usuario dentro del sistema M.I.D.:

- Perfil *Administrador*: configura y gestiona la aplicación, como por ejemplo añadir o borrar usuarios, asignaturas, etc.
- Perfil *Profesor*: Gestiona la actividad de los alumnos. Establece las preguntas y respuestas y puede obtener estadísticas de la evolución de la docencia (pruebas, asistencias, etc).
- Perfil *Alumno*: Accede al conjunto de las asignaturas matriculadas y utiliza su usuario para acceder a prácticas, tutorías y material de apoyo.

3.2. Control de Asistencia de Alumnos

Contabilizar la presencia del estudiante en el horario asignado para la realización de las prácticas representa un pequeño problema "administrativo" para el docente.

La solución más comúnmente adoptada ha sido la de pasar a todos los alumnos una hoja de firmas donde indiquen sus datos personales y confirmen su asistencia. Esta solución presenta algunas desventajas:

- Suplantaciones de identidad
- No se garantiza la realización completa de la práctica.
- Posibilidad de perder o manipular la hoja de firmas durante la práctica.
- Una vez finalizadas todas las prácticas, la tarea de comprobar en las diferentes hojas la

asistencia de todos los alumnos (uno a uno) es un proceso ingrato y costoso.

Sin embargo, mediante una herramienta M.I.D., el problema se resuelve de una manera más sencilla. Se distribuye un nombre de usuario y una contraseña a cada uno de los alumnos. Una vez que el alumno accede a los ordenadores de prácticas el proceso es el siguiente:

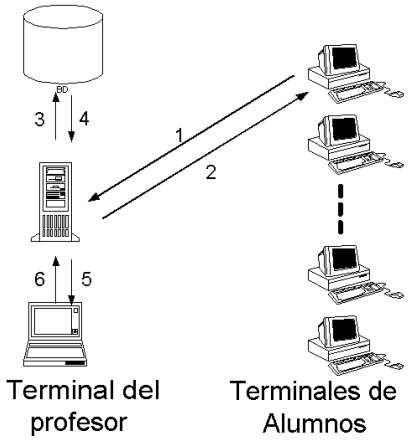


Figura 1. Proceso de Control de Asistencia

1. El estudiante ejecuta la aplicación M.I.D. instalada en todos los terminales de prácticas e introduce su usuario/contraseña. Éstos son enviados a un servidor de mensajería que actúa como sistema centralizado de control.
2. Desde el servidor se actualiza la base de datos informando de la presencia del estudiante en prácticas, y almacenando información útil como un identificador del estudiante, la fecha, la práctica a realizar y otros parámetros de interés.
3. En la base de datos se confirma el usuario y se valida su sesión. Además obtenemos datos útiles para la realización de la práctica: prácticas pendientes, mensajes *offline*, grupo de prácticas, etc.
4. Se informa al estudiante en su terminal de la sesión que acaba de iniciar y le mostramos la información del punto anterior.
5. El profesor puede consultar la asistencia a las prácticas bien mediante un cliente M.I.D. o via web mediante una aplicación *web* que nos muestre la información en formato HTML (esta segunda opcion resulta más práctica y ágil para la generación de informes).
6. Durante el transcurso de la sesión, el profesor puede solicitar información de la evolución de la práctica. Algunos datos útiles son: número de asistentes, preguntas contestadas (ver apartado 3.3), tiempos de respuestas, actividad de los alumnos y alumnos ausentes.

3.3. Prácticas evaluadas

La mensajería instantánea posibilita algunas facilidades que otros sistemas de evaluación no alcanzan a vislumbrar.

En la enseñanza tradicional de una asignatura de informática o de otra materia, la evaluación de los conocimientos prácticos se suele realizar mediante la entrega de un trabajo o programa realizado por el estudiante.

Se detecta que este método concentra el periodo de aprendizaje del estudiante en fechas próximas a la entrega de los trabajos. Además, posibilita la semejanza entre los trabajos presentados por los estudiantes.

Adicionalmente, el estudiante puede ser evaluado de una forma continua y sencilla mediante el uso de mensajería instantánea. Dicho proceso se ilustra en la Figura 2.

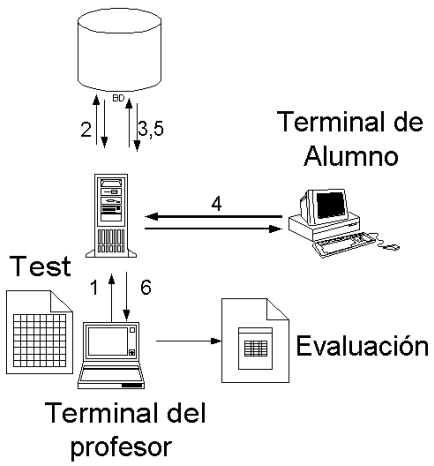


Figura 2. Evaluación de las prácticas

1. El profesor responsable configura la sesión de prácticas introduciendo preguntas e indicando posibles respuestas. Las preguntas de tipo *test* (a, b, c y d) se muestran como idóneas para este sistema, puesto que el método de corrección se

puede programar. En otro caso, se pueden configurar preguntas que no sean tipo *test* pero requiere de la posterior corrección por parte del profesor. Además se deben indicar otros parámetros como el orden y el tiempo en el que deben ser formuladas.

2. La configuración de la práctica guiada se almacena en la base de datos[1] del sistema.
3. Una vez que el alumno inicia la sesión, la aplicación servidor de M.I.D. obtiene la secuencia de preguntas desde la base de datos.
4. Durante el transcurso de la práctica el servidor envía las preguntas que el estudiante deberá responder.
5. Las respuestas de cada estudiante se almacenan en la base de datos para su posterior análisis.
6. Finalizada la sesión de prácticas, el profesor podrá recoger estadísticas de la evolución y comprensión de los estudiantes de una forma inmediata.

4. Implementación del sistema

El objetivo que se ha tratado de alcanzar ha sido el de verificar la mensajería instantánea como una herramienta adecuada en la docencia. Por tanto, no se detalla en el presente documento la explicación del desarrollo de la herramienta *software* utilizada[6].

En la Figura 3 mostramos el aspecto de las aplicaciones desarrolladas de servidor y de cliente. Su funcionalidad ha de ser muy similar a alguna de las ya conocidas por el alumnado. La aplicación que hemos detectado que más usaban los alumnos era *MSN Messenger*[2].

Para poder evaluar esta técnica hemos implementado el sistema y las aplicaciones descritas en los apartados 3.2 y 3.3.

El sistema ha sido probado en las prácticas de la asignatura "Redes de Comunicaciones"[7] que se imparte en la Escuela Politécnica Superior de Elche, perteneciente al plan de estudios de la Ingeniería Técnica de Telecomunicaciones.

Las prácticas se realizan en aulas con computadores que disponen de dos particiones: una con *Windows 98* y otra con *Linux*. Por tanto, se han desarrollado dos versiones de la aplicación cliente, una para cada sistema operativo utilizado en prácticas.

Por otro lado, el sistema operativo del servidor es *Windows 2000*, y la aplicación ha sido desarrollada con *Visual C++* y *Visual Basic*.

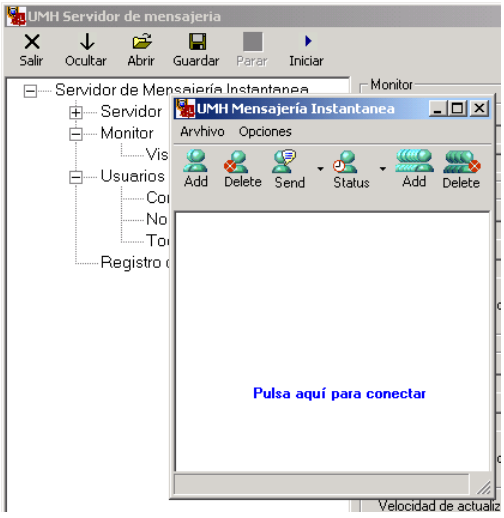


Figura 3. Aplicaciones Servidor y Cliente M.I.D.

5. Evaluación de la herramienta

La asignatura tiene un total de ocho prácticas en el primer cuatrimestre de dos horas de duración cada una. Éstas prácticas fueron realizadas desde noviembre de 2002 a enero de 2003 con un total de 72 estudiantes.

En la primera sesión se introdujo al estudiante en el manejo del sistema de mensajería. La utilización de esta herramienta se planteó como opcional. Sin embargo, la evaluación mediante este sistema influiría en la nota final de prácticas.

Durante la primera sesión de prácticas y a partir de la primera media hora, el 90% de alumnos habían iniciado su sesión y se avanzaban en la realización de la práctica.

Gracias a que el aspecto y las funciones del cliente son muy similares a las de uno de los sistemas más extendidos en la actualidad (*MSN*

1. [1] No entramos en detalle de la estructura de tablas de la base de datos, puesto que en este documento se trata de explicar un método docente y no su implementación real.

Messenger), los alumnos aprendieron rápidamente el manejo del cliente.

Durante las prácticas posteriores, los alumnos confirmaban su asistencia mediante el sistema de mensajería y el 100% de los asistentes a prácticas tenía iniciada su sesión.

Por otro lado, se configuraron un total de doce preguntas de dificultad media por cada sesión de prácticas de dos horas de duración. Cada pregunta hacía referencia a una cuestión que el alumno habría de haber desarrollado en esa sesión. El tiempo máximo de respuesta se configuró a treinta segundos.

	Firma M.I.D.	Firma Hoja	100% <X< 75%	75% <X< 50%	50% <X< 0%
P1	57	2	23	26	8
P2	55	0	25	24	6
P3	54	0	26	24	4
P4	58	0	27	24	7
P5	55	0	19	27	9
P6	56	0	28	22	6
P7	54	0	28	23	3
P8	54	0	27	17	10

Tabla 1. Resultados obtenidos en prácticas

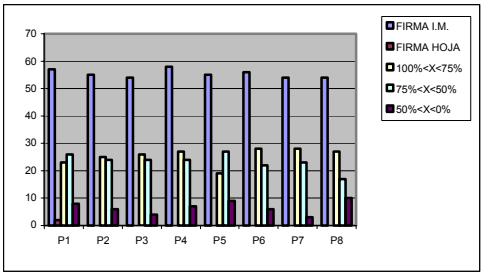


Figura 4. Control de asistencia y porcentajes de respuestas correctas

En la Tabla 1 se detallan los valores absolutos de asistentes a prácticas, número de estudiantes que confirmaron su asistencia mediante M.I.D. y mediante la hoja de firmas. Las tres últimas columnas detallan el porcentaje de respuestas acertadas durante la realización de las mismas. Estos datos se representan en la Figura 4.

Durante el desarrollo de las diferentes prácticas, se detectó que una pequeña parte del alumnado no estaba familiarizado con una herramienta de mensajería instantánea. Este hecho contradijo la suposición de la mensajería instantánea como una herramienta ampliamente conocida y motivó ofrecer una sesión adicional para practicar con la aplicación cliente.

6. Análisis de resultados

Se ha de tener en cuenta que las pruebas han sido realizadas con una población reducida y no garantiza la extrapolación a los resultados que se puedan obtener con una población mayor.

Para poder valorar los resultados del sistema M.I.D. hemos de tener en cuenta la opinión por parte del alumnado. Es necesario conocer de un modo objetivo la opinión real de los alumnos. Para ello se ha utilizado el sistema de encuestas de la página de la asignatura [7]. La encuesta ha estado accesible desde comienzos de las prácticas y los estudiantes han expresado su valoración.

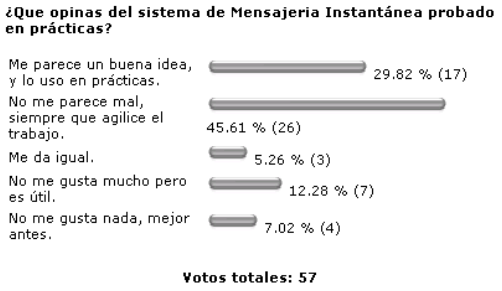


Figura 5. Captura de la encuesta realizada en el sitio en internet de la asignatura

La encuesta parece confirmar que los alumnos utilizan la herramienta y les resulta útil en el aprendizaje.

Por otro lado, en la Tabla 1 se aprecia cómo el número de alumnos que responden bien a más del 75% de las preguntas aumenta a medida que se suceden las prácticas. Es probable que esta evolución se deba a una adaptación por parte del alumno/a al proceso de preguntas y respuestas.

No obstante, al finalizar la práctica número 5, el número de estudiantes que habían superado correctamente el 75% de las preguntas fue sensiblemente inferior a prácticas anteriores. Este detalle puede ser tenido en cuenta para detectar y repasar aquellos conceptos que parecen no haber

quedado claros durante el desarrollo de la sesión práctica.

7. Conclusión

Se ha adaptado una tecnología de aplicación general a la docencia. La mensajería instantánea nos proporciona una herramienta útil para tutorías, prácticas y comunicación con el alumnado. Las pruebas realizadas y la respuesta por parte del alumnado parecen confirmar algunos de los objetivos que se habían marcado. La posibilidad de obtener información casi inmediata del rendimiento académico de cada alumno y las mejoras en los procesos de evaluación se han mostrado como mejoras alcanzadas con un sistema de mensajería.

No se limitan las posibilidades a los dos casos presentados. El conjunto de aplicaciones y utilidades es mucho más amplio; además, este sistema puede ser empleado tanto en asignaturas de informática como en asignaturas pertenecientes a otra disciplina.

Referencias

[1] Congreso de Mensajería instantánea , http://www.jupiterevents.com/im/fall02/
[2] http://www.msn.com
[3] http://www.icq.com
[4] http://www.netscape.com
[5] http://www.yahoo.com
[6] Halshall F., *Multimedia Communications* , Addison-Wesley 2001
[7] http://obelix.umh.es/redes

Adaptación de la técnica de planificación del trabajo personal aplicada a la educación

Antonio de Amescua, Juan José Cuadrado, Luis García Sánchez, María Isabel Sánchez
Departamento de Informática
Universidad Carlos III de Madrid
Avda. De la Universidad, 30
28911 Leganés (Madrid), Spain
e-mail: cuadrado@inf.uc3m.es

Resumen

Este artículo describe el método utilizado para implantar el uso de la técnica de Planificación del Trgabajo Personal (Personal Work Planning PWP), como una herramienta de gestión del tiempo, aplicable a los trabajos prácticos de los alumnos de Ingeniería del Software II de la Universidad Carlos III de Madrid. Además, se presentan los resultados obtenidos del estudio empírico llevado a cabo para determinar el nivel de satisfacción de los estudiantes con la utilización de esta técnica.

1. Introducción

La técnica de Planificación del trabajo personal (PTP) [2] [3] ayuda a mejorar los procesos individuales utilizados por los Ingenieros de Software mediante la utilización de históricos de datos sobre la ejecución real de los mismos procesos realizados por el mismo ingeniero en proyectos previos. Además, esta técnica puede ser utilizada para estimar con mayor precisión el tamaño, la duración y el esfuerzo de las tareas que deban ser realizadas [1].

El PTP propone los siguientes pasos para preparar un plan efectivo de trabajo personal dentro de un proyecto software:

1. Comenzar con una definición explícita del trabajo que se debe realizar y comprobar que es el exactamente requerido por el cliente.
2. Descomponer el trabajo que requiera mas de unos pocos días en pequeñas tareas y estimar cada una de ellas de forma separada.
3. Hacer las estimaciones básicas del trabajo utilizando datos históricos.
4. Grabar las estimaciones y compararlas con los resultados reales.

Como se verá en el apartado Descripción del Experimento, los autores partiendo de ideas extraídas de [4] y [5] modificaron este proceso de planificación personal con el fin de que los alumnos pudieran aplicarlo en la medida que lo requerían sus trabajos prácticos de la Asignatura de Ingeniería del Software II con los siguientes objetivos:

1. Enseñar a los estudiantes la importancia de trabajar con un método disciplinado de gestión del tiempo y del esfuerzo que cada individuo gaste en un proyecto software.
2. Observar el nivel de satisfacción de los estudiantes tras aplicar esta técnica. Para comprobar la consecución de este objetivo, los estudiantes debieron rellenar unos cuestionarios diseñados específicamente para este propósito [Anexo B].
3. Observar si la carga lectiva adicional correspondiente al aprendizaje de PTP tendría algún efecto en las calificaciones finales de los alumnos. Para comprobar si este objetivo ha sido cubierto sólo parte de los alumnos implicados en el experimento aplicó las técnicas, actuando el resto como elemento de comparación

2. Descripción del Experimento

El experimento fue llevado a cabo por los estudiantes de Ingeniería del Software II de tercer curso de Ingeniería Técnica en Informática de Gestión. Esta asignatura abarca conceptos y técnicas de gestión de proyectos. El curso se imparte por la mañana y por la tarde, y los profesores decidieron llevar a cabo el experimento en el grupo de mañana, al cual se denominará grupo piloto, el grupo de la tarde se denominará grupo de control.

Todos los estudiantes de la asignatura fueron agrupados de tres en tres para llevar a cabo las prácticas. El grupo piloto contó con un total de 23 equipos y el grupo de control contó con un total de 18 equipos. Ambos grupos tuvieron que realizar la misma práctica, pero sólo el grupo piloto utilizó la técnica PTP.

Al comienzo del experimento los profesores proporcionaron a todos los alumnos las especificaciones del proyecto software objeto de la práctica y la dividieron en tareas de una semana de duración [Anexo A].

Para que los equipos del grupo piloto pudieran poner en práctica el PTP, se diseñaron unas tablas de registro diario y semanal de tiempo invertido en la realización de las tareas propuestas para la práctica. Estas tablas fueron implementadas en Excel y distribuidas a todos los equipos.

Los estudiantes tenían que rellenar la tabla de tareas con los tiempos que estimasen para la realización de las mismas antes de realizar las prácticas, esto es, sin tener ninguna experiencia previa. Esta tabla se enviaba semanalmente a los profesores vía e-mail. Cada semana, durante las siguientes nueve semanas, y utilizando el conocimiento adquirido, cada equipo del grupo piloto, revisaba la planificación inicial y reasignaba tiempo a las tareas que no habían sido realizadas. Dicha tabla actualizada era enviada a los profesores.

A través de la tabla de registro diario los equipos del grupo piloto indicaban el tiempo invertido en las prácticas. Dicha información era utilizada para completar la tabla de registro semanal. La información recogida en las tablas de registro diario y semanal eran en todos los casos tablas que contenían datos reales.

Al final del cuatrimestre el grupo piloto rellenó el cuestionario en el que reflejó la satisfacción de la técnica PTP. Para completar el experimento los profesores evaluaron los datos obtenidos.

3. Resultados

3.1. Resultados obtenidos de las tablas de seguimiento

Después de completar las prácticas, los datos recogidos en las tablas de registro fueron analizados para identificar la evolución de las estimaciones de tiempo realizadas por los equipos del grupo piloto. El número final de equipos evaluados fueron 19 ya que 4 grupos habían remitido datos insuficientes o evidentemente erróneos. La relación entre el tiempo medio estimado y el tiempo real se muestra en la Tabla 1

SEMANA		
	Tiempo Medio Estimado (TME)	Tiempo Medio Real (TMR)
Semana 1	12,17	8,66
Semana 2	11,65	9,16
Semana 3	14,10	8,42
Semana 4	6,26	8,05
Semana 5	9,91	13,03
Semana 6	21,32	7,91
Semana 7	7,97	10,41
Semana 8	8,21	10,28

Tabla 1. Tiempo medio y tiempo real para los 19 grupos evaluados

Los datos de la Tabla 1 pueden observarse en la figura 1. Si analizamos con detenimiento dicho gráfico, podemos concluir que al comienzo del proyecto el tiempo medio estimado era siempre superior al tiempo medio que realmente se dedicaba a completar las tareas asignadas. Esta tendencia a la sobreestimación se da siempre en equipos inexpertos pero una vez que los equipos del grupo piloto fueron adquiriendo experiencia y utilizando el PTP dicha tendencia a la sobreestimación se invirtió (a partir de la cuarta semana) pasándose a observar una tendencia a la subestimación.

Mención a parte merecen los resultados obtenidos para la semana 6, en la que se observa una clara sobreestimación del esfuerzo. Este valor anómalo se puede explicar si se tiene en cuenta que las tareas que los profesores habían planificado, para ser realizadas por los alumnos en dicha semana eran absolutamente desconocidas para ellos, lo cual dificultaba sobremanera la realización de las estimaciones conducentes a determinar el tiempo que habría de emplearse para completarla.

La figura 2 muestra la diferencia entre los valores de tiempo estimado y observado para la realización de las tareas. Se han tomado en valor absoluto porque lo que se intenta resaltar es la exactitud en las estimaciones, sin tener en cuenta si se subestima o se sobreestima. Se ha excluido del análisis la información recogida para la semana 6 porque es un valor claramente anómalo.

Como puede observarse en la línea de tendencia existe una disminución en el error absoluto cometido al tomar las estimaciones como valores reales, por lo que la aplicación del PTP está teniendo un efecto inequívocamente positivo en la capacidad de los equipos para estimar con precisión las tareas que deben acometer para la realización del trabajo práctico propuesto.

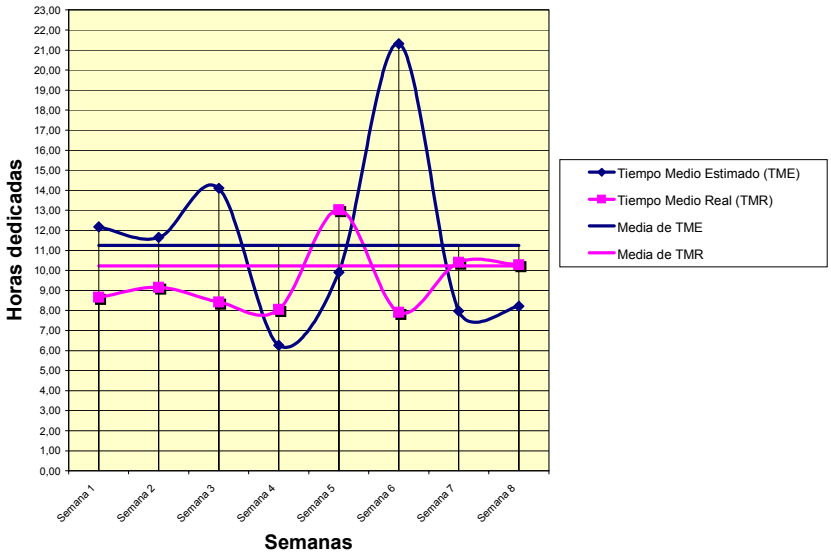


Figura 1. Relación entre tiempo medio estimado y tiempo real

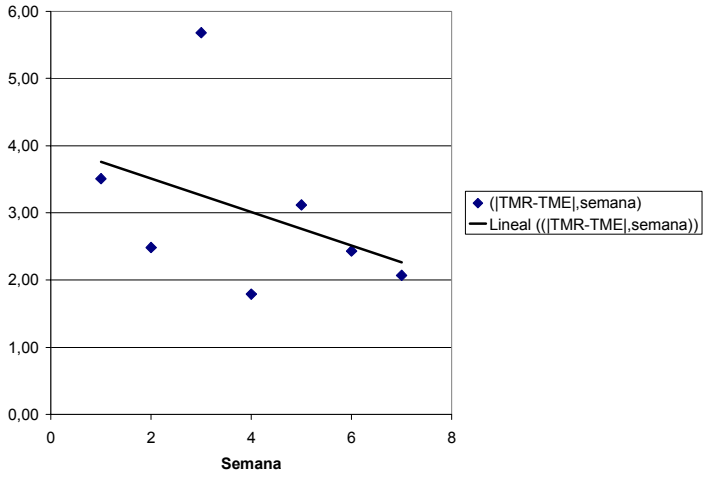


Figura 2. Diferencia entre tiempo medio estimado y real Resultados obtenidos del nivel de satisfacción de los estudiantes

En la Tabla 2 se presentan los resultados obtenidos de los cuestionarios de valoración personal de la técnica de PTP.

Cada una de las afirmaciones ha sido evaluada independientemente y los datos aparecen en Figura 3.

Cuestionario de satisfacción sobre PTP

	Valor 1: Desacuerdo	Valor 2: Indiferente	Valor 3: De acuerdo	Valor 4: Completamente de Acuerdo	No sabe/ No contesta
Afirmación 1	1	14	36	10	0
Afirmación 2	1	19	26	15	0
Afirmación 3	2	22	23	12	2
Afirmación 4	1	19	32	9	0
Afirmación 5	5	15	28	13	0
Afirmación 6	6	17	24	14	0
Afirmación 7	6	25	21	7	2
Afirmación 8	3	9	26	19	4

Tabla 2. Resultados de los cuestionarios de satisfacción personal

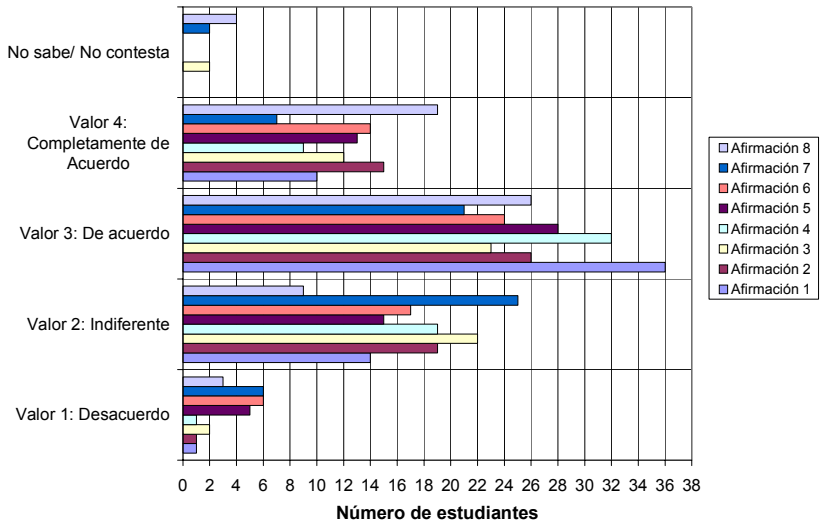


Figura 3. Resultados de los cuestionarios de satisfacción

4. Conclusiones

Durante la realización de este experimento se tuvieron que resolver algunos problemas, principalmente relacionados con un mal entendimiento de algunos aspectos de las tareas y la técnica que se estaba utilizando.

Del análisis de las hojas de seguimiento de tareas se puede concluir que la técnica PTP fue bien aceptada por los estudiantes. Sin embargo debemos resaltar que los estudiantes tuvieron dificultades en aprender a estimar utilizando esta técnica, ya que las tareas eran completamente diferentes cada semana, lo cual no les permitía utilizar la experiencia adquirida en actividades previas para estimar el tiempo de desarrollo de las actividades siguientes. Este sería un aspecto muy importante a tener en cuenta para alguien que estuviera interesado en repetir la experiencia. Una posible solución sería introducir tareas semejantes a lo largo del desarrollo del proyecto.

Con el fin de comprobar el impacto que el uso de la técnica de PTP ha tenido en las calificaciones, se comparan en la Figura 4 las calificaciones obtenidas en el grupo piloto y en el de control. El grupo de control tiene una media de 7,1 mientras que en el grupo piloto es de 7,0. Esto indica claramente que el exceso de carga teórica que implicó el aprendizaje de la técnica PTP no afectó a las notas obtenidas por los alumnos sobre el temario oficial de la asignatura. Sin embargo la aplicación de la técnica PTP incidió en la mejora de la calidad de las prácticas de los equipos pertenecientes al grupo piloto.

Como se puede ver en el anexo A las tareas propuestas a los alumnos se corresponden con distintas partes de un trabajo práctico particularizado para la asignatura de ingeniería II, pero para aplicarlo a otras materias sería suficiente con:

1. Realizar un seminario previo de aproximadamente dos horas de duración sobre PTP
2. Dividir la practica correspondiente en tareas simples y evaluar cada una de ellas por separado.

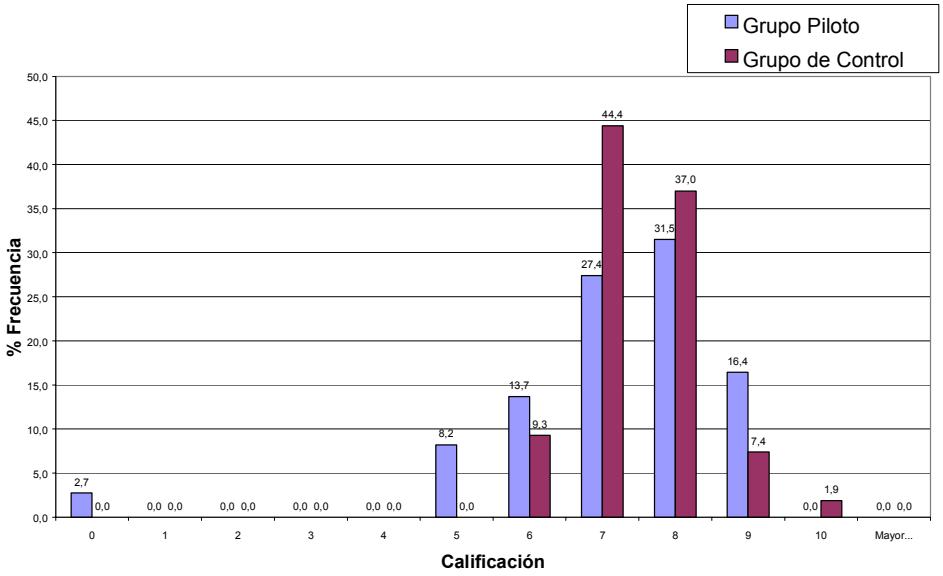


Figura 4. Distribución de calificaciones

Referencias

[1] Disney-AM, Johnson-PM – A critical analysis of PSP data quality: results from a case study. Empirical Software Engineering. Vol 4 nº4, 1999; p. 317-49.

[2] Humphrey, Watts S. (1999), Introduction to the Personal Software Process. SEI Series in Software Engineering, Addison Wesley.

[3] Humphrey-WS - Introducing the personal software process. Software Eng. Inst., Carnegie Mellon Univ., Pittsburgh, PA, USA. Annals-of-Software-Engineering. vol.1; 1995; p.311-25

[4] Miller-J; Mingins-C - Putting the practice into software engineering education. Proceedings. 1998 International Conference Software Engineering: Education and Practice (Cat. No.98EX220). IEEE Comput. Soc, Los Alamitos, CA, USA; 1998; xi+421 pp. p.200-8

[5] Mingins-G; Miller-J; Dick-M; Postema-M - How we teach software engineering. JOOP. vol.11, no.9; Feb. 1999; p.64-6, 74

Anexo A. Distribución semanal de tareas

Asignatura: Práctica Nº:
Grupo Nº:
Componentes: Fecha:

Trabajo a realizar:		Tiempo estimado				
FECHA	TAREA	Clase	Estudio	Ejercicios	Práctica	Total
de 23/03/00 hasta 29/03/00	Lectura detallada y comprensión de la especificación de requisitos propuesta.					0
	Primera aproximación a los DFDs de nivel intermedio.					0
	Construcción de los DFDs de nivel intermedio					0
de 30/03/00 hasta 05/04/00	Construcción del modelo de datos correspondiente al caso de estudio.					0
	Versión definitiva de los DFDs de nivel intermedio.					0
	Construcción de los DFDs de nivel inferior y cero (Diagrama de contexto).					0
de 06/04/00 hasta 12/04/00	Modelo E/R normalizado.					0
	Comprobar la consistencia entre modelo de procesos y modelo lógico del proyecto.					0
	Refinar la hoja Excel de estimación que automatiza la estimación del proyecto por el método MK-II, con los parámetros concretos del proyecto actual.					0
de 13/04/00 hasta 19/04/00	Contar entradas, salidas y entidades para cada transacción lógica definida.					0
	Asignar valores a las características C1..C19 propuestas por el método de estimación, para cada transacción lógica.					0
de 27/04/00 hasta 03/05/00	Documentar adecuadamente todo lo necesario para hacer entrega de una memoria con el modelo de procesos, modelo de datos y estimación realizada. Así como una explicación de los motivos que han llevado a la selección de los valores de las características C1..C19 de las transacciones lógicas.					0
de 04/05/00 hasta 10/05/00	Lectura detallada del enunciado de la práctica correspondiente a "Organización, Planificación, y Seguimiento de Proyectos".					0
	Si se aportan nuevos datos al proyecto, re-estimar.					0
	Lectura detallada de los pasos propuestos en la metodología Métrica Versión 2.0, relación entre estos y productos a obtener.					0
	Construcción de los WBS, PBS y RBS.					0

Anexo B. Cuestionario de valoración personal de la técnica PTP

		Valor asignado				
		1	2	3	4	NS/NC
1	Gracias a esta técnica he podido tener una visión mucho más clara de las tareas que debía llevar a cabo, de manera que estuviera previsto de antemano el tiempo y los recursos que se debían asignar a cada una de ellas.					
	Comentarios:					
2	Gracias a esta técnica de planificación del trabajo, se ha tenido mucho mayor control a la hora de hacer previsiones, controlar posibles cambios, etc., sobre cada una de las tareas que era preciso llevar a cabo.					
	Comentarios:					
3	El equipo de trabajo ha podido hacer una asignación de tareas mucho más realista que si no se hubiesen utilizado las técnicas de planificación del trabajo personal.					
	Comentarios:					
4	Las técnicas de planificación del trabajo personal me han ayudado a mejorar mis propias estimaciones de tiempo, es decir he podido aprender de la planificación previa de otra actividad similar para así mejorar planificaciones posteriores de actividades similares.					
	Comentarios:					
5	Gracias a las técnicas de planificación del trabajo personal, el rendimiento de los miembros del equipo de trabajo se ha visto mejorado pues el conocimiento del estado de la práctica ha sido mucho mayor.					
	Comentarios:					
6	El seguimiento que hemos podido realizar sobre la práctica, ha fomentado la colaboración entre los miembros del equipo de trabajo así como la implicación de estos en el trabajo a realizar, todo ello gracias a las técnicas de planificación del trabajo personal.					
	Comentarios:					
7	Sin duda utilizaría estas técnicas de planificación personal del trabajo, en otras asignaturas de la carrera.					
	Comentarios:					
8	Sin duda utilizaría estas técnicas de planificación personal del trabajo en otros proyectos dentro del mundo laboral.					
	Comentarios:					

El proceso de aprendizaje: herramienta para el desarrollo de competencias profesionales en primero de informática

Juan José Escribano Otero, Estrella Gómez Fernández
Maria Teresa Villalba de Benito, Manuel Ortega Ortiz de Apodaca
Dpto. de Programación e Ingeniería del Software
Escuela Superior de Informática
Universidad Europea
E:mail: jje@dpris.esi.uem.es,
estrella@dpris.esi.uem.es,
maite@dpris.esi.uem.es,
mortega@dpris.esi.uem.es

Resumen

La convergencia en la universidad europea iniciada con el Tratado de Bolonia, está provocando en la universidad española la revisión de los planes de estudios ofertados tanto en el caso de pre-grado como en el de post-grado. El acercamiento al desarrollo profesional de los planes de estudio, llevó a la Escuela Superior de Informática de la Universidad Europea de Madrid a incluir, en el primer curso de sus titulaciones, una asignatura llamada *Introducción a la informática en Red* donde, además de desarrollar conceptos fundamentales de las redes de ordenadores (y de Internet muy en especial), se pudiera experimentar con modificaciones en la metodología docente y en el sistema de evaluación, con el objetivo de hacer parte activa al alumno en ambos aspectos.

1. Introducción

La declaración firmada en Bolonia por 29 ministros de Educación, el 19 de junio de 1999 [3], supone el punto de partida de una de las revoluciones más trascendentales de la educación superior en Europa. Basándose en el informe "Trends in Learning Structures in Higher Education" realizado por Guy Haug y Jette Kirstein [7] para la confederación de conferencias de rectores de la UE, los ministros reunidos emiten un documento [3] en el que se definen las líneas estratégicas, con el objetivo de conseguir un espacio educativo superior único para toda al UE, en el año 2010.

En la declaración se fijan seis objetivos estratégicos que afectan fundamentalmente a la estructura de las titulaciones, y a la movilidad de pre y post graduados. En resumen, se trata de potenciar al máximo la posibilidad de que los estudiantes puedan cambiar de estado durante los estudios de pre-grado y post-grado, y que los títulos así obtenidos tengan validez en toda la UE. Como requisito indispensable para ello, hay que unificar la forma de medir la carga lectiva que lleva aparejada una titulación. La propia declaración señala la necesidad de establecer un sistema de créditos, similar a los ECTS de la iniciativa Erasmus.

Por mandato de la CRUE, el grupo de trabajo presidido por Domingo Docampo, Rector de la Universidad de Vigo, elabora el informe titulado "El crédito Europeo y el sistema educativo Español"[6], donde se analizan las consecuencias de la implantación de la nueva métrica en los planes de estudios españoles. Lo que queda claro en el informe citado es que la nueva métrica (pervive el nombre de crédito) debe valorar no solo, como ocurre actualmente, el número de horas que el alumno emplea en asistir a clase, ya sea de teoría o de prácticas, si no que es necesario computar también aspectos como el número de horas que el alumno emplea en estudiar, en confeccionar trabajos y, de forma general, en todas aquellas actividades relacionadas con su actividad formativa (algunos reivindican, incluso, el tiempo empleado en realizar fotocopias). Se incorporan también otro tipo de actividades encaminadas a que el alumno no solo adquiera conocimientos si no que también adquiera determinadas destrezas (competencias es la

palabra de moda), que le sean de utilidad en su posterior trayectoria profesional.

Es evidente que la nueva métrica y la limitación por curso de créditos ECTS, tienen consecuencias importantes en la tarea del profesor. Ahora, el acento se pone en el trabajo del estudiante y no sólo en las horas de clase que recibe. Lo importante es, por tanto, que "el alumno aprenda" y no que "el profesor enseñe". El profesor pasa de ser un transmisor y evaluador de conocimientos, a ser la persona que "conduce y asiste" el aprendizaje del alumno. Es necesario, por tanto, cambiar la dinámica de las clases, de forma que éstas no sean meros actos de transmisión/recepción de información, convirtiéndolas en actos en los que el alumno tiene un protagonismo activo.

En la Escuela de Superior de Informática de la Universidad Europea de Madrid, hemos considerado que la asignatura *Introducción a la Informática en Red*, por sus especiales características, constituye un banco de pruebas perfecto para trabajar en experiencias innovadoras, que nos permitan reorientar nuestra práctica docente y adaptarla a las nuevas exigencias emanadas del compromiso de Bolonia.

2. Asignatura de Introducción a la Informática en Red

A menudo, los alumnos que acceden al primer curso de una de las titulaciones de informática aportan una visión extraña de lo que supone o debe suponer el estudio de una ingeniería. Probablemente el hecho de que la misma herramienta (el ordenador) que usan para sus juegos, sus comunicaciones y su búsqueda de información coincida con la herramienta de la Informática por antonomasia produce una distorsión en el nuevo universitario sobre la profundidad y rigor de la computación como disciplina.

Por otra parte el primer curso de cualquier ingeniería es, probablemente, el que encierra mayor carga de contenidos teóricos en casi todas las titulaciones. Por ejemplo, en la Universidad Europea de Madrid, en los planes de estudio de 1996, la media en las tres carreras impartidas por la Escuela de Informática – Informática Superior, Informática Técnica de Sistemas e Informática Técnica de Gestión- la media de créditos impartidos en un laboratorio suponía un 14% sobre el total de ese curso. Esta circunstancia no está exenta de lógica ya que dichos conocimientos deben ser los pilares sobre los que se sustenten las posteriores prácticas.

Sin embargo esta situación provoca, en primer lugar, que el alumno se encuentre con mucha carga teórica en su primer año, justo cuando más motivado esta para experimentar con ordenadores. Además, el alumno de informática ve cómo los compañeros que cursan otras carreras utilizan el ordenador más que él y por consiguiente "saben más informática que él".

Todo esto motiva (junto con otros factores, naturalmente) que la tasa de abandono del primer año supere con creces a la de cualquier otro curso.

Para paliar estos efectos, en la Escuela Superior de Informática de la UEM se ha incluido como asignatura obligatoria de las tres titulaciones ofertadas la asignatura *Introducción a la Informática en Red* en la revisión del Plan de Estudios que se publicó en el año 2000. Con esta inclusión, el número de créditos impartidos en el laboratorio en este primer curso representa un 24% sobre el total.

Es sobre esta asignatura, su temario, su necesidad de coordinación entre profesores, su metodología docente, su sistema de evaluación y, sobre todo, sobre los objetivos tanto de contenidos como extra-académicos sobre lo que versa este artículo, confiando en que lo en él expuesto pueda servir de, al menos, herramienta de reflexión y punto de partida para la confección e impartición de otras asignaturas de primeros cursos con contenidos fuertemente prácticos, sin que por ello se descuide el necesario nivel académico de una enseñanza superior.

Como se puede consultar en los planes de estudios [2], esta asignatura es obligatoria, de primer curso, primer semestre, 4H/semana (6 créditos) para las tres titulaciones de informática actualmente ofertadas por la Escuela de Informática. Complementa un año cargado de matemáticas (Matemática discreta, Álgebra, Análisis), Física, asignaturas introductorias de Programación (Introducción a la Programación) y otros laboratorios (Laboratorio de programación, Laboratorio de Computadores).

2.1. Objetivos y dinámica de la asignatura

Con el propósito de centrar el contexto en el que se han aplicado las diversas ideas innovadoras, este epígrafe pretende presentar de forma esquemática las directrices de la asignatura.

Los objetivos que se persiguen se pueden resumir en:

1.-Familiarizar al alumno con el trabajo en red, introduciendo los conceptos generales sobre redes y muy particularmente sobre Internet

2.-Enseñar al alumno a utilizar los servicios de Internet para extraer información

3.-Permitir el desarrollo de ciertas competencias personales en el alumno [6]. Dichas competencias representan un subconjunto de las propuestas por la UEM en su informe sobre el particular [7]: Iniciativa, trabajo en equipo, innovación y creatividad, confianza en sí mismo, habilidades comunicativas, responsabilidad, flexibilidad, conciencia de los valores éticos, planificación

Para conseguirlos, la metodología de la asignatura se divide en las siguientes fases:

FASE INICIAL: ejecución, corrección y discusión en clase de un TEST de conocimientos previos. Duración de la fase inicial: una semana (4 horas de clase)

FASE I (Transmisión de conocimientos): clases magistrales explicando conceptos teóricos, junto con pequeñas prácticas para familiarizarse con los programas clientes y con los servicios de Internet, así como explicar al alumno el formato en el que deberá entregar sus posteriores trabajos. Duración de la Fase I: 6 semanas.

FASE II (Trabajo en equipo y búsqueda de información): confección de grupos, prácticas de búsqueda de información a cerca de temas relacionados con las redes e Internet. Duración: 5 semanas.

FASE III (Exposición y evaluación de trabajos): exposiciones de trabajos en el aula, evaluación de los trabajos y de la exposición por parte del grupo. Duración: 3 semanas.

2.2. Ejercicios intergrupales

Además del trabajo propio de cada profesor con su grupo de clase, se decidió introducir algunas actividades propias de la asignatura que trascendieran el ámbito del aula. Estas actividades pretenden dar coherencia horizontal a la asignatura. Dichas actividades son:

Cybergimkhana periódica: consiste en un juego de búsqueda de información en Internet, en el que el alumno va superando pruebas mediante búsquedas en la Red [8].

Competición grupal: pruebas que se resuelven en equipo. Como ejemplo de una de estas pruebas cabe destacar la simulación de una red de ordenadores. Este ejercicio consiste en proponer a cada alumno cumplir el papel de algún elemento de red: un alumno hará de DNS, otro de cable, otro de router, otro de servidor... Una vez montada dicha red humana, se simula una petición de red y se calcula el tiempo de respuesta. Gana la prueba aquel equipo que menos tiempo tarda en servir la petición.

Los objetivos de este tipo de actividades son múltiples:

1.- Fomentar el trabajo en equipo

2.- Fomentar la competitividad de dicho equipo. Cada grupo de alumnos forman un equipo, que compite con las demás clases (tanto de su titulación como de las otras dos ofertadas). Este hecho facilita la cohesión en el aula y la comunicación entre alumnos de todas los planes de estudio de Informática, para potenciar sinergias entre conocimientos e intereses de futuros profesionales afines

3.- Aumentar la responsabilidad del alumno, ya que estas actividades se realizan fuera del horario reglado de clases

2.3 Competencias personales: lo que piden las empresas

Como parte del trabajo desarrollado por el Departamento de Calidad de la Universidad, se realizó un ejercicio que consistía en preguntar a empresas del sector lo que exigían a los titulados para contratarlos. Fruto de este ejercicio (realizado anualmente, desde hace ya tres años) se recogió la siguiente información:

1.- Los titulados en informática, en general, salen bien preparados técnicamente de las universidades.

2.- Se hecha en falta en los titulados informáticos el desarrollo de competencias personales muy útiles (e incluso imprescindibles, en la mayoría de los casos) para el cumplimiento de las funciones que se les encomiendan [7].

Se consideró que favorecer el desarrollo de dichas competencias era una necesidad ya desde los primeros cursos. Debido a la ya citada carga teórica de la mayor parte de las asignaturas de los primeros cursos, se estimó oportuno que la asignatura de *Introducción a la Informática en Red* era muy adecuada para experimentar técnicas encaminadas en dicha dirección.

Para cumplir con el objetivo de permitir el desarrollo de competencias en los alumnos, se

llevaron a cabo las actividades ya comentadas y se arbitraron dinámicas específicas (explicadas con detalle en este mismo artículo) tanto para la adquisición de conocimientos como para la evaluación de los mismos.

2.4 Temario

Un grave riesgo cuando se experimenta con este tipo de técnicas es la pérdida de rigor o profundidad en los contenidos. De hecho, para poder integrar este tipo de actividades donde el alumno adquiere un mayor protagonismo en su propio aprendizaje [3], conviene diseñar un temario lo suficientemente flexible y dinámico como para que se vaya enriqueciendo con las aportaciones de los propios alumno, sin perder en el camino los contenidos teóricos necesarios.

Parte del problema, aunque no todo, se resuelve con la división de la asignatura en las distintas fases. De hecho, la realización de un test inicial el primer día de clase, antes incluso de especificar el temario completo, permite al profesor ajustar el momento de inicio de cada módulo del temario a los conocimientos demostrados por los alumnos.

Así pues, de manera general, la asignatura tiene el siguiente temario:

Módulo A: Redes de ordenadores. En este módulo se pretende presentar la terminología básica de redes, topologías, tecnologías, etc.

Módulo B: Fundamentos de Internet. En este módulo se introduce la terminología y los conceptos fundamentales directamente relacionados con Internet, tanto desde el punto de vista técnico como de uso (historia, arquitectura cliente/servidor, protocolos, etc)

Módulo C: Servicios básicos de Internet. Se presentan los servicios básicos accesibles a través de Internet, así como los programas cliente necesarios para su utilización. (correo electrónico, FTP, telnet, IRC ...)

Módulo D: Servicios elaborados y de búsqueda. Se presentan someramente los diversos servicios (www, gopher, motores de búsqueda, índices y metaíndices, listas de distribución, news...)

Módulo E: Servicios Web de valor añadido. En este módulo, se hace hincapié en el uso de Internet *para algo* (buscadores, portales, mensajería instantánea, comunidades virtuales...)

Módulo F: Servicios a través de Internet. Se pretende presentar al alumno las posibilidades de Internet como elemento dinamizador y moldeador de la Sociedad (juegos on-line, servicios de traducción, diccionarios, teletrabajo y teleenseñanza, administraciones del Estado ...)

Durante la FASE I de la asignatura, el profesor presenta cada uno de los módulos para permitir a los alumnos posteriormente desarrollar por equipos cada uno de ellos en mayor profundidad.

3. Metodología de trabajo propuesta a los alumnos

Como se ha pretendido explicar en este artículo, el flujo de información en esta asignatura sufre un cambio importante, haciendo al alumno elemento clave en su propia adquisición de conocimiento. Esta modificación supone un cambio en el enfoque que el profesor tiene sobre su papel en el proceso de aprendizaje y en el reparto de tiempo de dedicación al mismo.

3.1 Apuntes de la asignatura

Los apuntes considerados como básicos por parte de los profesores (no hay que olvidar que son varios los implicados en la impartición de esta asignatura que pertenece a tres planes de estudio) deben estar disponibles para los alumnos desde la segunda semana de clase. Esto es importante para permitir al alumno que organice sus lecturas desde casi el principio del curso.

Estos apuntes no pueden estar disponibles desde el primer día porque conviene revisar primero los resultados del test inicial por si conviene ampliar o suprimir algún tema, dependiendo del punto de partida que el profesor estime conveniente.

Como parte de los apuntes de la asignatura, se incluyen aquellos trabajos de años anteriores que se estimen oportunos, cumpliendo este material una triple función:

1.-Demostrar al alumno la importancia que se le otorga al trabajo que deberá realizar, ya que puede formar parte de los contenidos ofrecidos a los alumnos venideros

2.- Mostrar mediante ejemplos qué se considera un buen trabajo. Para cumplir correctamente este objetivo, conviene poner a disposición de los alumnos más de un trabajo de años anteriores con diferentes características e intentar evitar así la aparición de clones en cuanto a estructuras de los mismos.

3.- Permitir al profesor una mayor flexibilidad sin necesidad de redactar a última hora nuevos contenidos.

3.2. Confección del trabajo

Los trabajos de los alumnos son la pieza clave tanto de la metodología docente como del sistema de evaluación y por lo tanto se pone especial énfasis en su realización. Las características más destacables de dichos trabajos son:

1.- Se realiza obligatoriamente en equipos compuestos por un mínimo de tres personas y un máximo de 5. De esta manera se potencia la necesidad de coordinación de esfuerzos entre los alumnos y su responsabilidad

2.- Cada grupo elige un tema distinto, todos relacionados con el temario. De esta manera cada grupo trabaja en una dirección distinta completando los apuntes dados al principio del curso en uno de sus aspectos que luego expondrá a toda la clase.

El proceso se considera tan importante como el producto. Para ello, el trabajo consta de cinco puntos, tres de ellos representan el *producto* de dicho trabajo (texto del trabajo, direcciones favoritas comentadas, direcciones visitadas), otro es la *herramienta de comunicación* (el guión de la presentación, generalmente realizada en un programa para presentaciones gráficas) y el último comenta y explica el *proceso* ("cómo se hizo", documento que recoge las búsquedas e impresiones sobre las mismas de los alumnos).

3.3. Ejecución de la presentación

Otro de los elementos clave para el desarrollo de las competencias de carácter personal en los alumnos de primero es la necesidad de exponer su trabajo a todo el grupo. Para ello es para lo que se les ha pedido, como parte de su trabajo, la realización de una presentación gráfica. Dicha presentación supone, en el momento de su redacción, un interesante ejercicio de síntesis ya que los trabajos suelen ser extensos debido a la gran cantidad de información que en Internet se encuentra sobre sí misma.

Una vez redactada dicha presentación, cada grupo dispone de media hora del horario de la asignatura para mostrarla a sus compañeros.

Al auditorio de estas charlas (los compañeros de clase de los ponentes) se les entrega un pequeño formulario donde pueden evaluar de manera anónima diversos aspectos de la presentación (claridad de la exposición, calidad de la presentación, calidad del material entregado para seguir la ponencia, interés despertado por la misma, etc.) dichas evaluaciones las recoge el profesor y, tras un vistazo a los resultados (y la correspondiente toma de las notas que estime oportunas) se las entrega al grupo de ponentes para su estudio.

Este feedback(saber qué opinan sus compañeros sobre la presentación que les acaban de hacer) es de indudable interés pedagógico y dota al alumno de un mayor conocimiento sobre qué aspectos debe reforzar en el futuro cuando se encuentre en situaciones análogas. De hecho, los comentarios recogidos por los profesores son altamente positivos, teniendo de paso un efecto secundario interesante: el alumno valora con conocimiento de causa el esfuerzo que supone la preparación de las clases por parte de sus profesores. Ni que decir tiene que algunos de los comentarios recibidos provocan una sonrisa en su profesor(creí que esto de dar clase era más fácil; los del fondo no me han hecho ningún caso). Esta *evaluación entre iguales* (rápida y anónima) ofrece al alumno una visión clara de sus puntos fuertes y débiles como orador, así como qué aspectos deberá cuidar más en el futuro para preparar una presentación pública. Después de cada exposición y a la vista de los comentarios hechos por sus compañeros de clase, los alumnos que acaban de exponer su trabajo inician un pequeño debate sobre el mismo.

3.4 Confección de batería de preguntas para el examen final

Con las acciones expuestas hasta este momento, se pretende involucrar al alumno de manera directa en su proceso de adquisición de conocimiento. No obstante, se juzgó interesante hacerle partícipe también en su proceso de evaluación...

Con este objetivo, se les propuso a los alumnos, como un ejercicio más de la asignatura la confección de un total de 30 preguntas de tipo test, con cuatro alternativas y una sola respuesta correcta, divididas en dos grupos: 20 preguntas relativas al temario de la asignatura y 10 preguntas más que versaran sobre su propio trabajo.

El compendio de todas las preguntas así conseguidas se sometió a un proceso de revisión

por parte del cuadro docente. Fruto del mismo, se consiguió una batería de más de 700 preguntas de tipo test válidas. De entre estas preguntas, cada profesor elegiría 30 preguntas que conformarían el examen final de la asignatura. Además, este fichero con todas las preguntas, se puso a disposición de los alumnos una semana antes de la realización del examen. Con esta acción se pretende dirigir el estudio del alumno a aquellos temas que se consideran relevantes, pero apoyándose en sus propias propuestas.

Con esta práctica se pretendía, además de hacer parte al alumno como co-autor de su propio examen final, reforzar varias habilidades útiles (responsabilidad, capacidad de síntesis y comprensión de textos largos, entre otras) para los alumnos, así como asegurar (en la medida de lo posible) una *segunda vuelta* al estudio de los apuntes de la asignatura.

4. Sistema de evaluación

El sistema de evaluación propuesto para esta asignatura tiene en cuenta, como es lógico, todas las actividades desarrolladas por los alumnos, aunque de manera ponderada.

Algunas de dichas actividades se realizaron en grupos, mientras que otras son de carácter individual. Esta dualidad provoca que el sistema de evaluación debe contemplar diversos aspectos.

El trabajo hecho en equipo y la presentación del mismo, son las piezas claves del sistema de evaluación de la asignatura, junto con el examen final. El resto de las actividades evaluadas (prácticas de búsqueda, confección de preguntas de tipo test, pruebas objetivas intermedias, asistencia y participación en clase) sirven para *personalizar* la nota de cada alumno.

Uno de los riesgos más comunes en un sistema de evaluación basado en actividades de equipos de alumnos (como es el caso), es la posibilidad de que un alumno supere la asignatura sin haber aprendido los contenidos mínimos necesarios, simplemente *parasitando* a sus compañeros de equipo.

Para minimizar este riesgo, se diseñaron una serie de prácticas y pruebas intermedias, de carácter obligatorio e individual, que obligaran al alumno a un seguimiento tanto del desarrollo de la asignatura, como del trabajo escogido `por él y su equipo.

Como ejemplo, durante la Fase I de la asignatura, todas las sesiones debían terminar con un ejercicio de búsqueda de las palabras clave de la explicación del profesor. El resultado de dicha búsqueda debía enviarse por correo electrónico a la dirección del profesor. Dichas definiciones, junto con el resto de la materia explicada en clase, formaba parte del contenido de la asignatura y complementaba los apuntes entregados en las primeras clases del curso.

Con las notas así conseguidas (trabajos en grupo, prácticas individuales, exámenes), cada profesor calcula la nota final, bajo su propio criterio, pero respetando los siguientes principios:

1.- El trabajo es el eje central de la evaluación

2.- La entrega de todas las prácticas es obligatoria

3.- La exposición en clase del trabajo, así como las preguntas redactadas por cada uno, aporta información sobre la participación de cada alumno en su grupo de trabajo.

5. Metodología de trabajo de los profesores

Uno de los principales problemas de una asignatura de estas características es la necesidad de coordinación entre todos los profesores implicados. Este problema, común a todas las asignaturas que tienen más de un profesor, se ve agravado por los cambios introducidos en el flujo de información docente-alumno. Como, además, dichos cambios tiene un evidente carácter experimental, convenía automatizar al máximo dichos procedimientos (coordinación y transmisión de información) para facilitar la exportación de métodos y herramientas a otras asignaturas.

Para todo ello, se ha diseñado un documento Web propio de la asignatura [1] que sirve como portal de entrada tanto a profesores como a alumnos a la documentación asociada, y como principal medio de comunicación entre los alumnos y sus profesores.

Además, existe un servicio de FTP anónimo donde los profesores de la asignatura dejan todo el material que estiman oportuno y los alumnos pueden recoger con facilidad

No obstante, existen cinco grupos distintos (y por lo tanto cinco profesores) que, además de la información común a todos ellos, pueden requerir información específica (cada grupo de alumnos debe poder acceder a los contenidos de los trabajos que se presentaron en su clase), por lo que además de un directorio en el FTP anónimo

de la asignatura para cada uno de los grupos, se creó otro directorio, con acceso restringido a los profesores de la asignatura, donde poder hospedar toda la información común, permitiendo a cada profesor colgar su material específico en el directorio de su grupo y en este otro (llamado coordinación) aquellos documentos o referencias de carácter común.

Para facilitar la publicación de este material común, se desarrollaron dos sencillos programas shell-script que permiten publicar de una sola vez en todos los directorios un archivo alojado en coordinación, mientras que el otro elimina los enlaces de dichos directorios, dejando el archivo original accesible para los profesores para su posterior uso.

Como mecanismo de coordinación entre los profesores se creó una lista de distribución, llamada convenientemente iir, y se establecieron reuniones periódicas para compartir información y experiencias.

6. Evaluación de resultados

Debido al carácter experimental del método, los profesores de la asignatura pidieron ayuda al departamento de Calidad de la universidad. El objetivo era realizar preguntas específicas, en los cuestionarios de satisfacción que anualmente se reparten entre todos los alumnos de la universidad, sobre el sistema de evaluación y la percepción sobre la cantidad y calidad de aprendizaje conseguido.

Los datos obtenidos parecen indicar que los alumnos de Introducción a la Informática en Red se sienten satisfechos con la cantidad aprendizaje (3.72 de 5) y la calidad del mismo (3.89). En cuanto al sistema de calificación empleado los resultados también parecen expresar la conformidad de los alumnos (4.00)

Por último, el índice de aprobados es muy elevado (87% de entre los presentados) mientras que el de abandonos es sensiblemente inferior al de otras asignaturas del mismo curso (8% frente a un 13%,como media en primer curso).

7. Conclusiones y trabajos futuros

Entre las actividades que se plantea desarrollar para el próximo curso se encuentra un generador aleatorio de test que los alumnos pueden utilizar siempre que deseen. De este modo conseguimos dos cosas, por una parte aprovechar la numerosa colección de preguntas tipo test que hemos ido recopilando y por otra motivar al alumno a repasar los contenidos teóricos de la asignatura.

Referencias

[1] http://www.esi.uem.es/asignaturas/IIR.htm *Documento Web de la asignatura*

[2] *http://www.esi.uem.es Documento Web de la Escuela Superior de Informática de la UEM*

[3] The Bologna declaration on the European space for higher education: an explanation. http://europa.eu.int/comm/education/socrates/erasmus/bologna.pdf

[4] Brandt Scott, Constructivism: Teaching for Understanding of the Internet. Communications of the ACM. Octubre 1997.

[5] Bricall , Joseph M. *Universidad 2 mil*. Conferencia de Rectores de las Universidades Españolas(CRUE). 2000.

[6] Docampo Amoedo, Domingo, et all. *El crédito Europeo y el sistema educativo Español*. http://www.crue.org/espaeuro/encuentros/credito-vigo2002_.pdf

[7] Iñigo Alvarez, et als. *Plan de desarrollo de competencias en el alumnado de la Universidad Europea – CEES*, documento interno, 2002.

[8] Pedro José Lara et als. *Nuevas técnicas de aprendizaje: Cybergymkhana*, JENUI 2002

[9] Peter J. Denning and Robert Dunham.The Profession of IT COMMUNICATIONS OF THE ACM November 2001/Vol.44, No.11.

[10] Haugh, G. y Kirstein, J. Trends in Learning Structures in Higher Education. http://www.rektorkollegiet.dk/sider/publikationer/english/edutrends.htm

Estudio de un Sistema de Aula Virtual como Apoyo a la Docencia Presencial

María del Pilar Romay Rodríguez
Departamento de Programación e Ingeniería de Software
Escuela Superior de Informática
Universidad Europea de Madrid
e-mail: pilar@dpris.esi.uem.es

Carlos E. Cuesta Quintero
Departamento de Informática (ATC, CCIA, LSI)
Escuela Técnica Superior de Ingeniería Informática
Universidad de Valladolid
e-mail: cecuesta@infor.uva.es

Resumen

En esta ponencia se expone el planteamiento y diseño de un Sistema de Aula Virtual, que parte de técnicas y herramientas empleadas en el campo de la Educación a Distancia, con el objetivo de potenciar y apoyar a la Docencia Presencial.

Se detallan las motivaciones pedagógicas en las que se fundamenta esta propuesta, y se esboza la arquitectura interna y estructura lógica del Sistema desarrollado. A continuación, se discuten los distintos aspectos relativos a la aplicación del mismo, incidiendo especialmente en los puntos críticos que van a potenciar su aceptación dentro del proceso educativo presencial tradicional. Se concluye extrayendo algunas de las consecuencias de las primeras experiencias realizadas.

1. Introducción

La enseñanza de asignaturas de Informática, y en especial de las de tipo introductorio, muestra una clara problemática en el entorno actual. Por una parte, se trata de una disciplina cada vez más común en todas las titulaciones técnicas, y no sólo las específicamente informáticas. Sin embargo, no siempre cumple las expectativas del alumnado, en especial en una sociedad en la que los ordenadores son ya un elemento doméstico común.

Por otra parte, estas asignaturas resultan con frecuencia difíciles de coordinar, debido al gran número de docentes, alumnos, grupos y horarios que se implican en ellas, en especial si se consideran las sesiones de prácticas, y además en múltiples entornos y titulaciones.

En esta ponencia se propone la combinación de técnicas y utilidades propias de la Educación a Distancia con las de la Educación Presencial. Esto permite motivar al alumno con una perspectiva diferente, sin necesidad de reducir en absoluto el componente teórico. Esto se considera útil para mejorar tanto el modo de aprender del alumno, como la realimentación recibida por el profesor. De este modo, se enriquece el proceso global de enseñanza-aprendizaje.

Esta propuesta se concreta en la definición de un sistema de *Aula Virtual*, concebido no como reemplazo, sino como apoyo a la clase presencial convencional. Se incide aquí sobre los aspectos que han de ser controlados para poder potenciar el uso de este tipo de sistemas en este contexto.

Los puntos esenciales sobre los que se fundamenta el desarrollo del Aula Virtual son el cambio en los roles de los agentes que participan en el proceso educativo, la necesidad de coordinar el proceso de enseñanza-aprendizaje, y de potenciar una enseñanza menos utilitaria; así como la posibilidad de tener un mecanismo de realimentación que indique cómo se está llevando a cabo el proceso educativo.

Un punto clave para el éxito de esta iniciativa es un adecuado mantenimiento de la información contenida en el sistema. Con este fin se propone un mecanismo de delegación de su gestión, con un supervisor final, pero que implique por igual a alumnos y profesores. Al tiempo que distribuye el trabajo, esto permitirá potenciar la interacción entre profesores y alumnos, así como mejorar la coordinación entre distintos profesores, e incluso entre los propios alumnos.

2. Motivación Pedagógica

La distinción entre los términos de *enseñanza* y *aprendizaje* a la que se hace referencia en un sin fin de artículos dedicados al tema de tecnología educativa, revela la importancia que se le concede en la actualidad al *aprendizaje* dentro del proceso educativo. De hecho, ha alcanzado una importancia tal, que en ocasiones el papel de la *enseñanza* queda relegado. Se insiste con frecuencia en buscar mecanismos de aprendizaje más efectivos, olvidando que la enseñanza y el aprendizaje han de ser procesos coordinados. En este punto es donde va a incidir el sistema propuesto, cubriendo aspectos de enseñanza, pero que facilitan el aprendizaje del alumno.

Como se confirma en el Informe Delors [1], en el proceso de enseñanza-aprendizaje los roles del profesor y el alumno han cambiado en gran medida. El sistema propuesto pretende potenciar algunos de estos "roles para la Universidad del Siglo XXI", de los que se ha hablado; algunos de los relativos al profesor son los siguientes:

- Orientador, diseñador y motivador
- Transmisor de información y experiencia
- Organizador y evaluador de información
- Aprendiz, junto con los estudiantes
- Investigador de la práctica docente
- Colaborador con otros docentes

Otro aspecto sobre el que incide el sistema que se propone es el manejo de la información de la que se hace uso en un proceso educativo. La sociedad actual se caracteriza por la gran cantidad de información que se manipula, y que ha de ser procesada para dar lugar a un conocimiento aplicable en la resolución de problemas cada vez más complejos, que en muchos casos requieren de la integración de múltiples disciplinas. El mantener puntos en donde la información es organizada, revisada y valorada supone un paso sustancial en el proceso de enseñanza-aprendizaje. Más, si cabe, en áreas donde la información que se transmite proviene de una amplia diversidad de fuentes, como es el caso de la informática.

Con esta herramienta se quiere potenciar el proceso de aprendizaje, a través de tecnologías interactivas, transmisivas y colaborativas. Se pretende con este sistema que el alumno sea partícipe de su proceso educativo, y el profesor potencie a su vez su método de enseñanza, al disponer de otro mecanismo de realimentación, que le permita evaluar, tanto a sí mismo, como el aprendizaje de sus alumnos. Esto da pie a la posibilidad de elaborar documentación personal sobre la experiencia de su práctica docente; todo ello puede facilitar el desarrollo de elementos como la *carpeta* o *portafolio docente*, asociados habitualmente a la docencia de calidad.

Se ha de tener en cuenta, además, que un sistema de este tipo permite incluso evaluar una parte del *trabajo personal del alumno* durante el proceso de aprendizaje. Éste es un aspecto clave, usualmente ignorado, pero que adquiere cada vez mayor importancia. Esto puede verse, por ejemplo, en la definición de los créditos ECTS en el Espacio Europeo de Educación Superior.

3. Solución Propuesta: Uso de un Sistema de Aula Virtual

El sistema que se propone pretende incidir tanto en el proceso de enseñanza como en el de aprendizaje, permitiendo al alumno llevar un ritmo que se adecúe a sus circunstancias, haciendo que se responsabilice de su educación, no de forma aislada sino cooperativa. La idea básica es utilizar técnicas propias de la Educación a Distancia como apoyo a la enseñanza presencial. En absoluto se pretende sustituir a esta última, sino complementarla.

Con este fin, se parte de la metáfora de un *Aula Virtual*, un sistema informático en el que se pretende simular a través de la red la interacción y dinámica de una clase presencial. En este caso, este sistema no es una alternativa a, sino una extensión de la clase tradicional. El Aula Virtual permite ampliar el ámbito de trabajo; la clase puede continuar su relación fuera del aula, tanto temporal como geográficamente.

3.1. Arquitectura del Sistema de Aula Virtual

El sistema de Aula Virtual utilizado [4] se diseñó en torno a cuatro objetivos globales, a saber: aplicabilidad, extensibilidad, flexibilidad y portabilidad. Esto es, se pretendía disponer de una plataforma que fuese sencilla de utilizar, fácil de ampliar y adaptar a un contexto

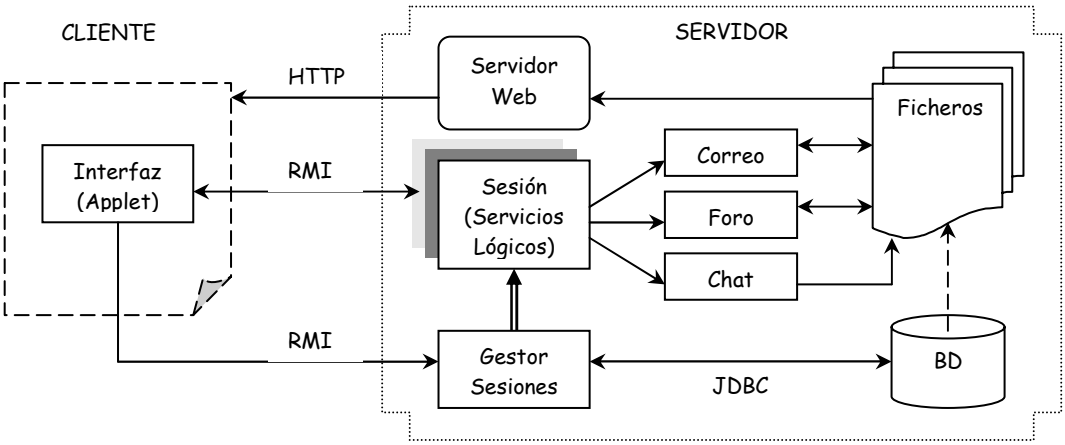


Figura 1. Arquitectura del Sistema de Aula Virtual

concreto, e independiente, en lo posible, de un entorno *software* o *hardware* determinado.

Por otra parte, un entorno de este tipo puede diseñarse, bien como un sistema cerrado, o bien como un sistema abierto a la red. En este caso, se ha optado por lo primero, con el fin de lograr un vínculo más fuerte entre la comunidad a la que va dirigido, y evitar al mismo tiempo el uso de la plataforma con fines diferentes. Esto no significa que el sistema deba ser excesivamente rígido; por el contrario, dentro de los límites establecidos ha de ser totalmente libre.

En la Figura 1 se muestra gráficamente la arquitectura del sistema de Aula Virtual utilizado, delineando su esquema global e identificando sus componentes principales. La arquitectura se enmarca claramente en un estilo Cliente/Servidor. Esto no se debe únicamente a la infraestructura tecnológica utilizada, sino que de hecho se deriva de modo natural de la metáfora en la que se basa el sistema.

En efecto, el Aula Virtual es un "espacio de reunión" común, un punto de encuentro entre sus usuarios, los actores del proceso educativo. En la arquitectura descrita, el Aula se identifica con el Servidor, lo que permite concebirlo como un lugar concreto, con una ubicación única.

En cambio, el componente Cliente se concibe simplemente como una "puerta de entrada", que provee a los usuarios con los mínimos recursos necesarios para permitir su integración en el Aula Virtual. Se trata de la *interfaz de usuario*, una simple máscara a través de la cual los actores pueden participar en la dinámica del Aula de manera controlada.

Así pues, se ha optado por implementar el componente como un *applet* Java. Esto permite tener toda la interfaz en un bloque único, que se distribuye con facilidad. El usuario tan sólo necesita acceder a una dirección dada, desde la que se descarga el *applet*.

El componente Servidor es, por tanto, el que encapsula toda la lógica del sistema. Se concibe como un conjunto de servicios y módulos, que se interconectan, bien localmente o de manera distribuida: esto no afecta al esquema global.

El punto de entrada al Aula Virtual es el Gestor de Sesiones, ubicado en una dirección bien conocida. Un proceso Cliente empieza su actividad con una conexión a este componente. Se realiza una fase de autenticación, en la que el Gestor comprueba las credenciales del Cliente con la Base de Datos. Si esto es satisfactorio, se *crea* un proceso Sesión específico para este Cliente, y se establece entre ambos una conexión RMI privada. Sobre ésta se realiza toda la interacción posterior.

El proceso Sesión, creado con unos privilegios determinados por el usuario, oferta a su Cliente una serie de servicios –*servicios lógicos*–, que son los que de hecho constituyen el Aula Virtual. A su vez, éstos se basan en otros módulos, internos al sistema, los *servicios físicos*. Éstos son los que en realidad acceden a la información. Consisten, en resumen, en un servicio de *Correo* o mensajería entre los

usuarios; un servicio de *Foro*, encargado de publicar información de forma interna al Aula; y el servicio de *Chat*, que soporta la comunicación interactiva entre dos o más usuarios.

Es evidente que la arquitectura descrita no es la única posibilidad, ni siquiera partiendo de un esquema basado en web. En la forma actual existen varios problemas de tipo técnico; entre éstos destacan los relativos a las restricciones de seguridad de los *applets*. Para superar estos problemas, se ha planteado una modificación de la estructura global, de cara a la elaboración de la próxima versión del sistema; sin embargo, esto no afectará al esquema general expuesto.

3.2. Servicios Lógicos y Servicios Físicos

El sistema que se ha desarrollado suministra un mecanismo de disposición y transmisión de información, a través de lo que se denominan sus *servicios*. Éstos identifican los recursos sobre los que se lleva a cabo la comunicación entre los actores del proceso educativo.

Servicios	Correo	Foro	Chat
Tablón anuncios		√	
Noticias		√	
Mensajería			
General	√		
Grupo	√		
Sala de reunión			
General			√
Privada			√
Grupo			√
Material docente		√	
Exámenes		√	
Documentos de interés		√	
Opinión y Sugerencias	√	√	
Tutorías			
Interactiva			√
Diferida	√		
Interactiva en grupo			√
Persistente		√	√
Seminarios			
Interactivo			√
Persistente		√	√
Directorio		√	
Opinión Aplicación	√	√	

Tabla 1. Relación entre servicios físicos y lógicos

Se distinguen dos tipos de servicios*: servicios lógicos* y *servicios físicos*. Los segundos son el pilar fundamental sobre el que se asientan los primeros. Los servicios lógicos se presentan al usuario como recursos tangibles en la transmisión y difusión de la información. Los servicios físicos afectan a la extensibilidad del *aula virtual,* al permitir una diversidad de servicios lógicos.

Las diferentes disposiciones de los servicios lógicos se presentan a los usuarios en función de las necesidades puntuales de una materia determinada y de la motivación que suponen tanto para el alumno como para el docente.

En la Tabla 1 se indica sobre qué servicios físicos se implementa una relación de servicios lógicos disponibles en el Aula.

Los servicios lógicos se interrelacionan entre sí a través de la información que manejan internamente, y en ocasiones incluso de manera explícita a través de referencias internas.

3.3. Acceso a los Recursos del Sistema

En un sistema de este tipo, sometido idealmente a una actividad considerable, la importancia de los detalles relativos a seguridad resulta indiscutible. Es preciso que el mecanismo de control de acceso sea suficientemente sólido: la aparición frecuente o espontánea de anomalías en este aspecto va directamente en contra de la reputación global del sistema, haciendo disminuir su popularidad ante los usuarios potenciales. Existe además entre ellos cierta inclinación natural a buscar y explotar este tipo de problemas, lo que hace que este punto sea particularmente comprometido.

Como contrapartida, un control extremo puede influir negativamente en la motivación del estudiante, haciéndole sentir que el Aula es más un mecanismo de vigilancia que un entorno abierto al que puede acudir libremente en su proceso de aprendizaje

Ha de tenerse en cuenta que, al trabajar dentro de un sistema cerrado, que sólo proporciona acceso a sus propios recursos, no

Ventajas	Puntos Conflictivos
Mecanismos para centralizar la información	Desfase en la actualización de la información
Disponibilidad para ser consultada en cualquier momento	Desmotivación en la utilización de la herramienta, tanto con fines de consulta como con fines de participación
El alumno puede ampliar conocimientos de forma guiada	Disponibilidad de acceso doméstico a Internet
Guía para los alumnos que van desfasados durante el curso	Susceptibilidades respecto a la identificación del alumno
Facilidad de comunicación entre los miembros de un grupo	Temporización con respecto a la clase presencial
Coordinación de la enseñanza entre profesores	Posible descenso en la asistencia al aula presencial
Enriquecimiento tanto para el profesor como para el alumno	Excesiva abundancia de información
Mejor aprovechamiento de las clases presenciales	Control de la calidad de la información
Permite abrir líneas de debate	Determinar el nivel de control adecuado
Mecanismo de publicación de experiencias	Establecer mecanismos adicionales de valoración y evaluación sobre la participación y el aprendizaje
Potencia la calidad docente	Tiempo dedicado a la preparación y revisión del material e información que se maneja
Fomenta el aprendizaje colaborativo y cooperativo	Tiempo para poder procesar toda la información que se presenta en los recursos, tanto por parte del alumno como por parte del profesor
Mecanismo de información de noticias sobre la asignatura	No debe limitar la interacción personal con los alumnos
Presenta la información enriquecida por varios puntos de vista	
Sistema alternativo de interacción en clases prácticas	
Permite realizar tutorías no presenciales	

Tabla 2. Ventajas y Puntos Conflictivos en el uso del Sistema de Aula Virtual

son necesarias técnicas avanzadas de seguridad y autenticación, basadas en complejos métodos criptográficos. Sólo se trata de tener cierto control sobre el acceso a los servicios lógicos, y a la información que se localiza en éstos.

El mecanismo de seguridad ha de ser versátil y flexible; se ha optado por un sistema basado en la clásica noción de *posibilidades* [5], que ha demostrado su especial adecuación para entornos distribuidos. Cuando un usuario entra en el sistema, se le adjunta un *ticket* o conjunto de posibilidades, que indica a qué recursos tiene acceso en el momento actual. Este conjunto puede ser alterado dinámicamente, bien para ampliarlo con nuevas posibilidades, bien para revocar alguno de los permisos existentes.

Este mecanismo también permite guiar al alumno en su evolución, actuando sobre su motivación. Se le puede "premiar" con distintos grados de acceso según el grado de participación e interés mostrado. Esto fomenta la cooperación y competitividad entre los alumnos.

4. Aplicación de la Herramienta

El interés en el desarrollo del Aula Virtual se centra en el estudio de los *aspectos críticos* que van a permitir que el sistema cumpla las expectativas pedagógicas previstas.

La clave es su nivel de aceptación, tanto por parte de los profesores como de los alumnos. Se ha realizado un estudio que contempla posibles *puntos conflictivos* que influyen negativamente sobre la aceptación en este tipo de sistemas. Este estudio se ha basado en conclusiones obtenidas en experiencias de Educación a Distancia y en Educación con Tecnologías de la Información y la Comunicación, principalmente en trabajos de la IFIP [1][6]. Otra fuente de información han sido los resultados que se han obtenido en un entorno de pruebas restringido, implicando a alumnos de Proyectos Fin de Carrera.

En la Tabla 2 se da una relación tanto de las ventajas como de los puntos conflictivos obtenidos como resultado de este estudio.

4.1. Organización de la Información

Uno de los aspectos críticos es la información de la que se hace uso. Su organización se va a centrar en la estructura de la información persistente y el proceso de filtrado.

La información que se maneja dentro de este Sistema de Aula Virtual incluye:

- La identificación, tanto de los servicios físicos como de los servicios lógicos
- Los propios recursos, esto es, el contenido de los servicios lógicos: conversaciones, documentos, mensajes y enlaces.
- La caracterización de los recursos

La información que se presenta en cada servicio está gobernada por pares atributo-valor que caracterizan las propiedades del servicio, a los que se denomina *esquemas de información*. En nuestro desarrollo actual se está procediendo a la reelaboración de estos esquemas con la tecnología XML, lo que va a permitir lograr una mayor flexibilidad. En definitiva, se proponen tres esquemas de información:

- Esquema para los servicios físicos
- Esquema para los servicios lógicos
- Esquema para caracterizar los recursos

Al considerar un esquema diferente para cada servicio lógico se hace posible destacar las diferencias entre ellos, incluso cuando se basan en los mismos servicios físicos.

Así por ejemplo, para el servicio de *tablón de anuncios* resulta primordial tener identificada la fecha de publicación de la noticia y fecha de validez de la misma, mientras que para el contenido del servicio de *foros* no tiene sentido la propiedad fecha de validez, aunque sí la fecha de publicación.

La inclusión de estas propiedades incide sobre los puntos críticos que se presentaban en la Tabla 2. En el ejemplo antes citado, un manejo inadecuado de las fechas del ejemplo incidiría sobre aspectos como la actualización de la información, y la motivación en la utilización del Aula Virtual.

La información va estar organizada dentro de cada servicio lógico. La estructura que va a permitir organizar los recursos es de tipo jerárquico. Se considera óptimo el número de dos niveles, si bien en algunos servicios físicos puede ser de utilidad considerar tres. Tal es el caso de los recursos asignados a los servicios lógicos *material docente* y *documentos de interés*. Así, conviene que los recursos del primero sean organizados según el esquema del plan docente de la asignatura. Esto permite una mejor coordinación con la evolución de la clase presencial y evita desorientar al alumno.

La organización básica que se propone considera tres niveles: unidad temática, tema y apartado. Esta estructura de organización es válida para otros servicios, aunque en éstos la organización no van a tener por qué coincidir con la distribución que se hace en la asignatura. Es el caso de las noticias, seminarios, tutorías persistentes, documentos de interés, tablón de anuncios y exámenes. Sin embargo, resulta interesante que en el tengan cabida focos de interés situados fuera de lo estrictamente programado para la asignatura.

Esta información de organización, al igual que la caracterización de información va a estar recogida en una representación atributo-valor. Cada recurso perteneciente a un servicio lógico va a contemplar, además de la información que la caracteriza, información sobre la estructura en la que está ubicado. Esto va a permitir al administrador una mayor flexibilidad en la modificación de la estructura.

Dentro del Sistema de Aula Virtual se va a permitir al alumno la incorporación de material al sistema. Es por ello por lo que se considera necesario evaluar el contenido de la misma para filtrar el material inadecuado. En el caso de los recursos que tienen como base el servicio físico *Foro*, este tipo de control va a permitir certificar los contenidos. Esta operación sólo podrá ser llevada a cabo por el administrador.

El filtrado de la información también va a estar ligado al mecanismo de control de acceso. Así, se podría gestionar la disponibilidad de los recursos con respecto a la clase presencial, controlando el acceso a los mismos. Del mismo modo, van a ser filtrados con relación al nivel de dificultad del contenido que presentan. En resumen, asignando *tickets* a grupos de alumnos con perfiles similares, se puede filtrar la información de modo que tenga el nivel apropiado a un grupo concreto.

Administrador	Moderador	Usuario
Características Funcionales Generales		
Control Total Adjudicado a Profesores Papel Activo Certificador de calidad del material Aprueba y convoca los seminarios Gestor único del Servicio de Noticias	Control del Recurso Moderado Adjudicado a ambos actores Papel Activo Evaluador de material Elabora y propone seminarios vinculados al tema sobre el que modera	Control restringido Adjudicado a Estudiantes Papel Activo o Pasivo No evalúa material, pero puede opinar Emisor de material sin certificar Propone y solicita seminarios
Políticas Globales Ligadas a la Función		
Políticas de Coordinación Temporización Gestión de Posibilidades Promoción de Moderadores	Políticas de Coordinación e Incentivación Políticas de Adjudicación de Recursos Temporización Gestión de Posibilidades	Limitados por la Definición de Perfiles Limitados por el Control de Seguridad

Tabla 3. Tipos de Usuario en el Sistema de Aula Virtual

4.2. Tipos y Perfiles de Usuario

En el Sistema de Aula Virtual se distinguen tres tipo de usuarios: administrador, moderador y usuario normal. En la Tabla 3 se expone una relación de las funciones asignadas a cada uno de ellos, así como de las políticas que permiten realizar un buen uso de la herramienta.

El administrador es el tipo de usuario que tiene control sobre todo el sistema. Es interesante que este trabajo esté coordinado por más de un profesor; si no, sería probable que la gran cantidad de aspectos a contemplar diera lugar a un abandono del sistema.

El interés por parte de algunos alumnos en ciertos temas puede fomentar una dinámica que haga útil considerar una nueva figura, la del moderador. La figura del moderador siempre va a estar controlada por el administrador.

Este papel puede ser elegido por el alumno o asignado a alumnos en función de su interés y su nivel de conocimientos.

La figura de usuario normal no implica un papel pasivo, aunque se le permite un uso más restringido de los recursos. Este usuario puede dar opiniones sobre los recursos, añadir nuevo material, o hacer propuestas que serán recogidas y valoradas por los moderadores. De hecho, su participación puede hacer que sea promovido al nivel de moderador.

Otro aspecto son los perfiles de usuario, que surge para poder caracterizar diferentes grupos con relación a las *posibilidades* que se les quiera asignar. Los criterios para la realización de esta caracterización dependen de cada entorno específico: una correcta actuación, su nivel de participación, su curso actual, o el nivel de conocimientos, entre otros muchos.

4.3. Realimentación

Este Sistema de Aula Virtual puede usarse como un mecanismo de realimentación del proceso de aprendizaje llevado a cabo dentro del aula. Permite tomar el pulso general a la clase, con lo que pueden tomarse decisiones más adecuadas en el planteamiento de las clases presenciales.

La realimentación se obtiene en una medida importante a partir de los servicios de opinión. A este tipo de servicios se puede acudir como usuario anónimo, por lo que las posibilidades de obtener una opinión sincera aumentan.

Otros puntos a partir de los que se puede obtener realimentación en el sistema son el nivel de actividad, el tipo de material incorporado, el tipo de preguntas en las tutorías, las descargas de documentos, o el número de accesos.

5. Trabajo Relacionado

El trabajo descrito en esta ponencia se relaciona con muchos de los estudios ya existentes en el contexto de la Educación a Distancia. En particular, el concepto de Aula Virtual, así como sus extensiones [1], no es algo novedoso. Existe, de hecho, cierto número de herramientas comerciales [3][7][8] basadas en la misma idea, pero siempre concebidas como una alternativa a la docencia presencial.

La idea de combinar estas técnicas con la docencia presencial no es tampoco desconocida; de hecho, no es extraño encontrar iniciativas de esta índole basadas en algún tipo de tecnología web, tanto en universidades españolas como extranjeras. No obstante, estas experiencias se realizan habitualmente de manera ocasional o informal. La novedad de esta propuesta está precisamente en realizar estas tareas de manera *sistemática* y con un objetivo, enmarcándolas en un proceso de enseñanza-aprendizaje complejo, y proporcionando la infraestructura adecuada.

6. Conclusiones y Trabajo Futuro

Aunque hasta el momento sólo se han realizado pruebas preliminares, se pueden extraer algunas conclusiones iniciales a partir de las mismas. En principio, el uso de este sistema, en un entorno suficientemente motivado, se manifiesta como algo muy positivo, con una buena acogida por parte de los estudiantes. La organización de la información se revela como un aspecto clave, y es ésta la que ha de guiar en el proceso al alumno, cuyo interés personal resulta sin duda un aspecto crítico.

De cara a una prueba en un contexto real, se han de determinar las diferencias a la hora de aplicar esta idea en un entorno puramente informático –titulaciones propias– o en otras titulaciones técnicas –ingeniería, etc.–. Se han realizado pruebas iniciales en ambos entornos, sobre asignaturas similares, con acogidas muy diversas; pero aún es pronto para confirmar una tendencia clara basada en este aspecto.

El auténtico alcance del sistema sólo podrá ser determinado a partir de futuros desarrollos en esta misma línea, tanto en los aspectos técnicos –evolución de la herramienta– como en los sociológicos.

Con respecto a lo primero, en el momento actual se está elaborando una segunda versión de la plataforma existente, con el fin de ampliar sus capacidades y superar algunas de sus limitaciones. En esta versión, el esquema global se mantiene, pero la arquitectura muestra algunas diferencias significativas. Por ejemplo, se elimina el *applet* en el extremo Cliente, derivando hacia un sistema puramente basado en web; la lógica de la interfaz de usuario se traslada al extremo Servidor, usando para ello un mecanismo de *servlets*.

Por otra parte, la nueva versión incorporará toda una serie de nuevas funciones, que refinan y amplían las ya existentes. Se prevé la inclusión de módulos capaces de analizar y manipular las estadísticas de acceso y uso del sistema de manera semi-automática. Parte de los datos obtenidos de este modo serán utilizados para alimentar un plan de métricas, basadas en algunos de los estudios existentes sobre entornos de Educación a Distancia, y que a su vez podrán ser utilizadas para realimentar y ampliar a la propia herramienta.

En lo que respecta a los aspectos sociales del experimento, se continúa realizando distintas pruebas del sistema. En un futuro cercano, se ha previsto utilizar el método descrito de manera más sistemática y continuada, con grupos de usuarios más numerosos o menos motivados, e incluso se plantea la posibilidad de reunir múltiples entornos en un mismo Aula. En este sentido, las capacidades estadísticas y métricas que van a incluirse en la nueva versión resultan particularmente prometedoras.

Referencias

[1] Davies, G., Levrat, B., Ebey, T. *Virtual Universities and Virtual Campuses*. En [6].

[2] Delors, J. *La Educación encierra un tesoro*. Informe de la Comisión Internacional sobre la Educación para el Siglo XXI, Santillana Ediciones UNESCO, Madrid, 1996.

[3] Lotus LearningSpace & Virtual Classroom, http://www.lotus.com/products/learnspace

[4] Rodríguez Herrera, R., Giménez Jiménez, J. *Análisis, diseño y prototipo de una plataforma de servicios para el apoyo a la*

docencia: aula virtual Alpha. Proyecto Fin de Carrera, Univ. de Valladolid, 2002.

[5] Tanenbaum, A. S. *Sistemas Operativos Modernos*. Prentice-Hall, 1997.

[6] Watson, D., Andersen, J., eds. *Networking the Learner: Computers in Education*. Kluwer Academic Publishers, 2002.

[7] WBT Systems TopClass Virtual Classroom, http://www.wbtsystem.com/products

[8] WebCT. http://www.webct.com

Los concursos de programación como herramienta didáctica

Agustín Cernuda del Río Daniel Gayo Avello

Departamento de Informática
Escuela Universitaria de Ingeniería Técnica en Informática de Oviedo
(Universidad de Oviedo)
C/ Calvo Sotelo, S/N – 33007 Oviedo
Tf.: 985 10 50 94 - Correo-e: {guti, dani}@lsi.uniovi.es

Resumen

Algunos docentes aprecian un creciente desinterés del alumnado hacia las materias específicas de Informática, y en particular de la programación. A fin de promover y valorar la afición a estas materias, los concursos de programación entre estudiantes pueden ser una buena opción.

A la hora de organizar estas actividades puede ser conveniente plantearse los diversos pros y contras de las posibles modalidades, o conocer experiencias realizadas en otros centros.

1. Introducción

La evolución de la Informática como disciplina laboral en los últimos años ha transformado su imagen ante la sociedad. Hay que recordar que hace dos décadas casi no se conocía el ordenador como instrumento de uso doméstico. En la actualidad, aunque aún falte un gran trecho para que la situación de la ingeniería informática se normalice, es conocido al menos que hay una notable demanda de profesionales [11].

Esto ha transformado el perfil de los estudiantes. Hace años era muy frecuente que el alumno de Informática fuese aficionado a la materia, su familia no entendiese bien a qué se dedicaba y sus problemas académicos estuviesen centrados en asignaturas supuestamente *no informáticas*. Hoy en día, sin embargo, suele darse el caso de alumnos que se matriculan movidos por las salidas laborales, espoleados por su familia, y que de hecho completan el ciclo de estudios teniendo aún pendientes diversas asignaturas de programación. Ese parece ser el caso, al menos, en la Escuela Universitaria de Ingeniería Técnica en Informática de Oviedo.

Ante este panorama parece oportuno promover la afición a la informática o el ejercicio lúdico de la misma, en especial en facetas relacionadas con la programación. Los concursos de programación son un recurso conocido para ello (en [14] puede verse incluso un enfoque para promocionar el estudio de la teoría y los métodos formales mediante concursos de programación), pero su organización puede plantear ciertas dificultades.

En este artículo se examinan desde diversas perspectivas los concursos de programación. A continuación se ofrecen diversas ideas sobre posibles modelos de concurso y posteriormente se describe una experiencia concreta llevada a cabo en la Escuela Universitaria de Ingeniería Técnica en Informática de Oviedo (EUITIO).

2. Ventajas e inconvenientes de los concursos de programación

¿Qué interés puede tener desde un punto de vista didáctico la organización de un concurso de programación? Cabe perseguir los fines siguientes:

- La mera publicidad del concurso permite llamar la atención de todo el alumnado (participantes y no participantes) y recordarles la programación y la relación directa con los ordenadores.
- El que se organicen estos actos también extiende la idea de que la programación no sólo es un aspecto importante en su profesión, sino que también puede ser algo divertido. Efectivamente, existe una suerte de *vocación* informática.
- Centrándonos en los alumnos participantes, el concurso no deja de ser un ejercicio más, que les obliga a plantearse retos y buscar soluciones. Dependiendo del tipo de concurso se puede promover incluso una preparación específica en algún lenguaje, técnica o familia de algoritmos.

Sin embargo, no conviene perder de vista los inconvenientes asociados a estos concursos:

- Cierto tipo de concursos puede intimidar a los alumnos e inhibir la participación.
- Paradójicamente, los concursos que requieran una preparación específica o gran cantidad de trabajo pueden resultar demasiado absorbentes para los participantes motivados e influir negativamente en su rendimiento académico.
- Es posible que el resultado del concurso parezca arbitrario, lo que minará la motivación de futuros aspirantes.
- La organización de estos concursos puede resultar bastante compleja y requerir notables recursos del centro o del profesorado. Si la organización resulta deficiente, también se verá afectada la motivación de los alumnos.

A la hora de pensar en un concurso de programación es necesario tener en cuenta los siguientes aspectos:

- **Duración del concurso**. Puede tratarse de horas o de meses; recuérdese la posible influencia en la dedicación de los alumnos a las asignaturas.
- **Criterios de valoración**. Se puede tener en cuenta el diseño, el estilo de codificación, la eficiencia (en términos de memoria o de velocidad de ejecución), la calidad (referida al número de defectos), etc.
- **Trabajo individual o en equipo**. El concurso puede ser individual o admitir la participación de equipos de alumnos, cosa que plantea desafíos adicionales a los participantes [6].
- **Número de problemas a resolver**. Se puede plantear un solo problema o muchos, con un sistema de puntuación; también es posible que el concursante pueda elegir los problemas que resuelve.
- **Subjetividad del fallo del jurado**. Dependiendo de lo que se someta a valoración, el fallo puede ser casi arbitrario ("elegancia de la solución") o claramente objetivo ("número de fallos frente a un banco de pruebas determinado").
- **Dificultad para el jurado**. La revisión de los programas puede requerir mucho tiempo o consistir en una simple prueba de ejecución.
- **Dificultad de organización**. En clara conexión con la duración del concurso, infraestructura necesaria o tareas que el jurado deba realizar.
- **Perfil del jurado**. En ciertos tipos de concurso podría plantearse el que fueran los propios alumnos los que puntuasen, por votación, los programas; esto puede resultar de interés didáctico para los votantes, aunque quizás no tanto para los concursantes (que podrían perderse implementando características *deslumbrantes*).
- **Lenguajes de programación**. Se puede orientar el concurso a un lenguaje dado o dar libertad. También conviene pensar en diferencias entre lenguajes imperativos, funcionales, etc., en especial al plantear el enunciado.

3. Grandes concursos y *parties*

La idea de un concurso de programación no es, en modo alguno, nueva; muchas organizaciones promueven diversos tipos de concurso. Es bien conocido el concurso de programación de ACM para estudiantes [1] (en adelante ACM-ICPC), que celebra en 2003 la final de su 27ª edición. En él toman parte equipos de más de 1.000 universidades de todo el mundo, a través de diversas fases eliminatorias.

Evidentemente, resulta muy apetecible obtener cualquier buen resultado en un torneo de estas características. No obstante, no todos los alumnos se sienten preparados para participar a este nivel (independientemente de que lo estén o no). Además, tanto el entrenamiento como la participación pueden requerir un notable esfuerzo, incluyendo en algunos casos desplazamientos, estancias en otras ciudades o países o conocimiento de idiomas. Es una opción válida, pero no parece la más asequible para fomentar una alta participación. Como ejemplo, la Universidad de Valladolid otorga créditos de libre configuración por participar en diversas fases del ACM-ICPC.

Otro tipo de eventos de cierta envergadura son los encuentros denominados *parties*, en cuyo marco se suelen organizar concursos orientados a la programación de *demos*, videojuegos y similares, a la seguridad, o también a la programación en términos generales. En España existen varios encuentros de este tipo [3]. Algunos, de hecho, han sido organizados por instituciones docentes universitarias [7]. Al igual que los concursos como ACM-ICPC, son una opción atractiva para los estudiantes; en este caso,

sobre todo, porque estos eventos constituyen una gran oferta de ocio en relación con la informática. No obstante, adolecen de problemas similares a los de los grandes concursos de programación:

- Necesidad de desplazamientos si la *party* no se celebra cerca del lugar de residencia o estudio.
- Pueden no adaptarse bien al perfil académico del alumnado (especialmente por el grado de dificultad) o no cubrir la temática que se busca (el uso de ciertos lenguajes de programación o el estudio de cierto tipo de problemas).
- Pueden no ser adecuados en términos de fechas y duración o no tener interés de forma global para el alumno.
- Muchas instituciones académicas no pueden afrontar la organización de un evento propio de estas características.

Una respuesta es abordar la realización de concursos más específicos, adaptados a las necesidades (y posibilidades) formativas de cada centro.

4. Concursos de programación

A continuación se comentarán algunos posibles tipos de concurso, en relación con las ventajas y desventajas que ofrecen.

4.1. Proyecto de desarrollo integral

Al estilo de algunos concursos de programación de juegos, *demos* o similares, los concursantes desarrollan durante un tiempo (semanas, meses) un producto que luego entregan. Se puede valorar cualquier aspecto (análisis, diseño...), incluyendo el número de características implementadas, o incluso la estética.

Este tipo de concursos permite trabajo en equipo, y además el alumno revisa todo el ciclo de desarrollo. Como contrapartida, la participación puede resultar muy costosa para el alumno, exige también bastante trabajo del jurado, y además el fallo puede parecer arbitrario dependiendo de qué y cómo se valore.

4.2. Satisfacción de requisitos

Se trata de construir un programa que simplemente cumpla los requisitos planteados. Una posibilidad es plantear un problema de gran dificultad, en el que la mera capacidad de análisis y síntesis sea definitiva. Un inconveniente es que existe la posibilidad de que ningún concursante acierte a dar con la solución (además, la programación no es decisiva en el resultado).

Otra opción es plantear problemas de dificultad normal (o incluso sencillos) y puntuar según cuántos se resuelvan en un tiempo dado. Este enfoque es el que se utiliza en el ACM-ICPC.

Los concursos basados en la mera satisfacción de requisitos tienen la ventaja de que se puede verificar de manera automatizada la solución; basta establecer restricciones claras sobre las entradas y las salidas. Este sistema se utiliza en el sitio web que la Universidad de Valladolid mantiene en relación con el ACM-ICPC y que cuenta con un *juez en-línea* [2].

Como mezcla de los dos enfoques, se puede plantear un problema cuya solución básica sea relativamente asequible pero cuyas ramificaciones sean de gran dificultad. Así ha hecho en el ICFP Programming Contest [8][13].

4.3. Eficiencia

El objetivo del concurso es resolver un problema mediante un programa que sea lo más eficiente posible; por ejemplo, el ganador es el programa que (resolviendo el problema) requiere menos tiempo de ejecución. Es una buena forma de recompensar un diseño cuidadoso de los algoritmos, y una de las pocas medibles. (Otras medidas de eficiencia, como el uso de recursos, administración de memoria, etc. pueden ser mucho más problemáticas).

La dificultad estriba en que en ocasiones es difícil hacer comparaciones de eficiencia si la diferencia no es grande, ya que hay muchos factores del entorno que pueden influir: los lenguajes utilizados, los equipos, etc.

4.4. Diseño

En este caso se valoraría el diseño, la elegancia de la solución, etc. Didácticamente resulta muy conveniente en el sentido de que se premian algunas de las características más significativas del perfil académico del alumno. Sin embargo, puede resultar difícil establecer criterios claros de valoración, y en consecuencia el fallo puede percibirse como subjetivo o ambiguo. Además,

requiere un notable trabajo del jurado, incluyendo la resolución de posibles diferencias de opinión.

4.5. Velocidad de programación

Se trata de una variante del concurso de satisfacción de requisitos; se premia exclusivamente el tiempo que el alumno tarda en construir un programa que resuelva un problema dado (asequible). Las ventajas de este enfoque son que resulta muy fácil de organizar frente a otros (la infraestructura y organización son muy similares a las de un examen práctico). Además, el resultado es objetivo y el criterio de valoración muy claro. Sin embargo, es evidente que la velocidad de programación no es precisamente el factor más importante en el perfil de un alumno, y ni siquiera cabe fomentarlo demasiado desde un punto de vista académico.

4.6. Lucha entre programas

Otra versión clásica de los concursos de programación es organizar enfrentamientos entre programas que puedan competir de alguna manera. Existen juegos de programación basados en esta idea; CROBOTS [4] fue lanzado en 1985, y hay otras versiones más recientes para otros lenguajes [10]. En estos juegos los programas "luchan" en la memoria del ordenador, pero como evolución de esto se han creado versiones parecidas con una representación visual tridimensional, como en el caso de los D-Robots, cuyo lenguaje es similar a Pascal [5] o más recientemente Terrarium para .NET [15].

Estos juegos son interesantes porque se fomenta el trabajo en equipo y se pueden crear proyectos relativamente creativos manteniendo sin embargo un criterio de valoración uniforme y objetivo (la victoria en las partidas). Además, la infraestructura suele ser asequible[1], y en el caso de que haya una representación gráfica los estudiantes que no concursan pueden involucrarse y seguir las partidas como cualquier competición deportiva. Los problemas vienen porque su preparación requiere del estudiante cierto esfuerzo y posiblemente un aprendizaje específico.

[1] La versión actual de D-Robots exige tarjetas de vídeo con capacidad de aceleración 3D.

4.7. Relevos

Se puede organizar una competición basada en alguna de las demás modalidades (por ejemplo, velocidad de programación) pero por turnos; un concursante empieza a resolver el problema durante un tiempo límite, y cuando el tiempo acaba le sustituye el otro miembro del equipo. Puede haber más turnos hasta llegar a la solución. Se puede permitir una breve charla en el seno del equipo al principio o, por el contrario, impedir toda comunicación fuera del código y / o la documentación.

Este tipo de concurso tiene importantes ventajas; frente a la velocidad de programación pura, que no resulta del todo apropiada como objetivo didáctico, se premia la claridad, el buen diseño, la documentación, el desarrollo de cara al mantenimiento... Además, el criterio de valoración puede ser muy claro (siguiendo con el ejemplo, la velocidad de programación), pero a la vez el buen diseño, programación y documentación son necesarios por motivos prácticos. Los inconvenientes son que hay que elegir los problemas cuidadosamente (para que no los resuelva el primer relevo y a la vez sean abordables por turnos). También son más difíciles de organizar y de desarrollo previsiblemente más largo.

4.8. Duelo

Este tipo de concurso se centra en la resolución de problemas. Los concursantes se enfrentan a pares, como en un torneo individual de tenis o ajedrez; uno plantea al otro un problema que tienen que resolver ambos en cierto tiempo, y luego el segundo plantea su propio problema. A cada jugador se le valora el resultado (del tiempo de programación, por ejemplo) de ambos programas.

Es interesante porque obliga al concursante a pensar problemas variados que, pudiendo resolver él mismo en un tiempo limitado, tengan alguna dificultad para el oponente. Además, cada alumno se enfrentará a diversos problemas. Como inconveniente, es posible traer problemas rebuscados que el oponente sea incapaz de resolver aunque la programación sea fácil cuando uno conoce la solución.

```
#include <stdio.h>
main(t,_,a)char *a;{return!0<t?t<3?main(-79,-13,a+main(-87,1-_,
main(-86,0,a+1)+a)):1,t<_?main(t+1,_,a):3,main(-94,-27+t,a)&&t==2?_<13?
main(2,_+1,"%s %d %d\n"):9:16:t<0?t<-72?main(_,t,
"@n'+,#'/*{}w+/w#cdnr/+,{}r/*de}+,/*{*+,/w{%+,/w#q#n+,/#{l+,/n{n+,/+#n+,/#\
;#q#n+,/+k#;*+,/'r :'d*'3,}{w+K w'K:'+}e#';dq#'l \
q#'+d'K#!/+k#;q#'r}eKK#}w'r}eKK{nl]'/#;#q#n')){}#}w'){){nl]'/+#n';d}rw' i;# \
){nl]!/n{n#'; r{#w'r nc{nl]'/#{l,+'K {rw' iK{;[{nl]'/w#q#n'wk nw' \
iwk{KK{nl]!/w{%'l##w#' i; :{nl]'/*{q#'ld;r'}{nlwb!/*de}'c \
;;{nl'-{}rw]'/+,}##'*}#nc,',#nw]'/+kd'+e}+;#'rdq#w! nr'/ ') }+}{rl#'{n' ')# \
}'+}##(!!/")
:t<-50?_==*a?putchar(31[a]):main(-65,_,a+1):main((*a=='/')+t,_,a+1)
  :0<t?main(2,2,"%s"):*a=='/'||main(0,main(-61,*a,
"!ek;dc i@bK'(q)-[w]*%n+r3#l,{}:\nuwloca-O;m .vpbks,fxntdCeghiry"),a+1);}
```

Figura 1. Ejemplo de C *enrevesado (obfuscated)*.

4.9. *Obfuscated*

Tienen gran tradición las secciones de *obfuscated code* o *código enrevesado* en las revistas de programación. El código enrevesado es código deliberadamente oscuro, generalmente recurriendo a sutilezas del lenguaje de programación, que resulta difícil de leer e interpretar. A veces parece imposible que compile sin errores; otras parece correcto pero su comportamiento no es el esperado. Algunos lenguajes son especialmente *apropiados* para esto, y existen diversos concursos; C tiene una larga tradición [9]. En la Figura 1 puede verse un famoso ejemplo (un programa que imprime por pantalla un poema).

Es un ejercicio interesante para conocer a fondo un lenguaje de programación, y puede ser apropiado para la modalidad de *duelo*. Pero también es cierto que esta habilidad, por sí misma, no resulta muy significativa de cara a la formación de un perfil profesional.

5. El I Concurso de Velocidad de Programación en la EUITIO

5.1. Antecedentes y convocatoria

Con el fin de fomentar el interés por la programación y, sobre todo, servir de recordatorio sobre la posibilidad de entender la programación como una actividad lúdica, la dirección de la EUITIO decidió organizar un concurso de programación. Este concurso se enmarcaba en las actividades de conmemoración del XX Aniversario de la Escuela.

De cara a la organización había varias dificultades; en primer lugar, los recursos disponibles eran muy limitados. La actividad diaria de la EUITIO y otros actos del XX Aniversario ya no dejaban mucho margen para trabajar en la organización del concurso.

Pensando en evitar problemas de interpretación y en las dificultades de encontrar miembros para el jurado, se decidió organizar un concurso de velocidad puro: un solo problema, que se resolvería en el momento y en el menor tiempo posible. De este modo, las dificultades de organización serían mínimas.

No teníamos mucha confianza en una respuesta entusiasta por una serie de razones, pero en la medida de lo posible intentamos presentar cada inconveniente como una ventaja o desafío:

- **El concurso se parece a un examen práctico contra el reloj, experiencia no muy agradable.** En los anuncios y bases del concurso se desafiaba a los participantes a "soportar la presión" (Fig. 2).
- **No se podía valorar la elegancia o el diseño, porque eso plantearía un mayor trabajo de organización.** Se incidió en la oportunidad de programar *libremente*, sin normas (¡por una vez!) sobre documentación, análisis o diseño.
- **Los medios de hardware, software y personal de soporte eran limitados**. Se advertía de que no cabía reclamar al respecto ("este es un concurso para programadores, no para llorones").
- **No había posibilidad de ofrecer premios en metálico**. Se ofreció como premio "sólo la gloria"; recibir un documento acreditativo en la entrega de diplomas y ver el propio nombre en la página web de la Escuela.

Figura 2. Cartel del concurso.

Se redactaron unas bases informales para el concurso, que pueden consultarse en [12].

5.2. Inscripción y participación

De este modo se abrió el plazo de inscripción, y se inscribieron 34 personas. Fue una agradable sorpresa, ya que las previsiones eran simplemente alcanzar un número de participantes que permitiese celebrar dignamente el concurso (al menos 5). El día del concurso acudieron 18 participantes. Las salas de ordenadores de la EUITIO albergan a unos 25 alumnos como máximo, por lo que preveíamos la utilización de dos salas y un esfuerzo extra para sincronizar en ambas el desarrollo del concurso; finalmente, no fue necesario.

Un dato curioso es la participación de las mujeres; sólo 2 se inscribieron (y acudieron al concurso), lo que representa un 6% de los inscritos y un 11% de los participantes reales. Esta proporción no representa en modo alguno la proporción de sexos en el alumnado de la EUITIO (que puede verse en la Figura 3). En la pasada Campus Party 2002 [3] el porcentaje de mujeres participantes fue de un 9%, frente a un 91% de hombres.

	♂		♀	
Plan antiguo				
I.T.I.Sist.	350	75,75 %	112	24,25 %
I.T.I.Gest.	204	43,68 %	263	56,32 %
Plan nuevo				
I.T.I.Sist.	179	81,73%	40	18,27%
I.T.I.Gest.	171	70,95%	70	29,0º5%

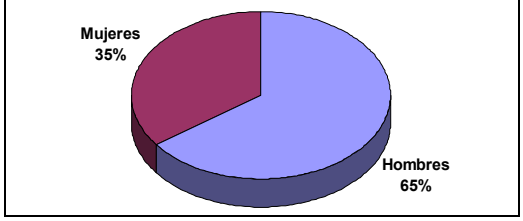


Figura 3. Distribución del alumnado por sexos en la EUITIO.

Sería interesante saber si los alumnos (en general) que no respondieron a la convocatoria no lo hicieron porque la materia (o la informática en sí misma) no les interesa, porque no creen que puedan ganar el concurso, porque la "gloria" y la competitividad no les parecen atractivas, porque les resulta demasiado estresante programar contra el reloj... y en qué medida estos motivos tienen más peso en uno u otro sexo.

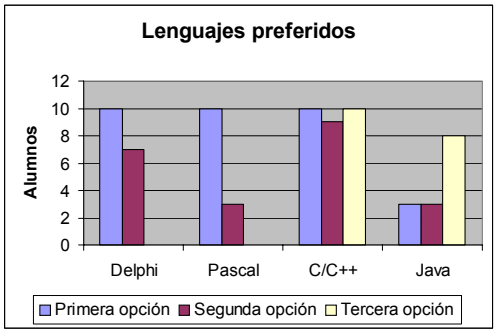


Figura 4. Distribución de lenguajes elegidos.

En el mensaje de inscripción cada alumno debía elegir sus lenguajes preferidos, por orden, a fin de tomar decisiones sobre el software a instalar. La distribución de las preferencias puede verse en la Figura 4 (incluye sólo los lenguajes más solicitados; por ejemplo, dos alumnos anotaron Visual Basic como segunda opción).

5.3. Preparación del problema

El programa a construir tenía que leer un fichero de texto de entrada y ordenar (línea por línea) sus palabras en un fichero de salida, con la salvedad de que debía ser capaz de manejar (teóricamente) un fichero de entrada de tamaño arbitrariamente grande. Es decir, no se considerarían válidas las soluciones que cargasen en memoria toda la información. Básicamente, se trataba de un simple algoritmo de ordenación secuencial de ficheros.

Por razones de disponibilidad de salas y personal de administración y servicios, el concurso podía durar sólo 3 horas en total. Con anterioridad a la celebración del concurso un miembro de la organización hizo la prueba, creando una versión del programa en C, relativamente compleja (porque incluía alguna pequeña mejora) que constituía una solución válida; se trataba de un programa de unas 200 líneas que estuvo terminado en algo menos de 50 minutos. Contando con una probable diferencia de velocidad a favor del profesor, parecía razonable esperar que algún alumno diera con la solución en el plazo de 3 horas.

5.4. Mecánica del concurso

Una vez vistas las preferencias respecto a lenguajes, se vio que el concurso podría celebrarse con la infraestructura disponible en los laboratorios. En concreto, unos días antes se informó a los alumnos de que estarían disponibles los compiladores de la Figura 5, a fin de que pudieran familiarizarse con el entorno o la línea de órdenes antes del concurso si lo necesitaban. (Suele haber otros compiladores, pero sólo se garantizaba la disponibilidad de estos).

Tras recibir a los participantes en una sala de teoría, se les expuso la mecánica del concurso y se les explicó también el enunciado del problema. Una vez atendidas las preguntas, se pasó al laboratorio donde tendría lugar la competición.

Lenguaje	Compiladores
C/C++	Borland C++ 5.02 Microsoft Visual Studio 6.0
Pascal/Delphi	Borland Delphi 5
Java	Borland Personal JBuilder 7

Figura 5. Compiladores disponibles en el concurso

Los alumnos podían utilizar papel y lápiz, pero no podían disponer de ningún material informático propio; el primer paso, además, fue desconectar físicamente las conexiones de red de todos los equipos. Acto seguido se les facilitó una copia impresa del enunciado, así como un disquete con un par de pequeños programas; uno servía para generar un caso de prueba, y el otro para verificar la solución que se había dado a dicho caso de prueba.

Una vez repartido el material, se puso en marcha el cronómetro. Cada alumno que creyese tener una solución avisaba a los organizadores; en ese momento, se paraba el cronómetro y todos los participantes debían interrumpir su actividad y quedarse de pie junto a su puesto mientras que se comprobaba la solución. Si esta no era buena, el cronómetro se ponía de nuevo en marcha y los concursantes podían continuar.

Estas interrupciones formaban parte del juego; de hecho, antes del concurso hubo algunas sugerencias acerca de incorporar estrategias de distracción (o incluso cortes de luz, según las versiones más radicales), que en principio se desestimaron hasta ver cómo se desarrollaba esta primera experiencia. Hubo tres "falsas alarmas". Dos de ellas se debían a programas que funcionaban, pero que de alguna manera cargaban toda la información en memoria. La tercera fue debida a un programa que, durante la verificación, falló (por un aserto).

Finalmente, este programa, una vez corregido, se dio por bueno a los 53 minutos de empezar el concurso. Se trata de un algoritmo muy ineficiente pero muy simple; menos de 60 líneas de código en C [12]. De acuerdo con las normas del concurso, este programa resultó ganador.

Para nuestra sorpresa, muchos participantes estaban dispuestos a continuar, por lo que se decidió seguir adelante hasta encontrar un segundo programa válido (unos 6 minutos después y en lenguaje Delphi) y un tercero (un par de minutos más tarde, también en Delphi). En este punto se dio por terminado el concurso.

5.5. Comentarios al resultado

A pesar de las muchas restricciones de material, personal y tiempo de organización, el concurso resultó ser un rotundo éxito. Hemos recibido diversos mensajes de felicitación de los alumnos,

pero además al término del concurso varios alumnos comentaron que les gustaría participar en otros similares, proponiendo diversas modalidades. A pesar de lo "árido" que puede resultar en algunos aspectos un concurso de velocidad de este tipo, los participantes lo vivieron realmente como una actividad lúdica. Ha tenido, además, una cierta repercusión entre el alumnado, y el alumno ganador recibió su premio en el acto académico de la fiesta de la Escuela, junto con los diplomas de la 18ª promoción de la EUITIO.

El ganador es un buen programador, pero por las preguntas que realizó sobre el enunciado sabemos que cuando entró en el laboratorio no sabía cómo solucionar el problema. Además, otros alumnos con gran afición por la programación y un excelente nivel no quedaron entre los tres primeros. Esto no tiene nada de sorprendente; este enfoque de la programación es, en gran medida, "deportivo", y como en cualquier encuentro deportivo aislado influyen en el resultado muy diversos factores, muchos de ellos circunstanciales.

Ciertamente, el ejercicio de programación planteado no es de gran nivel; además, resulta muy difícil medir el efecto real del concurso. Pero ha habido una respuesta entusiasta de muchos alumnos y el resultado es excelente dadas las limitaciones del planteamiento.

6. Conclusiones

A pesar de que el perfil típico del estudiante de Informática no necesariamente responde al de un aficionado a la programación, un concurso de programación puede despertar el interés de suficientes personas, sirviendo como un acicate en su formación para los participantes y como un "recordatorio" para el resto de los alumnos.

Aunque existen diversos concursos y ciclos de actividades con cierta tradición, es perfectamente posible organizar concursos propios, con recursos muy limitados y más adecuados al perfil e intereses de los estudiantes. Muchas de las limitaciones organizativas pueden transformarse en desafíos que incluso hagan más atractivo el concurso.

La respuesta a nuestra primera experiencia de este género ha sido sorprendentemente positiva, a pesar de que otro tipo de competiciones puede parecer más vistoso a priori. Los alumnos se han mostrado entusiastas ante la organización de este evento y esperan más en el futuro.

En la EUITIO tenemos intención de repetir este tipo de actividades, y tenemos especial interés en alguna modalidad de tipo D-Robots y en un concurso por relevos.

Referencias

[1] The ACM International Collegiate Programming Contest. http://www.acm.org/contest

[2] ACM International Collegiate Programming Contest. Universidad de Valladolid. http://acm.uva.es/problemset

[3] Campus Party. http://www.campus-party.org

[4] CROBOTS. http://www.nyx.net/~tpoindex/crob.html

[5] D-Robots. http://www.plasmacode.com

[6] Ernst, Fabian; Moelands, Jeroen; Pieterse, Seppo. *Teamwork in Programming Contests: 3 * 1 = 4*. ACM Crossroads Student Magazine, nº 3.2 (invierno 1996).

[7] FIBERparty. http://www.fiberparty.net

[8] International Conference on Functional Programming (ICFP). http://www.math.luc.edu/icfp

[9] International Obfuscated C Code Contest. http://www.es.ioccc.org/main.html

[10] JRobots. http://www.mobydisk.com/java/jrobots

[11] Martínez, Gloria; Fabregat, Germán. *Perfil profesional y académico de la informática en España*. Actas de las VIII Jornadas de Enseñanza Universitaria de la Informática, Cáceres, julio de 2002.

[12] Página web sobre el I Concurso de Velocidad de Programación de la EUITIO. http://www.euitio.uniovi.es/actividades/concursovelprog-i.php3

[13] Ramsey, Norman; Scott, Kevin. *The 1999 {ICFP} Programming Contest*. SIGPLAN Notices, Vol. 35, nº 3 (Marzo de 2000), págs. 73-83.

[14] Shilov, Nikolay V.; Yi, Kwangkeun. *Engaging Students with Theory through ACM Collegiate Programming Contests*. ACM Communications, Vol. 45, nº 9 (Septiembre de 2002), págs. 98-101.

[15] Terrarium. http://www.gotdotnet.com/terrarium

Cómo motivar al alumnado entrelazando las asignaturas Programación Avanzada y Tecnología de Computadores

Pedro Pablo Garrido Abenza
Héctor Francisco Migallón Gomis
Dept. Física y Arquitectura de Computadores
Universidad Miguel Hernández
03202 Elche (Alicante)
e-mail: pgarrido,hmigallon@umh.es

Resumen

En este artículo se presenta una experiencia realizada en las titulaciones técnicas de telecomunicaciones de la Universidad Miguel Hernández de Elche, con el fin de motivar al alumnado entrelazando dos asignaturas de su plan de estudios. Actualmente esta experiencia también se quiere implantar en la titulación de Ingeniero Técnico en Informática de Gestión (ITIG) de la misma universidad. Concretamente esta experiencia se basa en que mientras un alumno de ITIG cursa la asignatura de primer curso Tecnología de Computadores (TC), utiliza para el desarrollo de las prácticas de esta asignatura el software desarrollado por un compañero suyo en la asignatura optativa de tercer curso Programación Avanzada (PA), quien tiene la ventaja de haber sido en primer lugar un usuario y de alguna manera el que ha realizado el test de funcionamiento de un software diferente pero con las mismas funciones.

1. Introducción

A la mayoría de estudiantes de ITIG o de las distintas ingenierías técnicas en telecomunicaciones, el temario de la asignatura TC, o de las diferentes asignaturas de similar temario, les parece que está apartado del conjunto de conocimientos que tienen que adquirir en el transcurso de sus estudios superiores. Esto supone una traba para el profesorado que tiene que impartir una materia que no es del agrado del estudiante, y para el estudiante porque tiene la sensación de que no le va a servir de nada lo que está aprendiendo. Obviamente no existe duda alguna de que esto es erróneo.

Por otro lado el estudiante de PA se encuentra con una asignatura optativa de último curso de ITIG de la cual tiene buenas expectativas, ya que las asignaturas optativas son las que le van a dotar de conocimientos específicos y de las que espera contenidos que puedan aplicar directamente en su cercana vida profesional, por tanto es complicado cubrir esas expectativas.

Pensamos que un modo correcto de satisfacer esas expectativas es realizar una aplicación informática de simulación desde el principio hasta el final. Esto, además, es un buen incentivo para el alumno por ser la primera aproximación a un primer trabajo de desarrollo completo que puede utilizar como experiencia previa al Proyecto Fin de Carrera, sin confundir la diferencia de envergadura de ambos. Hay que destacar que, por lo general, el estudiante se encuentra con que sus trabajos prácticos normalmente son tareas concretas, independientes unas de otras y con una extensión en el tiempo que no suele superar las cuatro sesiones de dos horas. En nuestro caso proponemos un único trabajo que extenderemos durante todas las prácticas de la asignatura, consistente en el desarrollo de un simulador de circuitos digitales completo, el cual se realiza en fases sucesivas de forma continua, comenzando por el análisis y el diseño del mismo, y continuando con su implementación en el lenguaje de programación Java. Esto hace que también vayan perdiendo dependencia del grupo en conjunto y que se trabaje de verdad en un grupo reducido de personas, lo cual está más ajustado a la realidad que se van a encontrar una vez finalicen sus estudios.

2. Descripción de la asignatura "Tecnología de Computadores"

La asignatura TC [1], [2] basa su contenido en el estudio de:

- Sistemas de representación numéricos.
- Funciones lógicas.
- Circuitos combinacionales.
- Circuitos secuenciales.

Tanto los sistemas de representación numérica como las funciones lógicas no suponen ningún problema ni en su docencia ni en la predisposición del alumnado, que considera que esta parte sí es necesaria como base para las asignaturas de programación. De hecho las prácticas de la asignatura sí usan el contenido de esta primera parte de la asignatura pero no es necesario el refuerzo de dicha parte con sesiones prácticas específicas. Sin embargo, tanto la parte correspondiente a circuitos combinacionales, como la de circuitos secuenciales, es más difícil atraer la atención y la curiosidad del alumnado. Esto se debe a diversos motivos. El fundamental es que el estudiante considera que no le aporta nada, y en segundo lugar, que es poco tangible para él. Todo esto, unido a lo expresado anteriormente, nos lleva a utilizar todas las sesiones prácticas para desarrollar esta parte del temario de la asignatura.

3. Descripción de la asignatura "Programación Avanzada"

La asignatura PA basa su contenido en el estudio de un lenguaje de programación avanzado como es JAVA [3]. Si nos centramos en las características de la titulación de ITIG, antes de llegar a cursar esta asignatura se han adquirido los conocimientos básicos de programación en la asignatura Fundamentos de Programación (FP), donde han trabajado con el lenguaje C, viendo prácticamente la totalidad de sus características y prestaciones. La asignatura FP se imparte en primer curso. Posteriormente, en la asignatura de Estructura de Datos (ED) de segundo curso, se refuerzan los conocimientos de C y se entra a estudiar nociones de Programación Orientada a Objetos (POO) trabajando con C++. El estudio de la POO [4] es muy importante para el desarrollo de la asignatura de PA, ya que permite dedicarnos de pleno a las características avanzadas del lenguaje JAVA. Así pues, el temario de la asignatura lo distribuiremos en los siguientes temas o unidades didácticas:

- Conceptos básicos del lenguaje JAVA.
- Programación Orientada a Objetos.
- Conceptos avanzados de programación.
- Aplicaciones gráficas.
- Flujos de datos.
- Comunicaciones e Internet.

En el primer punto se presenta los elementos básicos del lenguaje Java, tales como las palabras clave y estructuras de control, muy similares a las del lenguaje C. En el segundo punto se hace un repaso de la POO de forma genérica, es decir, sin particularizar para ningún lenguaje concreto. Puesto que, como ya hemos adelantado, el alumnado ya ha adquirido estos conocimientos, se puede basar el contenido de la asignatura en los conceptos avanzados del lenguaje, que son los necesarios para la realización completa del software de simulación.

4. La "no" relación entre ambas asignaturas

Aunque es evidente que ambas asignaturas no tienen ningún tipo de relación que las una en el temario, es posible establecer la comunicación entre ambas asignaturas.

El hecho de que existe una tendencia a la desaparición de las incompatibilidades entre asignaturas, debemos tener en cuenta este factor a la hora de relacionar asignaturas. Podemos generar un problema a determinados estudiantes que por motivos cualesquiera lleguen a PA sin haber superado TC. Por tanto ha de comunicarse a los alumnos, en el programa de la asignatura que es aconsejable haber superado la asignatura TC. No obstante, lejos de ser un obstáculo, esta relación podría producir el efecto contrario, esto es, el desarrollo de las prácticas de PA podría ayudar al alumno en el aprendizaje de TC.

5. Justificación de la experiencia

La finalidad de esta experiencia es la de que el alumnado encuentre motivaciones para estudiar y trabajar con interés en ambas asignaturas.

Se obtiene un doble beneficio. El primero de ellos con sobre los alumnos de TC. La experiencia que tenemos hasta el momento es que el estudiante de TC acoge con mayor agrado el software de simulación creado por sus compañeros -aunque sólo sea por ver de lo que serán capaces de hacer y mejorar- que cualquier otro software comercial o de libre distribución estándar, los cuales suelen incluir muchas más funciones de las necesarias, dificultando así las fases iniciales del aprendizaje. El segundo de los beneficios se obtiene con los alumnos de PA, que agradece el desarrollo de un trabajo con ya determinada entidad, factor este muy importante en el desarrollo profesional del alumno.

Una característica importante a reseñar es que intentar motivar al alumnado en el desarrollo de TC se debe, en gran parte, a la inexistencia de laboratorios de electrónica en las titulaciones de informática, o la gran tasa de uso de éstos en las titulaciones de telecomunicaciones. Esta carencia, aunque es lógica, provoca que la asignatura TC tenga la problemática de que el alumnado no experimenta realmente con los circuitos integrados, y nunca llega a saber realmente como se trabajaría con ellos. El disponer de estos laboratorios sería un gran aporte para la asignatura TC, pero debe comprenderse que ésta sería una de las últimas prioridades para una universidad o escuela politécnica.

6. Motivación para el alumnado

Como hemos dicho, la finalidad de relacionar ambas asignaturas es motivar al alumnado. Esta motivación debemos llevarla a cabo en ambas asignaturas. Es decir, debemos intentar que, aunque una asignatura se apoye en la otra, exista motivación en ambas.

Dicha motivación la llevamos a cabo para la asignatura TC por medio de las siguientes premisas:

- Cada estudiante utiliza una herramienta creada por un compañero suyo, por tanto, ya no se parte de la idea de que es un software prefecto, pudiendo encontrar, por tanto, posibles fallos. Esto le lleva a experimentar, analizar y comprobar la herramienta.
- El conocer de primera mano, como usuario final de la aplicación, cuáles han sido los fallos o características no adecuadas de la herramienta, para más adelante utilizar los resultados de esta experimentación en el desarrollo de su propio simulador de circuitos digitales.

Y para la asignatura PA por medio de las siguientes premisas:

- El saber que la herramienta que está realizando puede ser utilizada por un compañero suyo hace que intente desarrollar un software robusto para ponerle difícil a su compañero el encontrar los posibles fallos.
- Será la primera vez que se comience una aplicación y se finalice por completo, de modo que esta experiencia puede ser utilizada como medio de aprendizaje para la realización de proyectos o para iniciarse en todas las fases de las que debe constar un proyecto.

Por último, hay una motivación espontánea en los alumnos de ambas asignaturas, relacionada con un objetivo psicomotor establecido en su programa. Este objetivo, no es más que el de dar buen ejemplo por parte de los profesores, haciéndoles ver a los alumnos la capacidad que pueden llegar a alcanzar, así como hacerles ver las ventajas de desarrollar herramientas software propias, que aunque no sean tan perfectas y completas como otras herramientas comerciales estándar, aquellas están más personalizadas y se pueden mejorar constantemente.

7. Prácticas a realizar en TC

En la asignatura TC las prácticas consistirán en la experimentación con puertas lógicas y en el diseño, realización y comprobación de diferentes circuitos secuenciales y combinacionales. La realización de estos circuitos y la comprobación de los mismos, por las características de las titulaciones de informática y sus dotaciones de laboratorios (aulas de informática), han de desarrollarse necesariamente bajo un software de simulación. Con este software de simulación los

alumnos manejarán todo tipo de puertas lógicas, realizarán sus circuitos y comprobarán el funcionamiento de los mismos.

En particular el simulador estará preparado para trabajar con:

- Puertas AND.
- Puertas OR.
- Puertas NAND.
- Puertas NOR.
- Puertas XOR.
- Puerta NOT.
- Entradas Digitales.
- Salidas Digitales.

Como el simulador con el que se trabaja no será ninguna versión que haya pasado un filtro de test la primera y segunda sesión de prácticas (4 horas) consistirán en poner a prueba dicho simulador, es decir, en hacerle un test de funcionamiento. Con esto conseguimos que se consoliden los conocimientos que se están adquiriendo en la asignatura, ya que para poner en duda el funcionamiento del simulador, se debe dudar realmente de los resultados mostrados por éste. Se orientará cómo comenzar la fase de test, empezando por las funciones lógicas básicas con puertas lógicas de dos entradas, construyendo puertas básicas de más de dos entradas a partir de varias puertas básicas (comprobando su equivalencia), construyendo puertas no incluidas en el simulador…. Esto se realiza en la primera sesión de prácticas. Es necesario hacer constar que, aun siendo software diferente, sí existe una uniformidad en las tareas básicas, lo cual facilita la explicación común del software al conjunto del alumnado.

La segunda sesión se dedicará, a modo de reto o de juego, consistente, a la búsqueda del fallo. Debe ser cada estudiante el que se invente diferentes circuitos, el que los monte en el simulador, el que calcule cómo deben funcionar y el que compruebe si dicho funcionamiento real coincide con lo que había calculado.

Esta sesión debe acabar con la emisión, por parte de cada estudiante, de un informe con los fallos detectados, si es que los hubiere, de manera que para la siguiente sesión de prácticas se sepa si el simulador con el que debe trabajar es adecuado, o es más recomendable la utilización de la versión desarrollada por nosotros denominada *Simulin*. Dicha versión no sólo ha sido testada por nosotros, sino que también lo ha sido por los propios alumnos de TC, como cualquier otro simulador desarrollado por alumnos de PA. Además, dispone otras funciones que vamos añadiendo continuamente, con el objetivo de cubrir la totalidad del temario de TC. Por ejemplo, podemos mencionar módulos para realizar conversiones entre sistemas de numeración, simplificación de funciones mediante mapas de Karnaugh (K-Maps), etc. Se les ofrece esta versión para evitar que la carencia de confianza en el software perjudique el desarrollo de sus prácticas.

Como se ha explicado, el inicio de trabajo con esta filosofía de reutilizar el software de otros compañeros, es completamente diferente al inicio con un software estándar, de modo que se incentiva el inicio de trabajo. Se constata que en las sesiones iniciales de prácticas la asistencia de alumnado es alta, y que es en este momento en el que hay que intentar motivarlos de tal modo que no desesperen, y para que continúen en el desarrollo completo de las prácticas. Con nuestra experiencia conseguimos disminuir el porcentaje de alumnado, que deja la asignatura para futuras convocatorias ya en el inicio de la asignatura.

Según nuestra experiencia, los problemas encontrados por los alumnos se deben más a la utilización de los diferentes eventos de ratón y teclado, que a problemas de funcionamiento correcto del simulador.

Se tiene en cuenta que, el software desarrollado por los alumnos de PA que se distribuye al alumnado de TC para realizar sus prácticas, ha sido evaluado por el profesor o profesores responsables. En esa evaluación se debe de marcar el software que sí es válido para la realización de las prácticas de TC, descartando, así mismo, el software que no sea válido. Este último caso no implica el suspenso en la asignatura de PA, sino que por diferentes motivos, ya sea por no ser lo suficientemente robusto, dificultad en el uso, documentación de ayuda en línea pobre,… no se considera adecuado para el uso de sus compañeros. Aun no significando un suspenso si debe verse penalizado en la nota.

8. Práctica a realizar en PA

En la asignatura PA dedicaremos todo el tiempo asignado a laboratorio a realizar un entorno de simulación de circuitos digitales, incluyendo

puertas lógicas, circuitos secuenciales y combinacionales, dotado de las características básicas de las que consta cualquier simulador de este tipo. Debe de cumplir, por tanto, con las características mencionadas para el simulador utilizado en el desarrollo de las prácticas de TC y por supuesto debe cumplir con unas reglas fijadas, puesto que, como ya hemos dicho, este software de simulación será utilizado por sus compañeros de TC.

La práctica realmente se desarrolla en dos fases claramente diferenciadas. En la primera de ellas se realizará un simulador modo texto, y tendrá las siguientes características:

- Permite todos los tipos de puertas antes mencionados.
- Todas las puertas lógicas, excepto la NOT, pueden tener dos ó más entradas.
- Los circuitos se construyen de forma programada, esto es, para implementar un circuito distinto tendremos que modificar el código fuente y recompilar.
- Obtención de la función lógica equivalente en notación prefija, como mecanismo de comprobación del circuito.
- Obtención de la tabla de verdad para la comprobación de resultados.

En la segunda fase, se dotará al simulador de una interfaz gráfica, la cual ya será la versión utilizable por sus compañeros en la asignatura TC. En esta versión se debe añadir las herramientas para visualizar las puertas, las conexiones y los módulos de entrada y salida a modo de circuito digital convencional, así como la creación de circuitos mediante el ratón y presentación gráfica de resultados (tabla de verdad).

La distribución de las prácticas en estas dos fases es importante, ya que se debe empezar desde la primera sesión con la realización del simulador, y, sin embargo, no será hasta avanzado la mitad del cuatrimestre cuando se adquieran los conocimientos básicos para el desarrollo del entorno gráfico. Por tanto, la base de la programación de las clases necesarias se debe ir desarrollando de manera que pueda ir probándose su corrección de algún modo, como por ejemplo, obteniendo la función lógica equivalente o la tabla de verdad del circuito. Se desarrollará los diferentes objetos, de los que debe constar el simulador, a nivel de comportamiento lógico y cómo relacionarlos con el resto de objetos, así como los módulos de obtención de resultados. Esta parte se puede desarrollar completamente prescindiendo del entorno gráfico y con la ayuda para la depuración de un lenguaje texto de simulación. Finalizada esta parte de desarrollo ya se dispone de los conocimientos necesarios para dotar a los objetos de su característica gráfica, y la construcción de aplicaciones gráficas.

Como acabamos de decir, la primera parte de la práctica implementa todas las funciones, y es manejable en modo texto. Tenemos que reseñar que sería aconsejable que el alumnado dedujera por sí mismo los esquemas de realización, ya que es una de las fases importantes en la realización de un proyecto, pero como esto supone un problema, se les ofrece una guía de cómo ir realizando el trabajo. El modo de utilizar las herramientas gráficas ya es una parte más creativa, en la cual puede aportar más ideas y contribuciones propias, fase en la cual no se ofrece guía de realización. Por último, la documentación de utilización del software está incluida en las tareas a realizar, y debe estar incluido en el software de simulación, de forma que el usuario posterior tendrá toda la documentación necesaria en el paquete de distribución.

En la evaluación de la práctica de PA debe de tenerse presente el hecho de que el software ha de poder reutilizarse, y es aquí donde se evitan los posibles problemas graves de reutilización del código. El código que cumple unas condiciones básicas es preparado para la realización de las prácticas de TC del siguiente curso. Puesto que la asignatura PA es de segundo curso, y TC es de primero, dispondremos de un mismo software de simulación para aproximadamente dos grupos (cuatro personas). Una opción que se desechó por problemas en el mantenimiento de las aulas de informática, era instalar una versión diferente en cada ordenador del aula.

9. Posibles mejoras

Una posible modificación que nos gustaría realizar en este planteamiento, es poder dejar a aquellos estudiantes aventajados que tienen las ideas muy claras, y un concepto diferente de cómo debe ser el simulador, libertad para la confección del mismo. Ello sólo sería posible, y desde nuestro punto de vista recomendable, si el tiempo disponible permitiera la confección de una ayuda

completa al usuario del software. De esta forma podríamos, al inicio de las clases prácticas de TC, impartir una breve introducción del programa, es decir explicar para qué sirve y qué es lo que queremos conseguir de él. Con esto y la ayuda en línea incorporada sería suficiente para que los alumnos de TC ya pudieran manejarse con el software de simulación.

Si es necesario la realización de una ayuda completa por el modo de enfocar las prácticas de TC, lo podemos conseguir dándoles comenzado el proyecto con determinados objetos ya parcialmente definidos. Así esta ganancia de tiempo la dedicarían a la implementación de una buena ayuda de usuario. Esta opción, que hemos desechado, lleva al alumnado a introducir el código entregado dentro de su software sin preocuparse de entenderlo, por lo que se decidió que desarrollaran el software completo.

10. Conclusión

De momento la experiencia nos dice que se ha conseguido la intención con la que comenzamos la planificación de este experimento, que no era más que motivar a nuestros estudiantes.

Además mostramos que es posible el trabajo común entre dos asignaturas no relacionadas en el temario, consiguiendo beneficios por ambas partes.

Comprobamos también, cómo en la asignatura TC partiendo de un inicio algo lúdico en las sesiones prácticas, el alumnado de alguna manera gana seguridad en sus conocimientos, y comprende que el poner en duda lo que se le transmite es una forma de aprender.

Por último en la asignatura PA hemos notado cómo incentivo el tener un desarrollo de cierta entidad, que el estudiante sabe que puede llevar a buen término con la ayuda del profesor, aunque dejándole bastante libertad en su ejecución.

Referencias

[1] Floyd, T.L. *Fundamentos de Sistemas Digitales*. Prentice Hall. 2000.

[2] Hayes, John P. *Introducción al diseño lógico digital*. Addison-Wesley Iberoamericana. 1996.

[3] Joyanes Aguilar, L., Zahonero Martínez, I. *Programación en Java 2*. McGraw-Hill. 2002.

[4] Muñoz Caro, C., Niño Ramos A., Vizcaíno Barceló A. *Introducción a la programación con orientación a objetos*. Pearson Educación. 2002.

Una revisión de métodos pedagógicos innovadores para la enseñanza de la programación *

Mercedes Gómez Albarrán
Dpto. de Sistemas Informáticos y Programación
Universidad Complutense de Madrid
Ciudad Universitaria s/n, 28040 Madrid
e-mail: albarran@sip.ucm.es

Resumen
Este trabajo realiza una profunda revisión de las diferentes tendencias actuales en lo que se refiere a métodos pedagógicos de soporte al aprendizaje de la programación en general.

Se incluyen referencias a numerosos trabajos que ejemplifican cada una de las tendencias así como direcciones en el World Wide Web en las que conseguir un buen número de herramientas.

1. Introducción

Es notorio que las nuevas tecnologías de la información y la comunicación han provocado cambios en la manera en la que los profesores impartimos nuestras clases y también en la forma en la que nuestros alumnos estudian.

El World Wide Web es utilizado actualmente por los educadores como una plataforma que hace viable una fórmula complementaria a la docencia presencial. Dicha fórmula incluye desde la disponibilidad de una ingente cantidad de material educativo (apuntes, transparencias, resoluciones de ejercicios, etc.) hasta el desarrollo de herramientas educativas atractivas y útiles que pueden estar al alcance de todos.

En este trabajo se presentan diferentes tendencias actuales en lo que se refiere a métodos de soporte al aprendizaje de la programación en general. Numerosas muestras de dichos métodos pueden ser encontradas diseminadas a lo largo del World Wide Web.

Antes de revisar dichos métodos, no queremos pasar por alto un cambio en las tendencias de los enfoques llevados a la práctica.

Algunos trabajos iniciales de soporte al aprendizaje de la programación se centraron en el desarrollo de tutores inteligentes. Un ejemplo es PROUST [21], un tutor basado en conocimiento para la enseñanza del lenguaje de programación Pascal que identifica errores no sintácticos. PROUST forma parte de un sistema más amplio que asigna tareas a los estudiantes, analiza el trabajo que éstos hacen y, posteriormente, realiza sugerencias apropiadas. Evidentemente, para llevar a cabo su tarea PROUST debe comprender lo que el alumno está haciendo, para lo cual cuenta con una gran cantidad de conocimiento tanto acerca de las tareas a realizar por el alumno como acerca de errores típicos en los programas escritos en Pascal. Un trabajo en la misma línea, pero dedicado a la enseñanza del lenguaje de programación Lisp, es Lisp Tutor [29]. Un poco más allá va The Programmer's Apprentice [30], una herramienta basada en conocimiento que no sólo ayuda en la fase de implementación sino también en las fases de análisis y diseño.

Los trabajos iniciales en el campo de la enseñanza de la programación fueron intentos altamente sofisticados, con una elevada componente de conocimiento. Sin embargo, los resultados fueron algo decepcionantes. Por ejemplo, los diagnósticos automáticos de errores en los programas no son una tarea fácil. Esto es algo que se puso de manifiesto en el sistema PROUST. Cuando se abordaban programas de cierta complejidad surgían problemas para comprenderlos y la capacidad del sistema para identificar errores se reducía drásticamente.

Las tendencias actuales en los métodos de ayuda a la enseñanza de la programación se sitúan

* Este trabajo ha sido financiado por el Ministerio de Ciencia y Tecnología (TIC2002-01961).

al otro lado del espectro. Se trata de enfoques menos sofisticados e "inteligentes". Entre estos enfoques nos encontramos:

- Sistemas que incluyen un reducido entorno de desarrollo
- Entornos basados en ejemplos
- Entornos basados en visualización y animación
- Entornos de simulación

Estos enfoques se tratan, en el orden indicado, en las secciones posteriores.

2. Sistemas con entorno de desarrollo incorporado

Algunos trabajos actuales se centran en el desarrollo de sistemas de enseñanza que incluyen un reducido entorno de desarrollo en el que los alumnos pueden escribir código.

Una idea común que hay detrás de estos trabajos es intentar conseguir entornos de desarrollo menos complejos que los comerciales. Estos últimos, en muchas ocasiones, resultan demasiado enrevesados para los alumnos que están aprendiendo a programar, quienes suelen utilizar un conjunto reducido de las facilidades que los entornos proporcionan.

Un ejemplo bastante reciente es AnimPascal [35], un entorno de enseñanza del lenguaje de programación Pascal que incluye capacidades de edición y compilación de código junto con visualización de la ejecución de los programas. Esta última capacidad consiste en que el alumno puede ver cómo afecta la ejecución de cada sentencia a los valores de las variables y la salida del programa. Esto no es algo novedoso pues los entornos de programación comerciales suelen ofrecer este tipo de facilidades.

Una característica que diferencia a AnimPascal de otros sistemas de enseñanza similares es que recoge el camino seguido por cada alumno en la solución de las cuestiones que aborda. Para cada cuestión se graban las diferentes versiones del programa solución elaborado por el alumno, junto con los resultados de las compilaciones sucesivas.

La información acerca del camino recorrido por el alumno para obtener una solución a un problema la consideramos de gran valor para poder sacar una idea de cómo concibe el alumno la programación y las técnicas de resolución de problemas. El uso concreto que hacen de dicha información en AnimPascal es que es procesada por los profesores posteriormente, quienes la utilizan para encontrar las "lagunas" de conocimiento y los problemas de aprendizaje de los alumnos.

Dentro de esta línea de trabajos se encuentra también el sistema BlueJ [2][24], una continuación del lenguaje y entorno Blue [22][23]. BlueJ es el resultado del trabajo conjunto de dos equipos de investigación: el de la Monash University, en Melbourne (Australia), y el del Mærsk Institute en la University of Southern Denmark.

BlueJ es un entorno de enseñanza para el lenguaje Java fácil de utilizar. Aparte de la facilidad de uso, también se hace énfasis en las técnicas de interacción, dando como resultado un entorno altamente interactivo que promueve la exploración y la experimentación.

Las versiones 1.2.2 y 1.3.0 beta1 de BlueJ, así como documentación acerca del entorno, se pueden obtener a partir de http://www.bluej.org.

3. Entornos basados en ejemplos

Sin lugar a dudas, los humanos utilizamos las soluciones a problemas previos para resolver nuevos problemas. Especialmente en el área de la programación, tanto los programadores con experiencia como los noveles a menudo se sirven de ejemplos de programas, ya hayan sido éstos desarrollados por ellos mismos o no. No en vano existe todo un área de investigación en Reutilización de software. Los profesores, conscientes de lo anterior, no dudamos en utilizar abundantes ejemplos de programas en nuestras clases.

No es de extrañar, por lo tanto, que como forma de ayuda al aprendizaje de la programación algunos trabajos giren en torno a la construcción de bases de ejemplos de programación y al desarrollo de mecanismos de acceso o selección (más o menos sofisticados) [10][18].

La falta de herramientas eficientes de selección de ejemplos es el cuello de botella de los entornos educativos basados en ejemplos. El problema crucial es que los alumnos noveles carecen del conocimiento y la experiencia necesarios para encontrar ejemplos relevantes

mediante herramientas como la selección a través de un menú [27] o la búsqueda mediante palabras clave de la especificación del problema o del propio código [17].

En [10] se analizan los problemas relacionados con la localización de ejemplos relevantes en este tipo de entornos, así como varios enfoques de selección de ejemplos:

- La selección conducida por el estudiante, para la que la tecnología hipermedia resulta altamente prometedora. Básicamente consiste en inspeccionar los ejemplos. La dificultad radica en cómo estructurar los ejemplos en un hiperespacio y qué tipo de enlaces proporcionar para la navegación
- La selección conducida por el sistema, en la que el sistema proporciona al alumno los ejemplos relevantes. Una posibilidad es que el sistema disponga de conocimiento acerca de qué ejemplos se adaptan a cada problema y, en base al problema que esté abordando el alumno, se sugiera lo que corresponda. Un enfoque más sofisticado es que el sistema disponga de conocimiento acerca del alumno y los problemas, y pueda usarlo para seleccionar "en vivo" el ejemplo relevante. En este tipo de enfoque, el alumno no suele poder pasar por alto la sugerencia del sistema.
- La selección colaborativa, en la que ambos agentes (sistema y alumno) intervienen: el primero sugiriendo ejemplos apropiados y el segundo realizando la selección definitiva.

Una idea más novedosa dentro del campo de la enseñanza basada en ejemplos es la que se presenta en WebEx [9], una herramienta basada en Web que permite explorar de manera interactiva ejemplos de programas autoexplicativos.

Lo novedoso de WebEx es el hecho de que los ejemplos sean autoexplicativos. En WebEx además se hace frente al problema de la heterogeneidad de los alumnos: alumnos con diferente nivel inicial de conocimiento y distintas capacidades de adquisición. La heterogeneidad hace que alumnos diferentes necesiten diferentes velocidades, ejemplos y nivel de detalle en las explicaciones de los ejemplos.

El núcleo de WebEx es una base de ejemplos de programación autoexplicativos: cada línea de código va acompañada de texto que aclara el papel de la misma en la solución global y su significado.

Los alumnos, en función de sus necesidades, pueden elegir su estrategia de exploración preferida. La diferencia entre las estrategias de exploración radica, básicamente, en la cantidad de comentarios asociados a los programas que está visible.

Para facilitar la navegación y la localización de ejemplos planean estructurar la base de ejemplos siguiendo un enfoque conceptual. Se trata de identificar los conceptos clave de los contenidos ejemplificados en los programas y mantener enlaces bidireccionales entre los conceptos y los ejemplos.

Otra característica de esta herramienta es que cada acción de un alumno en el entorno queda registrada, de forma que el profesor puede "monitorizar" la actividad de los alumnos y extraer conclusiones acerca de la forma en la que éstos trabajan con los ejemplos.

4. Entornos basados en visualización y animación

Sin duda alguna, uno de los métodos pedagógicos de enseñanza a la programación más explorado es el basado en la visualización y animación de algoritmos [37]. Se trata de presentar representaciones esquemáticas de programas y algoritmos, de manera que a los alumnos les sea más fácil entenderlos.

La animación y visualización de algoritmos se lleva utilizando desde hace más de dos décadas. Se puede decir que el comienzo de la animación y visualización de algoritmos es la cinta de vídeo *Sorting Out Sorting* presentada en 1981 en la conferencia ACM SIGGRAPH [1]. Desde entonces, numerosos sistemas y trabajos se han desarrollado centrándose en diferentes aspectos de la animación y la visualización.

En el World Wide Web podemos encontrar numerosos ejemplos de animaciones y visualizaciones predefinidas de algoritmos diseñadas por los instructores. Según los casos, estas animaciones y visualizaciones permiten un mayor o menor grado de intervención del alumno. En http://www.cs.hope.edu/~alganim/ccaa/index.html existe una enorme recopilación de enlaces a animaciones de algoritmos en el Web.

Aparte de las animaciones predefinidas, se han desarrollado numerosos sistemas de visualización y animación de algoritmos. La tendencia suele ser a conseguir independencia de la plataforma, facilidad de uso, soporte para inclusión del código fuente y descripciones textuales, etc.

A modo de ejemplo citaremos los siguientes:

- ANIMAL (A New Interactive Modeler for Animations in Lectures) [31][34], desarrollado en la Universidad de Siegen (Alemania). El sistema puede ser descargado a partir de http://www.animal.ahrgr.de/, donde también se pueden encontrar documentación, animaciones de ejemplo y publicaciones relativas al sistema.
- LEONARDO [13][16], desarrollado en la Universidad de Roma "La Sapienza" para la animación de programas escritos en C. Desde http://www.dis.uniroma1.it/~demetres/Leonardo/Leonardo.html se puede descargar el sistema (aún no disponible para Windows y Linux) junto con documentación adicional.
- XTANGO y POLKA, desarrollados en el Georgia Institute of Technology por el grupo de John Stasko –creador del sistema TANGO [36]– y accesibles desde http://www.cc.gatech.edu/gvu/softviz/algoanim/.
- JHAVÉ (Java-Hosted Algorithm Visualization Environment) [26], un entorno cliente-servidor que soporta la visualización de algoritmos en tres lenguajes de script diferentes: Samba, Animal y Gaigs. La versión disponible se puede conseguir desde http://csf11.acs.uwosh.edu/.

Otra línea de trabajos dentro de esta tendencia son las denominadas "teaching machines". En este caso el objetivo no es mostrar cómo se comporta un algoritmo concreto para facilitar la comprensión de la idea que hay detrás, sino mostrar el efecto que tiene la ejecución de las sentencias básicas de un cierto lenguaje de programación sobre los elementos de la computadora. Es decir, actúan a modo de "intérpretes" de las sentencias a través de la visualización del estado de la memoria y otros recursos. Se han construido herramientas de este tipo para enseñar lenguajes como C++ [8][38], un subconjunto de Java [3][25], e incluso una especie de máquina de Turing que ha sido usada para introducir la programación formal a los alumnos [14].

Recientemente, se están llevando a cabo proyectos en los que se recopilan tanto texto como animaciones y visualizaciones de algoritmos en los llamados "hypertextbooks" [7], como solución a la existencia actual de numerosas visualizaciones independientes que existen en el World Wide Web.

Para finalizar este apartado, señalar que existen trabajos en los que se ha evaluado la efectividad de los entornos basados en visualización y animación. Por ejemplo, en [32] se relata una experiencia de uso en grupos numerosos de alumnos y los resultados son esperanzadores.

Ahora bien, aún queda mucho camino por andar y los sistemas de visualización de algoritmos deben afrontar una serie de aspectos pedagógicos para que su uso sea más efectivo. Tal y como se desprende de trabajos como [19][33], sería conveniente que los entornos satisficieran, entre otros, requisitos pedagógicos como:

- Ser sistemas de propósito general, de forma que pudiesen integrarse a lo largo de todo un curso y no aplicarse sólo a aspectos muy concretos del temario.
- Permitir la introducción de datos con los que pueda trabajar el algoritmo, teniendo siempre cuidado con no sobrecargar al alumno excesivamente con la entrada de datos.
- Disponer de capacidad de "vuelta atrás", de manera que cuando un alumno se encuentre perdido o confundido por la representación esquemática proporcionada pueda retroceder.
- Disponer de predicción interactiva, es decir, interrumpir con preguntas en los puntos interesantes de la ejecución del algoritmo de forma que el alumno tenga que predecir qué va a ocurrir a continuación. Esto puede ayudar a despertar el interés de los alumnos.

  En un trabajo reciente [20] se pone en tela de juicio el valor de la predicción interactiva si los alumnos no se toman en serio las preguntas sino que la consideran un mero juego de adivinación. Para ello se sugiere satisfacer el siguiente punto.
- Recoger la interacción de los alumnos (las respuestas a las preguntas) en una base de datos, de manera que pueda ser utilizada por los profesores para detectar puntos en los que

los alumnos tienen dificultad e incluso como parte de la evaluación que se haga de los alumnos.

5. Entornos basados en simulación

Para finalizar, revisamos aquellos trabajos que se centran en el desarrollo de entornos de simulación. El objetivo de estos entornos es ayudar a las alumnos a comprender cómo funcionan los programas. Pero no se trata de visualizar la ejecución de los mismos viendo cómo afectan a los recursos de la computadora sino de ayudar a entender el efecto de las instrucciones de los lenguajes mediante algún tipo de simulación. Los alumnos observan un "mundo imaginario" en el que habitan seres cuyo comportamiento viene dictado por la ejecución de las instrucciones del programa. La ejecución de cada instrucción tiene definida de antemano una acción concreta que los seres llevan a cabo en ese mundo.

Indudablemente uno de los entornos de este tipo que ha tenido más éxito ha sido Karel, The Robot [28], utilizado para la presentación y enseñanza de los principios y estructuras básicos de la programación estructurada.

El robot habita en un mundo bidimensional muy simple compuesto por avenidas que van de norte a sur y calles que van de este a oeste. Las calles y las avenidas están numeradas de forma similar a cómo lo está el primer cuadrante del plano cartesiano, y existen muros impenetrables que hacen las veces de ejes de dicho cuadrante. Los robots sólo pueden estar en intersecciones de calles y avenidas, y sólo pueden estar mirando en una dirección (norte, sur, este u oeste). También puede haber muros impenetrables entre intersecciones adyacentes. Aparte de los robots, en el mundo existen otro tipo de objetos: timbres , los cuales sólo pueden estar situados en intersecciones.

El robot cuenta con una serie de periféricos: tres cámaras de vídeo que le permiten mirar de frente, a izquierda y a derecha; una brújula que indica la dirección en la que mira; un micrófono que le permite escuchar el sonido de los timbres; un brazo que le permite coger timbres, introducirlos y sacarlos de recipientes, y colocarlos de nuevo en el suelo; y una bolsa para guardar timbres.

Las seis acciones básicas que puede realizar el robot son: encenderse; moverse hacia delante; girar a la izquierda; coger un timbre e introducirlo en la bolsa; sacar un timbre de la bolsa y colocarlo en el suelo; y apagarse. Aparte de este reducido lenguaje del robot (correspondiente al igualmente reducido juego de instrucciones básicas del lenguaje de programación que se puede emplear para escribir los programas), los alumnos pueden definir sus propias funciones (implementadas en términos del lenguaje básico del robot), y el robot también puede comprobar condiciones para poder elegir entre alternativas o repetir acciones. De lo que no se dispone es de componentes básicas como la creación y manipulación algebraica de datos así como operaciones de entrada/salida.

El lenguaje utilizado por Karel es un lenguaje tipo Pascal. Cuando se ejecutan los programas, los alumnos pueden ver el mundo de Karel y observar los cambios que se producen en él a medida que avanza la ejecución.

Se puede obtener una versión de Karel a partir de http://www.sourceforge.net.

Tal ha sido el éxito de Karel que Joseph Bergin ha realizado una versión para la enseñanza de la Programación orientada a objetos utilizando una sintaxis similar a la de C++ y Java: Karel++ [5]. Los dos primeros capítulos de [5] pueden leerse on-line desde la página de Karel++: http://www.csis.pace.edu/~bergin/karel.html. Una versión más reciente con sintaxis Java pura está disponible desde hace poco también gracias a Joseph Bergin: Karel J. Robot [4][6]. Finalmente, JKarelRobot [11] es otra extensión del inicial que soporta estilos de programación tipo Pascal, Java y Lisp.

Otro ejemplo de este tipo de entornos de enseñanza es Alice [15], un entorno de animación en 3-D que permite crear mundos virtuales que reflejan el estado del programa y van cambiando a medida que cambia dicho estado. Alice ha sido desarrollado por el grupo de investigación Stage3 de la Universidad de Carnegie Mellon y está disponible desde http://www.alice.org/. Está siendo utilizado actualmente para abordar una estrategia *objects-first* en asignaturas de introducción a la programación [12].

6. Conclusión

Esta ponencia ha realizado un repaso en anchura y en profundidad por diferentes métodos pedagógicos que están siendo empleados en la actualidad en el área de la enseñanza de la programación.

Se han revisado cuatro métodos pedagógicos: los sistemas que incluyen un "limitado" entorno de programación, los entornos basados en ejemplos, los entornos basados en visualización y animación, y los entornos basados en simulación.

Todos los métodos son prometedores y están siendo aplicados con aparente éxito en diferentes centros. Sin embargo, los estudios de su efectividad son en su mayoría exploratorios. En este sentido, convendría realizar estudios empíricos que constatasen las aparentes mejoras que introducen en el proceso de aprendizaje de los alumnos.

Algunos retos que se pueden plantear a los diferentes métodos, y que en algunos trabajos concretos ya se afrontan, son:

- La inclusión de mecanismos que permitan el seguimiento del alumno. La motivación es doble: por un lado, ayudar a los instructores en la localización de lagunas cognitivas; por otro lado, poder utilizar los resultados como parte de la valoración general del conocimiento de los alumnos, a la vez que éstos utilizan de forma más responsable las herramientas.
- Afrontar la heterogeneidad (tanto de capacidad como de predisposición) de los alumnos. De esta forma, las herramientas resultan atractivas para un porcentaje mayor del alumnado.
- La introducción de mecanismos que permitan la colaboración y cooperación entre alumnos, soportando así el trabajo en equipo para la resolución de problemas en programación.

Referencias

[1] Baecker, R., 1981: "Sorting Out Sorting", *Procs. of SIGGRAPH*.

[2] Barnes, D.J., and Kölling, M., 2003: *Objects First with Java – A Practical Introduction using BlueJ*, Prentice Hall.

[3] Ben-Ari, M., Myller, N., Sutinen, E., and Tarhio, J., 2002: "Perspectives on program animation with Jeliot", *Procs. of Software Visualization: International Seminar, Lecture Notes in Computer Science 2269*.

[4] Bergin, J., 2000: "Introducing Objects with Karel J. Robot", *Procs. of the Workshop "Tools and Environments for Understanding Object-Oriented Concepts", European Conference on Object-Oriented Programming*.

[5] Bergin, J., Stehlik, M., Roberts, J., and Pattis, R., 1997: *Karel++ - A Gentle Introduction to the Art of Object-Oriented Programming*, John Wiley & Sons.

[6] Bergin, J., Stehlik, M., Roberts, J., and Pattis, R., 2003: *Karel J. Robot - A Gentle Introduction to the Art of Object-Oriented Programming in Java* (manuscrito sin publicar, disponible en http://csis.pace.edu/~bergin/KarelJava/Karel++JavaEdition.html)

[7] Boroni, C.M., Goosey, F.W., Grinder, M.T., and Ross, R.J., 2001: "Engaging Students with Active Learning Resources: Hypertextbooks for the Web", *Procs. of the 32nd SIGCSE Technical Symposium on Computer Science Education.*

[8] Bruce-Lockhart, M.P., and Norwell, T.S., 2000: "Lifting the Hood of the Computer: Program Animation with the Teaching Machine", *Procs. of the Canadian Electrical and Computer Engineering Conference*.

[9] Brusilovsky, P., 2001: "WebEx: Learning from Examples in a Programming Course", *Procs. of the World Conference of the WWW and Internet*.

[10] Brusilovsky, P., and Weber, G., 1996: "Collaborative example selection in an intelligent example-based programming environment", *Procs. of the International Conference on Learning Sciences*.

[11] Buck, D., and Stucki, D.J., 2001: "JKarelRobot: A Case Study in Supporting Levels of Cognitive Development in the Computer Science Curriculum", *Procs. of the 32nd SIGCSE Technical Symposium on Computer Science Education.*

[12] Cooper, S., Dann, W., and Pausch, R., 2003: "Teaching Objects-First in Introductory Computer Science", aceptado para *34th SIGCSE Technical Symposium on Computer*

Science Education (disponible desde http://db.grinnell.edu/sigcse/sigcse2003/programAtaGlance.asp)

[13] Crescenzi, P., Demetrescu, C., Finocchi, I. and Petreschi, R., 2000: "Reversible execution and visualization of programs with Leonardo", *Journal of Visual Languages and Computing*, 11(2).

[14] Dagdilelis, v., and Satratzemi, M., 2001: "Post's Machine: A Didactic Microworld as an Introduction to Formal Programming", *Education and Information Technologies*, 6(2).

[15] Dann, W., Cooper, S., and Pausch, R., 2000: "Making the Connection: Programming with Animated Small World", *Procs. of the Annual Conference on Innovation and Technology in Computer Science Education*.

[16] Demetrescu, C., and Finocchi, I., 2000: "Smooth animation of algorithms in a declarative framework", *Journal of Visual Languages and Computing*, 12(3).

[17] Faries, J.M., and Reiser, B.J., 1988: "Access and use of previous solutions in a problem solving situation", *Procs. of the 10th Annual Conference of the Cognitive Science Society*.

[18] Guzdial, M., 1995: "Software-realized scaffolding to facilitate programming for science learning", *Interactive Learning Environments*, 4(1).

[19] Hundhausen, C., Douglas, S., and Stasko, J., 2002: "A Meta-Study of Algorithm Visualization Effectiveness", *Journal of Visual Languages and Computing*, 13(3).

[20] Jarc, D., Feldman, M.b., and Heller, R.S., 2000: "Assessing the Benefits of Interactive Prediction Using Web-based Algorithm Animation Courseware", *Procs. of 31st SIGCSE Technical Symposium on Computer Science Education*.

[21] Johnson, L.W., and Soloway, E., 1985: "PROUST: Knowledge-Based Program Understanding", *IEEE Transactions on Software Engineering*, 11(3).

[22] Kölling, M., 1999: "Teaching Object Orientation with the Blue Environment", *Journal of Object-Oriented Programming*, 12(2).

[23] Kölling, M. and Rosenberg, J., 1996: "Blue - A Language for Teaching Object-Oriented Programming", *Procs. of 27th SIGCSE Technical Symposium on Computer Science Education*.

[24] Kölling, M., and Rosenberg, J., 2001: "Guidelines for Teaching Object Orientation with Java", *Procs. of the Annual Conference on Innovation and Technology in Computer Science Education*.

[25] Levy, R., Ben-Ari, M., Uronen, P., 2003: "The Jeliot 2000 Program Animation System", *Computer & Education*, 40(1).

[26] Naps, T., Eagan, J., and Norton, L., 2000: "JHAVÉ – An Environment to Actively Engage Students in Web-based Algorithm Visualization", *Procs. of the 31st SIGCSE Technical Symposium on Computer Science Education.*

[27] Neal, L.R., 1989: "A system for example-based programming", *Procs. of Human Factors in Computing Systems*.

[28] Pattis, R., 1981: *Karel the Robot*, John Wiley & Sons.

[29] Reiser, B.J., Anderson, J.R., and Farrell, R.G., 1985: "Dynamic Student Modelling in an Intelligent Tutor for Lisp Programming", *Procs. of the 9th International Joint Conference on Artificial Intelligence*.

[30] Rich, C., and Waters, R.C., 1988: "The Programmer's Apprentice: A Research Overview", *Computer*, 21(11).

[31] Röβling, G., and Freisleben, B., 2001: "The Extensible Visualization Framework ANIMAL", *Procs. of the 32nd SIGCSE Technical Symposium on Computer Science Education.*

[32] Röβling, G., and Freisleben, B., 2000: "Experiences in Using animations in Introductory Computer Science Lectures", *Procs. of the 31st SIGCSE Technical Symposium on Computer Science Education.*

[33] Röβling, G., and Naps, T., 2002: "A Testbed for Pedagogical Requirements in Algorithm Visualizations", *Procs. of the Annual Conference on Innovation and Technology in Computer Science Education*.

[34] Röβling, G., Schüler, M., and Freisleben, B., 2000: "The ANIMAL Algorithm Animation Tool", *Procs. of the Annual Conference on Innovation and Technology in Computer Science Education*.

[35] Satratzemi, M., Daglilelis, V., and Evagelidis, G., 2001: "A system for program

visualization and problem-solving path assessment of novice programmers", *Procs. of the Annual Conference on Innovation and Technology in Computer Science Education*.

[36] Stasko, J., 1990: "TANGO: A Framework and System for Algorithm Animation", *Computer*, 23(9).

[37] Stasko, J., Domingue, J., Price, B.A., Brown, M.H., 1998: *Software Visualization: Programming as a Multimedia Experience*, MIT Press.

[38] The Teaching Machine of C++: http://www.engr.mun.ca/~theo/TM.

Docencia de la Programación Orientada a Eventos

Carlos Rioja del Río
Dept. de Lenguajes y Sistemas Informáticos
Universidad de Cádiz
11003 Cádiz
e-mail: carlos.rioja@uca.es

Resumen

Se presenta una vía de enseñar programación orientada a eventos o dirigida por eventos basándose en el conocimiento de la programación orientada a objetos (POO), mostrando la orientación a eventos como una extensión de la orientación a objetos.

1. Introducción

La enseñanza de una nueva metodología de programación supone para el alumnado el esfuerzo de cambiar la manera de pensar a la hora de abordar un problema. Por todos es bien conocido la dificultad que supone modificar o ampliar los esquemas de razonamiento de una persona si tiene otros esquemas distintos bien arraigados. Es el caso de los alumnos de los primeros cursos que una vez han conseguido trabajar de manera eficiente con la metodología de programación estructurada, se enfrentan al aprendizaje de la programación orientada a objetos, lo cual les hace cambiar por completo su manera de ver el universo que les rodea. Este problema se acentúa si la nueva metodología a aprender es la de la programación lógica o la programación funcional.

En definitiva, cualquier cambio que ha de producirse no en el conocimiento, sino en las vías de adquisición de conocimiento, confieren un gran esfuerzo por parte del alumno y del profesor, y normalmente algunas de las metodologías impartidas nunca llegan a asimilarse por completo por parte del alumno debido a que el alumno tiende a abordar los problemas siempre desde los mismo puntos de vista, aunque la solución obtenida exceda en dificultad a otras soluciones alternativas. Por tanto, es conveniente minimizar las dificultades que pueda encontrar el alumno a la hora de conocer una nueva filosofía de programación. En algunos casos, es una tarea compleja, como en los comentados anteriormente, poco tiene que ver la programación estructurada con la programación lógica o la orientación a objetos. Sin embargo, a la hora de enseñar programación visual o programación orientada a eventos, no debemos presentarla al alumnado como una novedad absoluta, más bien, como una evolución de la orientación a objetos, buscando el aprendizaje por analogía. Se presentan a continuación las vías para conseguirlo.

2. Contexto y Motivación

En la Ingeniería Técnica en Informática de Gestión de la Universidad de Cádiz, la asignatura principal en la que se enseña el paradigma de la programación orientada a objetos es obligatoria y se imparte en segundo curso. Así mismo, la asignatura de desarrollo rápido de aplicaciones y por tanto, la programación orientada a eventos es optativa para los cursos segundo y tercero.

Nos encontramos entonces con la posibilidad de que el alumno que se enfrenta a la programación orientada a eventos ya posea conocimientos del paradigma orientado a objetos, o que se encuentre cursando las dos asignaturas a la vez. En el primer caso el profesor puede y debe apoyarse en los conocimientos de POO adquiridos para presentar las novedades de la orientación a eventos. Y en el segundo caso es necesario un seguimiento del avance en ambas asignaturas para reforzar, si cabe, el aprendizaje de las partes comunes de ambas metodologías, que son muchas e importantes.

Por supuesto, se podría enseñar programación orientada a eventos obviando cualquier relación

posible con la orientación a objetos, pero esto aumenta el esfuerzo a realizar por el alumno, que está recibiendo dos metodologías como completamente distintas, cuando una es pilar en el desarrollo de la otra.

3. El paradigma de la orientación a objetos.

En un sistema orientado a objetos, el software se organiza como un conjunto finito de objetos que contienen tanto datos como operaciones y que se comunican entre sí mediante mensajes [1].

Un alumno con conocimiento de la programación orientada a objetos debe ser capaz de distinguir las clases y objetos de un determinado problema y describir la importancia de la herencia en los programas orientados a objetos. Mediante la *herencia*, una clase puede heredar atributos y métodos de otra, a la que llamaremos superclase, de esta manera podemos formar jerarquías de clases.

La *encapsulación* y la *ocultación* de datos son características fundamentales que nos ayudan en el diseño de programas orientados a objetos. De esta manera, las clases encapsulan un conjunto de datos y operaciones comunes en un mismo nivel de abstracción. Gracias a la ocultación de datos podemos separar la información necesaria para la implementación del programa de la interfaz con la que trabajará el usuario.

Decimos que una clase que presenta interfaz para varias clases es de tipo *polimórfico* [2]; el polimorfismo es también uno de los elementos fundamentales de la POO.

4. La programación orientada a eventos

En las herramientas de desarrollo rápido de aplicaciones, la escritura de un programa se basa en el uso de componentes prediseñados que se colocan en formularios o ventanas y se ajustan a las necesidades del problema mediante el establecimiento de las propiedades de los componentes en cuestión.

En la programación orientada a eventos, la línea de ejecución del programa no está dictaminada de antemano, no conocemos cuáles serán las líneas de código que se ejecutarán en cada caso. Es el usuario o el sistema los que determinan la ejecución de las funciones como respuesta a eventos provocados por ellos.

Un evento es una acción que es reconocida por un objeto y que normalmente es provocada por el usuario al interactuar con la interfaz del programa (la pulsación de un botón del ratón, la pulsación de una tecla, etc.). Muchos objetos tienen predefinidos un conjunto de eventos que pueden reconocer, si uno de ellos ocurre, se ejecuta un manejador de evento (función) como respuesta, por lo tanto, una aplicación para el sistema operativo en realidad lo que hace es ejecutar funciones para tratar los distintos eventos que se vayan produciendo. De ahí viene la analogía con la POO, en un sistema dirigido por eventos, los componentes prediseñados que usamos pueden verse como objetos, pues tienen una serie de propiedades o atributos y un conjunto claramente definido de funciones miembro o métodos.

El flujo interno de una aplicación dirigida por eventos puede describirse a través de los siguientes pasos:

1. La aplicación carga en memoria y visualiza el formulario de inicio. (Este paso puede variar entre distintas herramientas RAD).
2. La aplicación espera hasta que se produzca un evento.
3. Se determina el tipo de evento producido.
4. Se determina el objeto afectado por el evento.
5. Se ejecuta el manejador de evento correspondiente.
6. La aplicación vuelve al paso 2.

Un alumno con conocimiento de la programación orientada a eventos debe ser capaz de distinguir, para cada problema particular, la interfaz gráfica del usuario, los componentes que la compondrán, las propiedades de los mismos, y los manejadores para los posibles eventos que puedan producirse en el sistema. El desarrollo de un programa orientado a eventos se simplifica en estos tres pasos. Sin embargo, el aprendizaje del entorno de desarrollo puede ser deficiente si no mostramos la orientación a eventos como una derivación de la orientación a objetos.

5. Establecimiento del paralelismo

Una vez el alumno conoce los conceptos fundamentales de la orientación a objetos tales como clase, objeto, polimorfismo, herencia, etc, es el momento de presentarle cómo en una herramienta de desarrollo rápido tenemos una serie de clases de las cuales se derivan subclases con los componentes que vaya a utilizar en cada aplicación.

Utilizaremos para ello como herramienta de desarrollo rápido Borland C++ Builder, cuyo lenguaje de programación embebido es C++. Una aplicación desarrollada con Borland C++ Builder, al igual que en el resto de lenguajes visuales, está formada por una serie de formularios que contienen componentes y responden a los eventos producidos en esos formularios. De esta manera, podemos considerar el formulario como el pilar fundamental sobre el que se asienta cualquier aplicación orientada a eventos. En el caso de Borland C++ builder, cada uno de los formularios se corresponde con tres archivos tales como Unit.cpp, Unit.h, Unit.dfm. El archivo .cpp contiene el código C++ con los manejadores de eventos necesarios para la correcta ejecución del programa. El archivo .h contiene inclusiones de otros ficheros de cabecera, declaraciones, prototipos de funciones, constantes, etc. Y el archivo .dfm tiene la descripción gráfica del formulario.

En este punto debemos destacar que el compilador coloca en el archivo .h mucha información de manera automática, y esta información suele pasar desapercibida para el alumno. Es en este caso en el que debemos mostrar la importancia de conocer realmente qué ocurre cuando diseñamos una aplicación con formularios y cómo éstos son objetos de una clase derivada de la clase TForml la cual nos proporciona el entorno de desarrollo.

Cuando estamos diseñando una aplicación, podemos observar en el archivo .h cómo ese formulario con el que estamos trabajando se implementa como un puntero a una clase derivada de TForm, esto ocurre de la siguiente manera:

```
class TForm1 : public TForm
{
__published:
private:   // User declarations
public:    // User declarations
     __fastcall
TForm1(TComponent* Owner);
};
//---------------------------
extern TForm1 *Form1;
```

Podemos observar en el código cómo automáticamente en el archivo .h se deriva la nueva clase TForm1, siendo Form1 el nombre del formulario que estamos diseñando. También se incluye una declaración de tipo externa para que el formulario pueda ser modificado por todos los archivos que incluyan este fichero de cabecera. La definición del objeto Form1 aparece en el fichero .cpp

Es importante evitar que el alumnado pase por alto estas inclusiones automáticas y se limite a seleccionar componentes y ubicarlos en los formularios. La realidad es bien distinta, cuando generamos nuevos formularios para nuestra aplicación, estamos derivando subclases de la clase madre TForm, proporcionada por Borland C++ Builder, y cuando se seleccionan componentes para incluirlos en un formulario, estamos añadiendo atributos a la clase derivada TForm1. A su vez, estos componentes son punteros a clases proporcionadas por el entorno de desarrollo y como tales, poseen propiedades y métodos.

6. En la práctica. Paso a paso.

Vamos a mostrar a continuación cómo a través de un ejemplo práctico podemos ir paso a paso mostrando aquellos aspectos del desarrollo rápido de aplicaciones que nos permiten establecer un símil con la programación orientada a objetos. Desarrollaremos una aplicación con un formulario que contiene dos componentes botones y un componente etiqueta que mostrará un cierto texto al usuario.

El primer paso es mostrar al alumno cómo en cuanto comenzamos a desarrollar la aplicación, el formulario principal que estamos usando es un puntero a una clase derivada tal y como hemos comentado en el punto anterior.

A partir de este momento la inclusión de los componentes en el formulario se corresponde con la inserción de atributos en la clase derivada TForm1. En la figura 1 observamos la interfaz gráfica de los componentes en el formulario y en la figura 2 el correspondiente archivo de cabecera generado automáticamente.

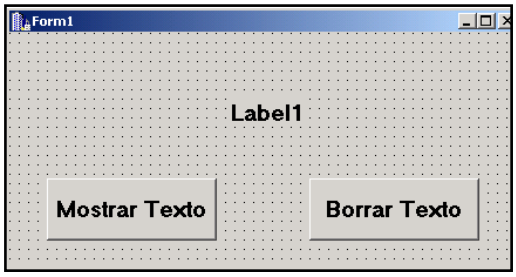


Figura 1. Formulario de la aplicación

Si observamos el código del fichero de cabecera correspondiente al formulario de la figura 1, podemos constatar cómo los tres componentes del formulario son punteros a clases ya existentes, y además, son atributos de la clase derivada a la que pertenece el formulario Form1.

```
   class TForm1 : public TForm
   {
   __published:
       TButton *Button1;
       TButton *Button2;
       TLabel *Label1;
   private:   // User declarations
   public:    // User declarations
       __fastcall
TForm1(TComponent* Owner);
   };
   //---------------------------
   extern TForm1 *Form1;
```

Figura 2. Clase derivada TForm1

El texto que presentan los botones corresponde con el establecimiento de su propiedad *Caption*, ésta propiedad puede establecerse en tiempo de diseño o mediante una instrucción de asignación C++. Por ejemplo, con el componente Button1, como se ve en la figura 2, es un puntero a la clase TButton, y la propiedad Caption, pertenece entre otras a ésta clase. Sólo queda asignarle el valor correspondiente:

```
Button1->Caption="Mostrar Texto";
```

El operador -> accede al objeto apuntado por Button1 y selecciona el atributo deseado, en este caso Caption. Análogamente debería realizarse con Button2.

El segundo paso nos mostrará cómo la escritura de los manejadores o gestores de evento de nuestro programa de ejemplo supone la inclusión de funciones miembro en la clase derivada TForm1. En este caso particular, los eventos que debemos contemplar son la ejecución de un click en cada uno de los botones, el primero de ellos mostrará un mensaje en el componente etiqueta, y el segundo borrará el texto de la etiqueta.

En el entorno de desarrollo debemos seleccionar el evento Click del componente botón Button1 pues queremos darle respuesta cuando se produzca, es decir, queremos escribir un manejador para ese evento. Una vez seleccionado escribimos código C++ con la instrucción correspondiente, la asignación del texto al atributo Caption de la Clase TLabel en este caso. Esta instrucción se realiza de manera similar en Button2.

```
   //---------------------------
   void __fastcall
TForm1::Button1Click(TObject *Sender)
   {
   Label1->Caption = "Jenui 2003";
   }
   //---------------------------
   void __fastcall
TForm1::Button2Click(TObject *Sender)
   {
   Label1->Caption = "";
   }
   //---------------------------
```

Figura 3. Archivo cpp con los manejadores de evento

La clase Tlabel, al igual que la clase TButton, posee entre sus atributos de clase la propiedad Caption. El borrado del texto se realiza en este caso asignando la cadena vacía a la propiedad Caption. Una vez hecho esto, el ejecutable generado espera a que el usuario haga click en alguno de los botones para ejecutar el gestor de

evento correspondiente. La funcionalidad se presenta en la figura 4.

Figura 4. Ejecución de la aplicación de ejemplo

Hasta aquí la realización del programa. Sin embargo, existen aspectos importantes que refuerzan la idea del paralelismo entre el paradigma de la orientación a objetos y el desarrollo de aplicaciones dirigidas por eventos. En la figura 3 se observa cómo las funciones manejadoras de eventos son en realidad funciones miembro de la clase TForm1, pues presentan el operador de resolución de ámbito (::) en la escritura del cuerpo. Además, accedemos al componente Label1 de igual manera a como se acceden a los atributos de una clase, pues como hemos comentado antes, los componentes son atributos de la clase. Este tipo de detalles son en principio transparente al usuario de la herramienta como al usuario final de la aplicación.

Resta comentar otro aspecto importante, sabemos que en C++ los prototipos de las funciones son obligatorios, sin embargo no hemos comentado nada al respecto y la aplicación funciona perfectamente. Esto es debido a que en cada archivo .cpp se incluye el .h correspondiente y el entorno de desarrollo ha incluido automáticamente los prototipos en el fichero de cabecera. ¿De qué manera se han incluido los prototipos? Los prototipos están incluidos como correspondientes funciones miembro de la clase derivada TForm1, así se comprueba en la figura 5.

```
class TForm1 : public TForm
{
__published:
    TButton *Button1;
    TButton *Button2;
    TLabel *Label1;
    void __fastcall
Button1Click(TObject *Sender);
    void __fastcall
Button2Click(TObject *Sender);
  private:   // User declarations
  public:    // User declarations
    __fastcall
TForm1(TComponent* Owner);
  };
  //---------------------------
  extern TForm1 *Form1;
```

Figura 5. Clase TForm1 con funciones miembro

Como se observa en la figura 5, la herramienta incluye los prototipos como métodos de la clase TForm1. El alumno debe conocer este aspecto pues de lo contrario nunca sabrá que está derivando una clase y añadiendo atributos y métodos mientras diseña una aplicación orientada a eventos.

7. Conclusión

Buscamos fomentar el aprendizaje por analogía. En la puesta en práctica de la idea planteada aquí, los alumnos asimilan mucho mejor los conocimientos porque pueden relacionarlos con otras materias recibidas, facilitando su comprensión. Uno puede aprender a desarrollar herramientas dirigidas por eventos ignorando los aspectos de la programación orientada a objetos que se han comentado. Sin embargo, es más sencillo para el alumnado con conocimientos del paradigma de la orientación a objetos saber que en además está derivando clases cada vez que genera un formulario, añadiendo atributos cada vez que selecciona un nuevo componente para ese formulario, y añadiendo métodos cuando escribimos un manejador de evento.

No sólo esto, es obligación del alumnado conocer al detalle la realidad de los programas que desarrolla, y obviar este punto de vista sería acotar la realidad. Una vez tenga suficiente soltura en el desarrollo de este tipo de aplicaciones, el alumno podrá abstraerse de este punto de vista como de cualquier otro, sin embargo para el aprendizaje y los primeros acercamientos a la programación dirigida a eventos, es una gran ayuda y un aspecto fundamental relacionarlo con la orientación a objetos.

Referencias

[1] Aburruzaga García, Gerardo; Median Bulo, Inmaculada & Palomo Lozano, Francisco. *Fundamentos de C++*. Servicio de Publicaciones, Universidad de Cádiz, 2001.

[2] Stroustrup, Bjarne. *El Lenguaje de Programación C++*. Addison-Wesley, 2001.

[3] Wu, C. Thomas. *Introducción a la programación orientada a objetos con Java.* McGraw Hill, 2001.

[4] Charte Ojeda, Francisco. *Programación con C++ Builder.* Anaya Multimedia, 1997

[5] Miano, John; Cabanski, Tom & Howe, Harold. *Borland C++ Builder How-To*. Wait Group Press, 1997.

[6] Calvert, Charles. *Borland C++ Builder Unleashed.* Sams Publishing, 1997.

JDESK: Simulador de Eventos Discreto Basado en Web [1]

Inmaculada García García, Ramón Mollá Vayá
Dpto. de Sistemas Informáticos y Computación
Universidad Politécnica de Valencia
46022 Valencia
e-mail: ingarcia@dsic.upv.es
rmolla@dsic.upv.es

Resumen

JDESK es un simulador generalista de eventos discreto que permite simular y obtener resultados de un modelo implementado en código Java. Es un simulador basado en web, por lo que puede utilizarse desde cualquier navegador. El modelo de simulación obtenido puede ejecutarse de forma local en cualquier plataforma. JDESK es un simulador versátil y con tiempos de simulación bajos. Los modelos de simulación se implementan definiendo los componentes del modelo y su interconexión. Por la facilidad de creación y modificación del modelo simulado puede usarse como prototipador. JDESK permite simular cualquier modelo y no impone restricciones al modelo a simular. Permite definir cualquier tipo de comportamiento del sistema, por caprichoso que sea. El simulador incluye un asistente para principiantes, que permite crear modelos de forma intuitiva. Incluye también bibliotecas de componentes del modelo y de ejemplos de modelos completos, que permiten guiar al alumno en el proceso de creación del modelo del sistema a simular.

1. Introducción

La simulación es una herramienta poderosa en el proceso de aprendizaje. El aprendizaje basado en simulación se define como "aprender haciendo" [11]. Permite al estudiante seleccionar experiencias y obtener sus propias respuestas. La simulación basada en web no es un campo nuevo dentro de la simulación, sino la necesidad de adaptar el campo de la simulación a las nuevas tecnologías [3]. Gran parte de los simuladores basados en web son versiones web de otros simuladores existentes.

La simulación puede utilizarse para resolver cualquier problema que pueda modelarse como un sistema de eventos discretos [13]. Algunas áreas de aplicación son [1]: procesos de producción, ingeniería de la construcción, diseño de semiconductores, aplicaciones militares, procesos de transporte y distribución, procesos de gestión,…

La simulación basada en web está permanentemente disponible desde cualquier navegador, lo que facilita el acceso de los estudiantes desde los laboratorios o incluso desde su propia casa, lo cual es una ventaja respecto a la simulación tradicional. Algunos simuladores disponen de bibliotecas y repositorios de modelos propios o aportados por los usuarios, lo que permite a los estudiantes crear modelos más complejos de forma sencilla o aprender a modelar sistemas mediante ejemplos. El simulador no debe estar disponible en local, lo que facilita su posibilidad de utilización y permite que esté siempre actualizado.

Una ventaja importante de la simulación en web es que permite construir simulaciones distribuidas (Distributed Interactive Simulation [9]), donde varios usuarios interaccionan. Los estudiantes pueden crear equipos de trabajo con estudiantes en diferentes ubicaciones, lo que enriquece su aprendizaje.

Existe un gran número de simuladores de eventos discretos basados en Web [12], como JSIM, SILK o SIMJAVA. JSIM [16] [17] es un simulador basado en web escrito en Java de código fuente abierto [6] [7]. Permite definir el

[1] Este trabajo ha sido financiado por la OCYT de la Generalitat Valenciana bajo el proyecto de investigación CTIDIB/2002/344.

modelo de forma gráfica, conectando nodos de diferente tipo. Incluye animaciones durante la ejecución de la simulación. Permite definir el modelo de forma modular y reutilizar componentes. SILK [4] es un simulador generalista escrito en Java como una biblioteca. Los modelos se escriben directamente en Java utilizando una biblioteca de clases. Permite definir los modelos de forma gráfica y modular, aunque también de forma textual. SIMJAVA [5] es un simulador de eventos discreto basado en SIM++ de código abierto [8]. Permite representar los objetos de la animación como iconos animados. Una simulación es un conjunto de entidades ejecutándose independientemente y comunicándose mediante eventos.

Entre los simuladores no basados en web hay dos simuladores utilizados en docencia y con dos filosofías completamente diferentes, que los hacen especialmente interesantes por su diferente forma de descripción del modelo y operación. SMPL [14] es una librería de C, lo que permite utilizar todo el potencial de este lenguaje en la implementación de los modelos. Destaca por su velocidad de simulación, pero su gran inconveniente es que la definición del modelo es complicada y trabajosa, pues supone definir el modelo como la secuencia de todos los posibles eventos del sistema. Deben definirse los algoritmos de planificación, el control de tiempos,... Modificar y depurar el modelo es complicado. No permite modelar cualquier sistema: existe un valor límite del número de clientes y de Estaciones de Servicio (ES) [2] en el modelo. QNAP [18] es un simulador implementado como un lenguaje propio. Definir un modelo en este simulador es sencillo y rápido, pues únicamente deben definirse las ES del modelo, sus propiedades y su interconexión. Modificar y depurar el modelo es rápido y sencillo. Los algoritmos de planificación y el control de tiempos los realiza el simulador automáticamente. No impone limitaciones en el modelo a simular. Por contra, el tiempo de simulación es entre 4 y 6 veces mayor que SMPL.

2. Objetivos

JDESK [10] se ha diseñado para aunar las siguientes características, que le permiten adaptarse a las necesidades de la docencia, integrando lo mejor de cada simulador:

- Creación rápida y fácil del modelo de simulación, El modelo del sistema es una descripción de los elementos que lo componen, siguiendo el estilo de modelado de sistemas de QNAP.
- Fácil depuración y modificación, debido al estilo de modelado similar a QNAP.
- Multiplataforma, pues el modelo es código escrito en Java.
- El modelo debe estar implementado en un lenguaje fácil de aprender y que ofrezca la posibilidad de integrar elementos externos a la simulación. En JDESK el modelo de simulación está implementado en Java.
- Posibilidad de simular cualquier modelo, incluso con comportamientos no convencionales. SMPL permite la implementación de cualquier comportamiento del sistema, pero con un coste de implementación muy elevado. JDESK permite implementar comportamientos caprichosos de forma sencilla y modular.
- Creación modular del modelo de simulación:
  - Reutilización de componentes de otros modelos o del propio modelo, debido a la utilización de objetos de Java.
  - Soporte a simulación distribuida, permitiendo crear equipos de alumnos trabajando sobre un mismo modelo, por la utilización de Java como lenguaje de implementación del modelo.
  - Creación de un repositorio de ejemplos y componentes de modelos, mediante bibliotecas escritas en Java.
  - Compartir el conocimiento. Los modelos escritos por un usuario pueden ser utilizados por otros usuarios como parte del sistema simulado.

Los simuladores basados en web no reúnen todas las características requeridas para su utilización en docencia. Estos simuladores permiten definir los modelos a simular de una forma más o menos intuitiva, dependiendo del simulador en concreto. Los simuladores más complejos ofrecen completos manuales de usuario. Permiten simular sistemas con comportamientos predeterminados que pueden ser adecuados para simular la mayor parte de los sistemas. Si embargo, no ofrecen el marco para

simular de forma fácil e intuitiva sistemas con comportamientos caprichosos.

El objetivo principal JDESK fue crear un simulador apto para su uso en cualquier entorno, y especialmente en el campo de la docencia. Aúna todas las características deseables en un simulador dedicado a la docencia. En cuanto a la forma de operatividad y velocidad, el diseño de JDESK se basó en SMPL y QNAP. JDESK permite definir el modelo siguiendo el estilo marcado por QNAP (definiendo únicamente las ES y su interconexión), de forma fácil y rápida. La velocidad de simulación supera con creces la obtenida por SMPL en la mayor parte de los modelos simulados.

En los siguientes apartados se da una visión general de la forma interna de operación de JDESK y de su utilización por parte del usuario.

3. Características de JDESK

JDESK es un simulador generalista de eventos discretos orientado a eventos. Está implementado en Java, lo que lo hace especialmente apropiado para su ejecución vía web [15].

Permite crear modelos describiendo la topología del sistema y las características de cada uno de sus elementos. Los modelos, por tanto, son fáciles de depurar y modificar. Esta característica lo hace adecuado para cualquier etapa del proceso de simulación: como prototipador o para obtener resultados finales.

JDESK incluye un asistente que permite guiar al alumno en el proceso de creación del modelo. Para modelos con comportamientos usuales o para usuarios no expertos se recomienda el uso del asistente. El asistente está implementado como un Applet de Java.

Los modelos puede describirse utilizando orientación a objetos y metodologías top-down, lo que hace posible la reutilización de partes del modelo en el propio sistema simulado o la creación de bibliotecas de componentes de modelos de simulación. La posibilidad de crear bloques de simulación contenidos en bibliotecas permite reutilizar el esfuerzo de otras personas en la definición de modelos y, además, crear simulaciones distribuidas entre un equipo de personas, de forma que los módulos desarrollados por los diferentes componentes del equipo puedan enlazarse para obtener la simulación de un sistema complejo.

El simulador incluye una serie de comportamientos predefinidos de sistemas, de forma que cualquier sistema convencional pueda modelarse fácil y rápidamente. También ofrece al usuario la posibilidad de definir comportamientos más sofisticados; por ejemplo, políticas de planificación no convencionales que incluso varíen dependiendo de las características del cliente actual o del estado del sistema. El control de la simulación puede realizarse en cualquier parte del sistema, dependiendo de cualquier evento. Las características del modelo, e incluso su topología, pueden variar dinámicamente en tiempo de simulación. Pueden crearse o destruirse ES dinámicamente, cambiar su interconexión o sus características. Permite la utilización de características avanzadas de modelos como semáforos, recursos o creación de hijos. En resumen, JDESK ofrece el marco para simular cualquier sistema que el usuario sea capaz de definir.

Las características del simulador lo hacen especialmente adecuado para realizar simulaciones en tiempo real, pues el usuario puede conocer en todo momento que es lo que está ocurriendo en el sistema. Esta facilidad también permite depurar sistemas complejos o demostrar como funciona el sistema simulado a los alumnos.

En JDESK los modelos se definen escribiendo de forma textual el modelo del sistema a simular en Java. El asistente de JDESK automatiza el proceso de escritura del modelo, pero el resultado siempre es un fichero java. Si el usuario ha definido el modelo de forma correcta, el simulador permite obtener el fichero Java del modelo para ejecutar la simulación en local. Las ventajas de escribir el modelo en Java para el proceso de aprendizaje son las siguientes:

- El estudiante no debe aprender un lenguaje específico para la simulación.
- Pueden utilizarse funciones de Java en la simulación no pertenecientes al simulador. El estudiante puede utilizar objetos implementados para otros propósitos en la implementación del modelo.
- La principal ventaja de utilizar Java como lenguaje de implementación del modelo es que el código implementado en Java es

multiplataforma. Para su utilización en enseñanza esta característica tiene grandes ventajas, pues funciona independientemente de la ubicación de los alumnos. El profesor puede implementar el modelo a simular y tener la certeza de poder usarlo para demostrar a los alumnos el funcionamiento del modelo simulado, independientemente de que tenga que usarlo en diferentes plataformas. No es necesario volver a acceder al simulador.

4. Funcionamiento del Núcleo de JDESK

Un modelo en JDESK está basado en dos entidades básicas: ES y clientes. Estas son las estructuras con las que el programador construye los modelos de simulación. Una ES modela una parte del sistema real que proporciona un determinado servicio. Está compuesta de una serie de servidores y una cola de clientes esperando ser servidos. Toda la información del modelo la contienen las ES. Los clientes son elementos de la simulación que viajan a través del sistema, cambiando su estado y el de las ES que atraviesan. Los clientes se pueden generar automáticamente mediante fuentes o bien el programador puede crearlos explícitamente.

La dinámica del sistema la modelan las ES que componen el modelo y la cola de eventos.

La cola de eventos es una estructura interna de JDESK y no visible al usuario. Está compuesta de clientes. Los clientes sirven de soporte para modelar los eventos del sistema. En JDESK sólo hay un posible evento: un cliente debe abandonar la ES donde está recibiendo servicio porque ha finalizado su tiempo de servicio. La cola de eventos está ordenada por el tiempo de finalización de servicio de los clientes.

La simulación comienza porque un cliente entra en una ES (figura 1 a). Si la ES tiene un servidor libre, comienza a recibir servicio (figura 1 b). Se genera un evento de salida del cliente al cabo de un tiempo determinado (figura 1 c).

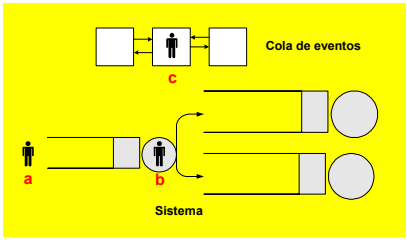


Figura 1. El cliente entra en la ES

Hasta este momento la simulación ha estado controlada por la ES donde ha entrado el cliente y ahora el control pasa a la cola de eventos. La cola de eventos ejecuta el siguiente evento, eliminándolo de la cola (figura 2 a). Sólo hay un posible evento: el fin del tiempo de servicio de un cliente. La cola de eventos invoca a la ES correspondiente para que libere al cliente. El cliente deja la ES y entra en otra ES o sale del sistema. El control de la simulación pasa a la ES destino. En la ES destino vuelve a comenzar el proceso. Si a la llegada de un cliente a la ES, no hay servidores libres, se inserta en cola de espera y no se genera ningún evento (figura 2 b).

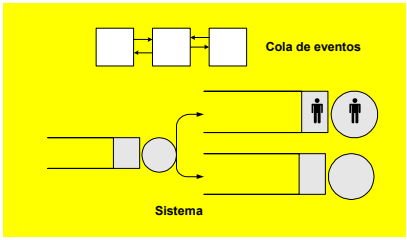


Figura 2. El cliente sale de la ES

Cuando no hay eventos en la cola de espera la simulación termina (aunque ha podido terminar con anterioridad por cualquier otra causa: ha finalizado el tiempo de simulación definido por el usuario o explícitamente el usuario ha decidido finalizar, dependiendo del estado del sistema o de algún evento en concreto).

El núcleo de simulación recorre la cola de eventos, extrayendo el evento en cabeza (es una cola ordenada por tiempo) e invocando a la ES correspondiente (figura 3). El control de la simulación pasa alternativamente de las ES a la cola de eventos.

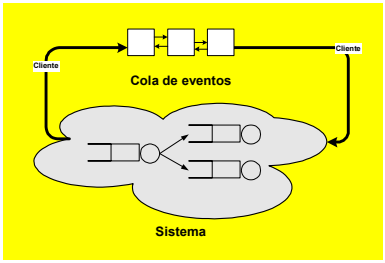


Figura 3. Dinámica del sistema

5. Modelado de Sistemas con JDESK

El modelo de un sistema en JDESK se crea, básicamente, definiendo las ES que componen el modelo.

5.1. Implementación de ES

JDESK define tres tipos de ES:

- Fuentes: tienen como objetivo principal generar clientes dinámicamente. La cadencia con la que la fuente genera clientes y las propiedades de estos, los marca el programador al definir la fuente. Una vez los clientes se crean, dejan la fuente para encaminarse a otra ES.
- Servidores: está compuesto de una cola de espera y una serie de servidores. Cuando un cliente entra en una ES, si no hay un servidor libre, espera en la cola de la ES. Si el cliente entra en un servidor, permanecerá en él recibiendo servicio durante un tiempo determinado.
- Recursos: un recurso en JDESK aúna recursos y semáforos. Un recurso está compuesto por una cola de espera y una serie de recursos que puede reservar el cliente. Si a la llegada del cliente, el número de recursos demandados no está disponible, el cliente espera en la cola a que se liberen recursos suficientes. Si los recursos demandados por el cliente se le pueden asignar, el cliente continúa fluyendo libremente por el sistema. El cliente puede liberar los recursos en cualquier momento.

El comportamiento de la ES se define mediante una serie de parámetros. Estos parámetros varían dependiendo del tipo de ES. En la tabla 1 se muestran los parámetros de la ES, su tipo y cuales de ellos están presentes en la declaración de cada ES. Entre estos parámetros hay:

- Constantes: el valor del parámetro debe obligatoriamente seleccionarse entre una serie de valores constantes.
- Variables: permite cualquier valor del tipo correcto.
- Funciones: se debe pasar como parámetro una función previamente definida. Existen funciones de biblioteca en JDESK para algunas de estas funciones.

PARÁMETROS DE LA ES	SERVIDOR	RECURSO	FUENTE	TIPO DE PARÁMETRO
Nombre	✓	✓	✓	String (variable)
Número de servidores o recursos	✓	✓		Valor (variable)
Política de servicio preemtiva	✓			Constante
Tiempo de servicio demandado por los clientes	✓	✓	✓	Función
Algoritmo de planificación (inserción en cola de espera de la ES)	✓	✓		Función
Algoritmo de planificación (reinserción en cola de espera de la ES)	✓			Función
Número de recursos demandados por el cliente		✓		Función
Algoritmo de encaminamiento de clientes a la salida de la ES	✓	✓	✓	Función
Función de usuario	✓	✓		Función

Tabla 1. Parámetros de definición de ES

El comportamiento de las ES se define en su mayor parte por funciones. Cada una de estas funciones tiene un cometido y se pone en funcionamiento en un instante determinado de la simulación. El usuario puede implementar las funciones o bien utilizar funciones de biblioteca suministradas por JDESK.

Para usuarios inexpertos o modelos con comportamientos típicos, es recomendable usar funciones de biblioteca. Cuando el sistema tiene un comportamiento inusual se debe implementar este comportamiento mediante estas funciones. Las funciones de comportamiento dan potencia y flexibilidad al simulador.

Pueden utilizarse, además de para su cometido específico, para cualquier otro que el usuario considere apropiado.

Permiten variar dinámicamente el sistema: su topología, su estructura o sus características. También son estas funciones las que permiten hacer simulaciones en tiempo real, incluir multimedia, alarmas,...

La función de tiempo de servicio obtiene el tiempo que un cliente determinado debe recibir servicio en la ES actual. El tiempo de servicio asignado podría depender de su estado, de valores aleatorios, de distribuciones temporales (las principales funciones de distribución temporal están implementadas en el simulador como funciones de biblioteca: uniforme, exponencial, erlang, hiperexponencial y normal).

La función de planificación para inserción en cola de espera permite insertar en cola de espera de la ES un cliente que no puede servirse porque todos los servidores están ocupados. Las principales políticas de servicio están implementadas en el simulador como funciones de biblioteca: lifo, fifo, según prioridades y las versiones preemptivas de estas políticas.

La función de planificación para reinserción en cola de espera (sólo para ES preemtivas), con la llegada de un cliente más prioritario, el cliente que está recibiendo servicio actualmente deja de recibir servicio y se reinserta en la cola de espera. El hecho de diferenciar entre inserción y reinserción permite dotar de una mayor flexibilidad al simulador.

La función de encaminamiento define la ES donde transitará el cliente cuando abandone la ES actual. El conjunto de las funciones de encaminamiento define la topología del sistema. Usar funciones para definir la topología del sistema permite variarla dinámicamente.

La función de obtención del número de recursos permite obtener el número de recursos que solicita un cliente cuando llega a una ES tipo recurso. El número de recursos demandados podría modificarse dinámicamente dependiendo de algún criterio.

La función de usuario no es necesaria para el funcionamiento del simulador, pero, se ha incluido en JDESK, pues dota al simulador de una gran flexibilidad. Esta función se ejecuta cuando un cliente comienza a recibir servicio en una ES. Puede servir para hacer trazas, para simulaciones en tiempo real,... es decir, para cualquier comportamiento atípico.

Estas funciones son métodos de determinadas clases definidas a tal efecto.

5.2. Pasos en la creación del modelo

Para definir un modelo en JDESK se deben seguir los siguientes pasos:

- Inicializar el simulador
- Definir las ES del modelo:
  - Implementar las funciones de comportamiento de cada ES o usar funciones predefinidas o implementadas para otras ES.
  - Determinar las características de la ES.
- Introducir clientes en el sistema. Para sistemas cerrados o para sistemas abiertos con comportamientos determinados, es necesario comenzar la simulación con clientes en el sistema. La creación de un cliente supone insertarlo en una ES.
- Comenzar la simulación. Se puede indicar la forma de finalización de la simulación. La forma más usual de finalizar la simulación es indicar el tiempo máximo de la simulación. También, permite comenzar a obtener medidas de la simulación en un instante determinado para eliminar estados transitorios.
- Obtener resultados de la simulación.

5.3. Clases de JDESK para la Simulación

JDESK proporciona al usuario biblioteca de clases para la simulación. Los métodos de estas clases

permiten: control de la simulación (simulación, inicialización, finalización, inicio de contabilidad y gestión de errores), creación de ES de cada uno de los tipos, gestión de hijos, gestión de recursos y contabilidad. Estos métodos pueden usarse dentro de las funciones de comportamiento de la ES con el objeto de dotar al sistema de comportamiento específicos.

5.4. Utilización de JDESK

JDESK es un simulador basado en texto, por lo que el modelo simulado es código escrito en Java con funciones de biblioteca específicas del simulador.

JDESK contiene un editor de texto que permite escribir el código del modelo de simulación. Permite editar modelos creados anteriormente o ejemplos de modelos completos del repositorio de JDESK. El código debe tener el formato de un programa principal de Java.

Para usuarios inexpertos es aconsejable usar el asistente (figura 4). El asistente es un Applet de Java que permite crear modelos de forma rápida y sencilla mediante controles, escribiendo únicamente algunas funciones de comportamiento. El asistente construye el código Java del modelo de simulación a partir de las especificaciones del usuario.

Figura 4. Asistente

El asistente permite:

- Definir los parámetros generales de la simulación, como tiempos de inicio y final de la simulación.
- Añadir funciones al modelo. El asistente permite ver ejemplos de funciones de la biblioteca de JDESK. Si el usuario elige escribir directamente la función, aparece una ventana de edición con un esqueleto de la función (en forma de método de la clase correspondiente), de forma que el usuario sólo debe escribir el código estrictamente necesario. Esta función se podrá asignar posteriormente a una ES.
- La inserción de una ES en el modelo simulado utilizando el asistente se realiza utilizando los controles de la ventana de inserción de ES (figura 5). El asistente permite seleccionar el tipo de ES y según el tipo seleccionado habilita únicamente los controles correspondientes. Permite definir las funciones de comportamiento de la ES: seleccionándolas entre las previamente definidas, para ésta o para otras ES, o entre las funciones de biblioteca o bien escribir la función directamente. Si se escribe la función muestra un esqueleto de la clase correspondiente.
- Insertar clientes: permite crear clientes con unas determinadas características e insertarlos en una ES previamente definida.
- Obtener resultados: permite indicar que resultados de la simulación se desea obtener y en que formato.

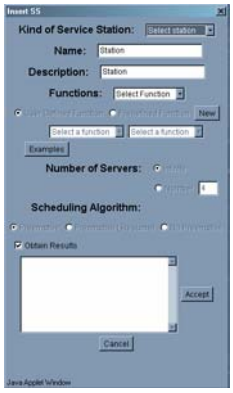


Figura 5. Creación de una ES utilizando el asistente

El código creado mediante el asistente aparece en el editor de texto y puede ser modificado para ajustar o variar su comportamiento.

Una vez se ha definido el modelo, se compila utilizando para ello un control de JDESK. Si la compilación produce errores se mostrarán los

mensajes correspondientes en la ventana de mensajes. Si la compilación tiene éxito, el fichero conteniendo el modelo de simulación queda disponible para que el usuario lo ejecute de forma local. Este modelo es un fichero Java.

6. Conclusión

JDESK es un simulador generalista de eventos discreto escrito en Java. El modelo de simulación se implementa describiendo la topología del sistema: las entidades del sistema, su comportamiento y su interconexión. JDESK facilita la definición, depuración y modificación de modelos, lo que lo hace especialmente apropiado para su utilización en docencia. JDESK no impone restricciones en el modelo a simular y permite definir cualquier comportamiento del sistema. Permite incluir elementos externos dentro de la simulación, como multimedia o alarmas y simulación en tiempo real. Las características del sistema pueden variar dinámicamente, incluso su topología. Permite crear o destruir dinámicamente elementos del modelo de simulación.

JDESK incluye un asistente para ayudar a los estudiantes a definir modelos con comportamientos típicos, de forma fácil y rápida. El asistente guía al alumno durante todo el proceso de creación del modelo, permitiéndole definirlo controles.

El modelo de simulación está implementado en Java, por lo que, una vez compilado por JDESK, puede ser ejecutado en diferentes plataformas. Además, el alumno no debe aprender un lenguaje específico para utilizar el simulador.

JDESK es un simulador basado en web, lo que permite su acceso desde cualquier navegador. Tanto alumnos como profesores pueden acceder a JDESK desde laboratorios o desde su propia casa.

Referencias

[1] Coos, R. *Simulación: un enfoque práctico*. Limusa, Mexico.1992

[2] Fishman, G.S. *Conceptos y Métodos en la Simulación Digital de Eventos Discretos*. Limusa, 1978.

[3] Fishwick, P.A. *Web-Based Simulation: Some Personal Observations*. 28th Winter simulation conference, California, 1996.

[4] Healy, K.J. Kilgore, R.A. *Introduction to Silk and Java-Based Simulation*. 30th Winter simulation conference.1998.

[5] Howell, F. McNab, R. *Simjava: a Discrete Event Simulation Package for Java with Applications in Computer Systems Modelling*. First International Conference on Web-based Modelling and Simulation. 1998.

[6] http://chief.cs.uga.edu/~jam/jsim/

[7] http://chief.cs.uga.edu/~jam/home/theses/huang_thesis/userguide/userguide.html

[8] http://www.dcs.ed.ac.uk/home/hase/simjava/

[9] http://www.sei.cmu.edu/publications/articles/arch-dist-int-sim.html

[10] http://www.sig.upv.es/proyectos/simulacion/JDESK.htm

[11] Kindley, R. *The Power of Simulation-Based e-Learning*. The e-Learning Developer's Journal. 2002. www.eLearningGuild.com

[12] Kuljis, J. Paul, R.J. *A Review of Web Based Simulations: Whither we wander?* 2000 Winter Simulation Conference.

[13] Law, A.M. Kelton, W.D. *Simulation Modeling and Analysis*. McGraw-Hill Series in Industrial Engineering and Management Science. 1982.

[14] MacDougal, M.H. *SMPL - A Simple Portable Simulation Language*. Amdahl. 1980.

[15] McNab, R. Howell, F.W. *Using Java for Discrete Event Simulation*. 12th UK Computer and Telecommunications Performance Engineering Workshop (UKPEW), 1996.

[16] Miller, J.A. Nair, R.S. Zhang, Z. Zhao, H. *JSIM: A Java-Based Simulation and Animation Environment*. 30th Annual Simulation Symposium. 1997.

[17] Miller, J.A. Seila, A.F. Xiang, X. *The JSIM Web-Based Simulation Environment. Future Generation Computer Systems*. 17-2. 2000.

[18] *QNAP: queuing network analysis package*. 1st Int. Conf. on the numerical Solutions of Markov chains. 1990.

Terminales sin disco basados en GNU/Linux para docencia

David Úbeda, Katja Gilly, Salvador Alcaraz, Guillermo Martínez
Dpto. de Física y Arquitectura de Computadores
Universidad Miguel Hernández
03202 Elche (Alicante)
{ubeda,katya,salcaraz,g.martinez}@umh.es

Resumen

El avance hardware y software de los últimos tiempos ha sido espectacular. El nuevo software docente que se utiliza en nuestros laboratorios, obliga a una actualización continua a nivel hardware de los computadores, incluso llegando al extremo de tener que sustituir equipos por otros más potentes.

Por otra parte, los nuevos planes de estudio, con una carga docente práctica muy elevada, han provocado un uso intensivo de los laboratorios de informática, lo que origina una degradación progresiva del rendimiento de las máquinas, y obliga al personal técnico a realizar trabajos de mantenimiento, con costes elevados.

Tradicionalmente, hemos utilizado software bajo licencia de uso, que aunque con las ventajas de uso docente, también incrementan los costes de instalación y mantenimiento de las aulas de informática.

En este proyecto, planteamos una solución alternativa a los tradicionales laboratorios de informática basados en una configuración de máquinas monousuario. Nuestra propuesta nos permite obtener unas ventajas en ahorro económico, tanto en costes de adquisición y mantenimiento hardware y software, como costes de mantenimiento de personal técnico, así como una mejora del uso, gestión y administración del laboratorio. La solución consiste en la utilización de terminales sin disco duro bajo sistema operativo GNU/Linux [1]. Paralelamente, proporcionamos un aprendizaje al alumno de dicho sistema operativo y de las herramientas de trabajo que lo complementan, y el cuál se está imponiendo en los entornos tanto universitarios como empresariales, afianzándose en las diferentes áreas de Informática, Telecomunicaciones e Ingeniería en general.

1. Introducción

Uno de los objetivos históricos en las grandes multinacionales del sector tecnológico ha sido intentar comercializar equipos informáticos sin disco duro, con muy poca memoria y con un procesador de baja potencia, administrados desde un servidor, dado que es muy casual que un usuario tipo, sin unos conocimientos extensos en informática, consiga aprovechar el máximo rendimiento que le proporciona su máquina.

Estas empresas, incluso invirtieron un capital importante apostando por tal proyecto y algunas de ellas obtuvieron grandes pérdidas económicas, sin citar la gran pérdida de esfuerzo humano. Una de las primeras a comercializar dicho producto fue IBM, que tuvo que abandonar la idea rápidamente al ver como las ventas de este producto no aumentaban tan rápido como ellos esperaban.

Las principales causas para que tal proyecto no se llevara a cabo satisfactoriamente, fueron principalmente:

- La bajada de precios de los ordenadores personales (PC), apareciendo máquinas clónicas mucho más económicas que las que fabricaban gigantes como IBM.
- Internet por aquella época no se encontraba tan extendido como lo está en la actualidad.
- Un servidor potente, capaz de soportar algunas decenas de clientes terminales, era inalcanzable en cuanto a coste se refiere.
- Los ordenadores clónicos siguieron su descenso de precio hasta equipararse a los terminales sin disco, por lo que los compradores se decantaban por los primeros.

Aprovechando el fuerte crecimiento de Internet durante los años 1996 y 1997, varias fueron las empresas que retomaron de nuevo el proyecto de los terminales, creyendo que era la oportunidad esperada, aunque algunas de ellas decidieron apartarse de dicha empresa a tiempo y no perdieron un capital importante. La primera en lanzarse al abismo fue Oracle[2], que incluso fabricó algunas máquinas de prueba con un sistema operativo creado por ellos mismos.

Una vez dentro del ámbito universitario, la pregunta que nos planteamos es: *¿qué hace que apostemos desde nuestro área por tal proyecto?*

Nosotros, desde nuestra posición, buscábamos utilizar material docente de uso libre para impartir nuestras asignaturas, ya que de esta manera, nuestros alumnos serian capaces de poderse llevar el software a sus hogares y poder instalarlo sin ningún tipo de problema ni restricción alguna. Además de por este motivo, apostamos por el uso de GNU/Linux y por el software libre en general, ya que pensamos que abre infinitas puertas e ideas a nuestra imaginación, al menos muchas más que otro tipo sistema operativo cerrado propietario.

Una vez que disponíamos de un entorno estable y que nos garantizase un éxito en la impartición de nuestra docencia, decidimos adecuar tales necesidades a un ahorro económico por parte de nuestra área, y a un punto de partida totalmente distinto de lo que habíamos visto hasta ahora.

Tal apuesta no nos ha salido nada mal, ya que poco a poco hemos conseguido ir avanzando en proyectos con un nivel de escalabilidad bastante aceptable, gracias al uso de terminales.

Se decidió llevar a cabo dicho proyecto con asignaturas que encajaran perfectamente dentro de la filosofía que pretendíamos proponer, y así optamos por impartir todas las clases prácticas en un aula, que diseñamos previamente, dotada de terminales sin disco y utilizando un servidor central. Es tal la versatilidad que nos proporciona, que somos capaces de impartir la mayoría de docencia práctica de nuestras asignaturas del Área de conocimiento, entre las que cabe destacar las siguientes:

- *Telemática*: en la que usamos simuladores de redes totalmente libres y de uso gratuito, como pueda serlo CNET [3].
- *Complementos de telemática*: en la cual nos basamos en dos aplicaciones: SPIN[4] (Simple Promela Interpreter) y XSPIN. La primera de ellas en modo texto y la segunda en modo gráfico (X-Window).
- *Fundamentos de Sistemas Operativos*: donde utilizamos software docente para análisis de llamadas al sistema en Linux y con conexión remota a servidor, o incluso local.
- *Programación en C y C++:* venimos usando entornos IDE basados en también en Linux para un entorno X-Window, además de los compiladores y linkadores necesarios.
- *Programación Concurrente*
- *Programación Multihilo*
- *Programación basada en Sockets*
- *Programación Java*[5]: Al igual que en C y C++ venimos utilizando entornos IDE de varias empresas, como Forte de Sun Microsystems, además de JDK[6] y otras herramientas.

Seguidamente vamos a tratar con cierta profundidad la problemática actual del mantenimiento de aulas de informática, y como la migración al sistema operativo GNU/Linux, nos ha ayudados a resolver tales problemas, hasta llegar al uso de terminales sin disco basados en dicho sistema.

1.1. El concepto de equipo obsoleto

El primer problema de un aula de informática surge cuando necesitamos instalar un software docente más o menos actual que va a consumir unos recursos bastante importantes en la máquina y que normalmente ésta no es capaz de rendirlos. Este motivo, generalmente, hace que nos planteemos un cambio de equipos por otros más modernos, con un microprocesador más potente, con más memoria RAM y con un disco duro más rápido y de mayor capacidad.

En la actualidad, en la mayoría de universidades, seguimos utilizando, mayormente, software docente basado en Microsoft Windows, a veces por comodidad, porque conocemos exhaustivamente ese software y no deseamos probar algún otro similar basado en otro sistema operativo, o porque no poseemos conocimientos en otros sistemas operativos, o es probable que no conozcamos la existencia de software que realice las mismas funciones que la versión de Microsoft Windows, y por tanto usamos el software basado

en este Sistema Operativo que consumirá bastantes recursos de nuestra máquina e incluso hará que nos planteemos un cambio de la misma.

Estos son algunos de los principales motivos por los que creemos que nuestros equipos antiguos se han quedado obsoletos.

Existen varias soluciones a este problema, entre las que cabe destacar dos, que suelen ser las más usadas por los responsables de aulas de informática de la propia universidad, o por los Técnicos de Laboratorio:

- La primera de ellas, a priori es la más sencilla, pero también es la más cara, y es comprar equipos nuevos que sustituyan a los antiguos.
- La segunda de ellas es proponer la migración de software a otro sistema operativo que nos plantee más ventajas, incluso sin renovar los equipos. Este sistema operativo podría ser GNU/Linux, que actualmente tiene un crecimiento bastante importante y que posee ciertas ventajas frente a cualquier otro sistema operativo propietario.

Llegados a este punto, podemos pensar que adoptando la segunda solución -la migración a GNU/Linux- se soluciona en parte la problemática de que los equipos puedan quedarse obsoletos. Esto en parte es cierto, pero no del todo, ya que aunque generalmente las herramientas basadas en GNU/Linux suelen consumir menos recursos que las basadas en Microsoft Windows, puede no ser suficiente y nuestros equipos antiguos aun así se queden obsoletos. Al menos, hasta aquí, nos quedaremos con que podemos disminuir considerablemente el consumo de recursos de nuestras máquinas, pero que probablemente no sea una solución factible a posteriori.

1.2. Coste elevado de Licencias

Otro de los problemas con el que nos encontramos en un aula, es el elevado coste de las licencias de software docente, teniendo que recurrir incluso a licencias de tipo Campus [7] para instalar dicho software en varias aulas de informática, con el gran desembolso que supone tales licencias por parte de la universidad.

En GNU/Linux, en cambio, nos encontramos con que la mayor parte del software es libre, y con libre nos estamos refiriendo, como bien dice la comunidad GNU, a:

- La libertad de correr el programa, con cualquier propósito (libertad 0).
- La libertad de estudiar como funciona el programa, y adaptarlo a sus necesidades (libertad 1).El acceso al código fuente es una precondición para esto.
- La libertad de distribuir copias de manera que se puede ayudar al vecino (libertad 2).
- La libertad de mejorar el programa, y liberar las mejoras al publico de tal manera que toda la comunidad se beneficia. (libertad 3). El acceso al código fuente es una precondición para esto.

En definitiva, no debemos pagar o pedir ningún tipo de permiso.

1.3. Restauración de equipos

Como sabemos, periódicamente, el disco duro de los ordenadores de las aulas de informática se restaura con una imagen que previamente se creó y replicó en todas y cada una de las máquinas. Este problema podemos solucionarlo mediante el uso de servidores, en los cuales se instala el software necesario una sola vez, y no es necesario instalarlo en los clientes, con la consiguiente comodidad y ahorro de tiempo.

Es latente el ahorro que supone no tener que pagar licencias de programas que se dedican a la replicación de imágenes de discos duros o particiones, e incluso, alargamos notablemente la vida de nuestros discos al no tener que restaurarlos periódicamente.

Imagínense pues, si además de que se consigue reutilizar los equipos antiguos, que nos ahorrásemos el dinero en licencias y que no necesitáramos disco duro, porque vamos a trabajar por red y con la memoria RAM.

2. Terminales Sin Disco basados en GNU/Linux

2.1. ¿Que es un terminal sin disco?

A grandes rasgos, podríamos definir un terminal sin disco como un conjunto de hardware que carga su sistema operativo por red y trabaja contra un servidor, o incluso se le podría instalar un disco duro para que además de trabajar contra un servidor, pudiese lanzar aplicaciones locales para evitar que todas se ejecuten en el servidor.

2.2. Ventajas de un terminal sin disco

Voy a pasar a resumir brevemente las numerosas ventajas que posee la utilización de terminales sin disco bajo GNU/Linux:

- Como ya hemos dicho, una de las principales ventajas es que estamos utilizando continuamente software libre, bajo licencia GPL [8], por lo que el coste es bajo e incluso nulo. Éste tipo de software libre, posee un desarrollo abierto, y podemos disponer del código fuente del mismo en cualquier momento.
- El proyecto esta pensado para multipuesto, con todas las ventajas que ello conlleva. Imaginemos por un momento varias aulas de este tipo en nuestra universidad funcionando todas bajo un único servidor.
- No poseen disco duro, con la consecuente ventaja que éste no se puede romper al no existir, evitando algunos desastres que todo administrador ha sufrido en algun momento de su carrera, como grandes pérdidas de datos.
- La realización de copias de seguridad es muy sencilla ahora, puesto que basta con guardar en otra máquina o en algún soporte los datos contenidos en el home de los usuarios.
- Los terminales son de fácil reemplazo, ya que cualquier máquina nos serviría para utilizarla de terminal. Podríamos reutilizar pues, máquinas que la universidad se quiera deshacer de ellas.
- Tenemos la posibilidad de utilizar todo el software disponible en GNU/Linux, tanto de oficina, como de diseño, etc. Algunos ejemplos son OpenOffice, Netscape...
- Podemos aprovechar la velocidad actual de las redes de área local, con velocidades de 100 Mbps o incluso de 1 Gbps con las conocidas Gigabit Ethernet, por lo que conseguimos además de velocidad, acabar con los cuellos de botella que se podrían producir en nuestra red.
- Tendremos también un arranque del terminal inmediato, con el consiguiente ahorro de tiempo.
- Optimizaremos al máximo nuestros recursos, ya que no necesitamos disco duro, y la cantidad de memoria RAM es muy baja. Hablaremos pues de un ahorro económico importante.
- Posibilita la gestión, administración y obtención de datos estadísticos del funcionamiento del aula.
- El diseño del aula es totalmente flexible, pudiendo añadir o eliminar máquinas fácilmente.
- También nos encontramos con que son muy fáciles de mantener, y poseen gran escalabilidad.
- Mediante el ahorro en máquinas para las aulas de informática, se consigue, al mismo tiempo, un incremento en la calidad de infraestructuras de los laboratorios, puesto que conseguimos máquinas más potentes invirtiendo en la ampliación o el cambio del servidor por otro con características superiores.
- Los alumnos mantienen las ventajas que obtenían mediante la utilización de otros sistemas operativos también en el uso de los terminales, y además, añaden otras como la robustez y la fiabilidad del sistema operativo GNU/Linux y la posibilidad de trabajar en las prácticas desde casa conectándose remotamente vía Telnet, SSH o X-Window remotas.

2.3. Inconvenientes de un terminal sin disco

El principal inconveniente que se nos plantea a la hora de llevar a cabo el diseño de un aula con terminales bajo Linux, es que necesitamos personal técnico con conocimientos avanzados de administración bajo éste sistema operativo.

Debemos tener en cuenta que todo el software se instalará en un servidor que deberá tener algún tipo de mantenimiento, como lo hubiese necesitado un aula repleta de computadores con Microsoft Windows, pero ahora, en lugar de tener que instalar el software en 40 ó 50 máquinas, sólo será necesario instalarlo en una: *nuestro servidor*.

Otro inconveniente bastante importante es el de la seguridad. Como veremos a lo largo de este articulo, los terminales sin disco utilizan el servicio NFS (Network File System) para la transferencia de archivos por red entre el servidor de terminales y los clientes. Dicho servicio tiene algunas carencias en materia de seguridad, por lo que formaremos una LAN donde ubicaremos todos nuestros terminales y así aislaremos de Internet todos los servicios que puedan ser mas o menos vulnerables frente a posibles ataques desde el exterior. Por tanto, y si nuestro servidor tiene acceso a Internet, es sencillo deducir que poseerá dos interfaces de red: uno para nuestra red local y el otro para salir a la red de redes.

En nuestro diseño del aula, hemos optado por la creación de un firewall por software, aprovechando las características de ruteo que nos proporciona el kernel de GNU/Linux. Así hemos solucionado los posibles fallos de seguridad.

2.4. Especificaciones Técnicas

Hardware en el lado del servidor

El tipo de hardware y la calidad del mismo va a ser un punto importante en este lado, ya que pensemos que vamos a tener unos accesos bastante grandes a nuestro disco duro y un tráfico de red muy alto, incluso con un número elevado de máquinas cliente podríamos formar un cuello de botella en nuestra red local.

Los componentes más importantes, por tanto, serán el disco duro y la memoria RAM.
Es recomendable que el disco duro a instalar en el servidor sea SCSI, por la elevada velocidad que nos proporcionan este tipo de discos, lo que ayudará a que las aplicaciones se ejecuten en el cliente en un tiempo óptimo.

De la memoria RAM también dependerá la velocidad de ejecución de las aplicaciones en los clientes, y es tan importante como el disco duro. Por tanto es recomendable un tamaño elevado de dicha memoria.

También podríamos incluso incorporar una tarjeta Gigabit Ethernet para solucionar consecuentemente los posibles cuellos de botella que se podrían formar en nuestra red local. Este punto no es imprescindible, ni siquiera necesario,

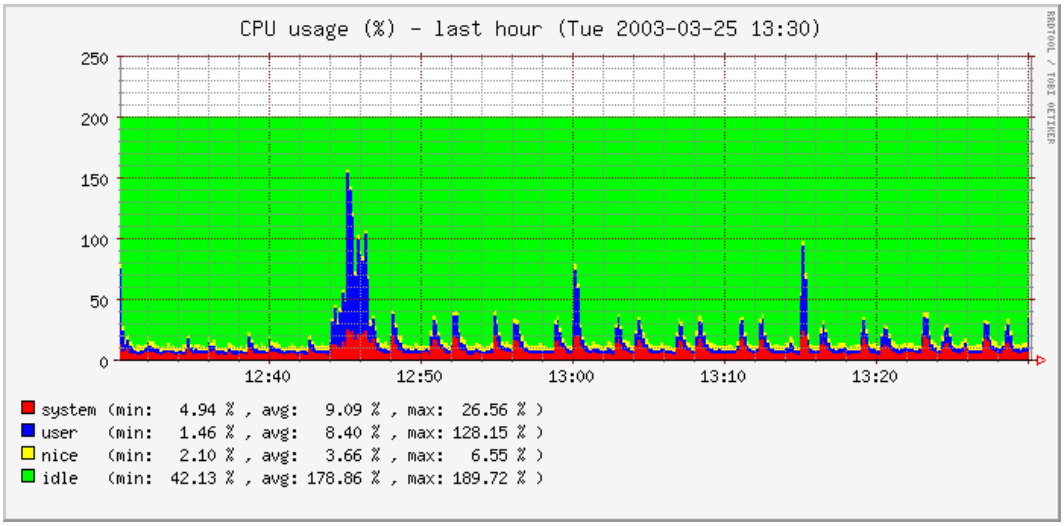


Figura 1. Uso de CPU

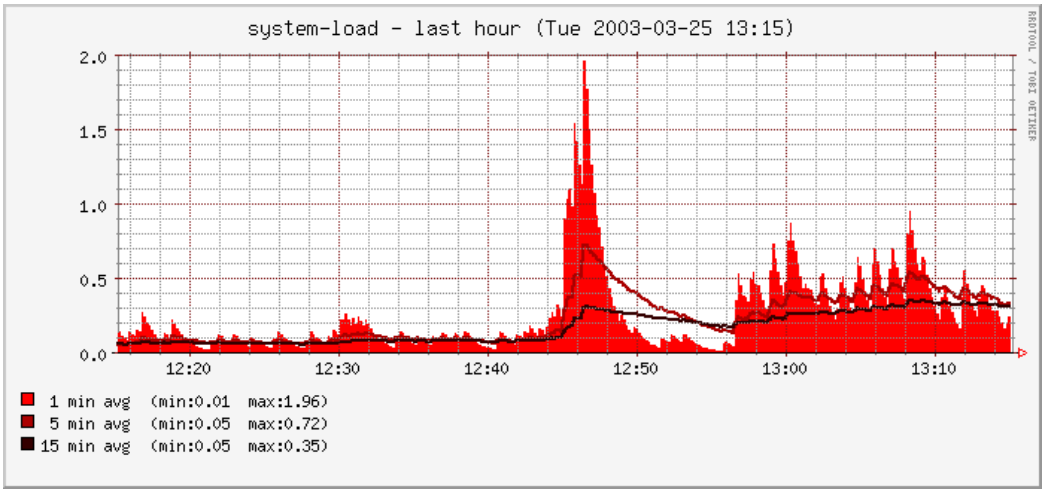


Figura 2. Carga del sistema

pero probablemente con un número muy elevado de clientes –en torno a 120- quizá se pueda producir un retardo en las conexiones debido a un pequeño cuello de botella en nuestra red.

Hardware en el lado del cliente

Respecto a los clientes, lo que se pretende es reutilizar las piezas o los ordenadores completos, así que no hacen falta unos componentes sobredimensionados, o con un coste elevado. Ni que decir tiene, que no es necesario que porten disco duro, ya que esto es la clave de este proyecto junto con la reutilización y ahorro.

Vamos a citar algunos ejemplos de configuraciones de clientes:

- Se han hecho pruebas con equipos i486 con 16MB de RAM, obteniendo resultados muy satisfactorios.
- Se ha trabajado con 100 estaciones de trabajo Pentium 166 con 32MB de RAM, destinadas a hoteles.

Servicios que se utilizan

Este proyecto se basa en una serie de servicios puntuales para su correcto funcionamiento.

Se realizan una serie de pasos desde que se pulsa el botón de encendido de un terminal hasta que obtenemos una pantalla para hacer login. En relación a los servicios, caben destacar que:

- Es necesario hacer una petición DHCP o BOOTP desde un cliente al servidor para obtener una dirección IP a través de la MAC. Esta petición se realizará en BROADCAST. Por tanto, un servidor DHCP será necesario desde el lado de nuestro servidor, ya que este le proporciona a los clientes, además de la IP, la máscara de red local, el path del kernel para bajar y el path del sistema de archivos a montar.
- Nuestros clientes descargan el kernel del sistema operativo a través de TFTP (Trivial File Transfer Protocol) así que será necesario también, dotar a nuestro servidor con servicios de TFTP.
- Para montar nuestro sistema de archivos en la máquina cliente, será necesario utilizar un servidor NFS (Network File System), por tanto será otro de los servicios a tener en cuenta.
- Si nos vamos a basar en terminales X-Window, también será necesario instalar un servidor de X-Window remotas, a través de conexiones XDMCP.

Tipos de terminales

Cabe destacar que se las estaciones de trabajo se pueden configurar de varias formas, atendiendo a nuestras necesidades.

Imaginemos, por ejemplo, que dedicaremos nuestros terminales exclusivamente a la programación en C, y que por lo tanto no será necesario desplegar un sistema X-Window. Tenemos pues una solución sin tener que usar X-Window que consiste en realizar conexiones en modo Telnet a nuestro servidor.

Ahora imaginemos que estamos utilizando los terminales para correr aplicaciones de diseño en CAD. Obviamente necesitaremos una infraestructura X-Window desplegada en nuestro cliente para tal fin.

Estas dos configuraciones serán las que podremos realizar con nuestros clientes, además de una tercera que será de depuración ante posibles fallos en la configuración de nuestro servidor o nuestros clientes.

Rendimiento del servidor

Durante nuestra extensa experiencia con terminales sin disco bajo GNU/Linux, hemos procurado siempre ofrecer garantías educativas, pero también hemos apostado por soluciones fiables, robustas y eficientes a nivel de laboratorios.

Para conseguir este tipo de soluciones, previamente realizamos un exhaustivo estudio de nuestras ideas con el fin de lograr el máximo rendimiento y confianza, antes de ponerlas en marcha adaptándolas a nuestra docencia. Prueba de ello son las Figuras 1,2 y 3 que adjuntamos en este artículo, en las que reflejamos parte de un estudio del servidor a nivel de uso de CPU, de memoria y carga del sistema. En estas pruebas, concretamente, se ha probado el rendimiento del servidor durante el arranque de 20 terminales sin disco y la consiguiente validación de otros tantos usuarios en cada terminal.

En la Figura 1, podemos observar un pico de alrededor del 150% de uso de los dos procesadores a 800 MHz que tiene el servidor. Este pico esta provocado por la validación de los 20 usuarios instantáneamente en el sistema, lo que conlleva un uso muy grande de ambos procesadores, aunque podemos observar que nos resta un 50% libre. Vemos, también, que estos porcentajes se reducen a medida que los usuarios se validan y empiezan a trabajar con el sistema, obteniendo picos de uso de la CPU de alrededor

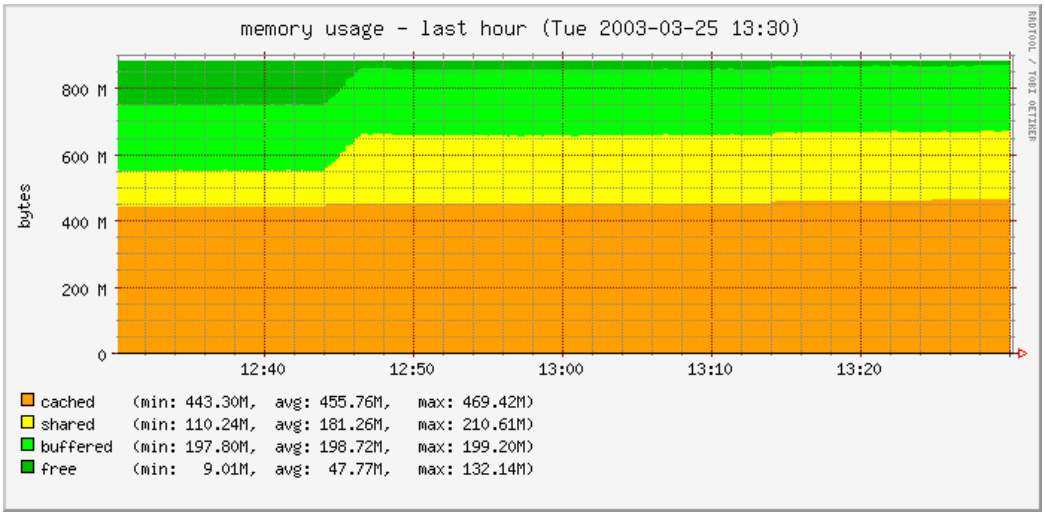


Figura 3. Consumo de memoria RAM

del 40% máximo.

En la Figura 2, podemos percatarnos que la carga del sistema, durante el proceso de arranque + validación de usuarios, ha sido muy pequeña, y que los procesos que ha arrancado –alrededor de 170 procesos con entorno gráfico basado en KDE- apenas tienen peso en nuestro servidor, aun siendo un entorno muy cargado, respecto a otros, en GNU/Linux.

En la Figura 3, podemos observar el uso de memoria RAM, que ha sufrido un incremento en el consumo de solo 100 Mbytes sobre el que ya tenía el servidor. La memoria de SWAP o de intercambio no ha sido utilizada por el sistema, lo que nos da un margen enorme de consumo.

3. Conclusión

Desde nuestra área de Tecnología de Computadores de la Universidad Miguel Hernández, apostamos firmemente por el proyecto que se muestra en este artículo, el cual nos proporciona la versatilidad que deseamos para impartir nuestra docencia práctica con unos resultados satisfactorios y además, que nuestros alumnos la reciban con la mayor calidad posible.

Representa también un desahogo laboral para nuestros técnicos y responsables de aulas, que una vez introducidos en el sistema operativo GNU/Linux, les resultará más sencillo y con un menor gasto de tiempo las configuraciones de las aulas de informática, pudiendo dejar configuradas varias de ellas con un mismo servidor.

Referencias y Bibliografía

[1] GNU/Linux Debian
http://www.debian.org
[2] Oracle
http://www.oracle.com
[3] CNET Network Simulator
http://www.cs.uwa.edu.au/cnet/

[4] SPIN
http://netlib.bell-labs.com/netlib/spin/whatispin.html
[5] Java
http://java.sun.com/
[6] JDK (Java Development Kit)
http://java.sun.com/
[7] Licencias Campus Microsoft
http://www.microsoft.com/education/?ID=Terms
[8] Licencia GPL
http://www.gnu.org/copyleft/gpl.html
[9] Netscape
http://www.netscape.com
[10] GNU
http://www.gnu.org
[11] Linux Terminal Server Project
http://www.ltsp.org
[12] K-12Linux Project
http://www.k12ltsp.org
[13] X-Terminal at City of Largo
http://dot.kde.org/995949998/
[14] Diskless Nodes HOW-TO for Linux
http://www.tldp.org/HOWTO/Diskless-HOWTO.html
[15] Important notes on security issues for X-terminals
http://etherboot.sourceforge.net/doc/html/security.html
[16] Swieskowski, Patrick. Making an X Terminal from a PC. Published in Issue 68 of Linux Gazette, July 2001
[17] Kaszeta, Rich. Linux X Terminals A new use for old and outdated PCs, April 1998 Linux Gazette
[18] Linux NFS-HOWTO:
http://www.tldp.org/HOWTO/NFS-HOWTO/index.html
[19] XDM and X Terminal mini-HOWTO:
http://www.tldp.org/HOWTO/mini/XDM-Xterm/index.html
[20] Debian packages page:
http://www.debian.org/distrib/packages
[21] Girard, Kim. Ellison resurrects network computer. CNET News.com November 16, 1999
http://news.com.com/2100-1001-*233137.html?legacy=cnet*

Multimedia e informática gráfica

El proyecto MEIIGA

Pedro M. Latorre, Francisco Serón
Grupo de Informática Gráfica Avanzada (GIGA)
Dpto. de Informática e Ingeniería de Sistemas
Universidad de Zaragoza
50012 Zaragoza
e-mail: platorre@posta.unizar.es

Resumen

En esta comunicación se presenta el proyecto educativo MEIIGA (*Material para la Enseñanza Interactiva de la Informática GráficA*), que tiene como objetivo desarrollar un conjunto de materiales multimedia para la exposición y el aprendizaje de las asignaturas del bloque optativo Informática Gráfica, que se imparte dentro de los estudios de Ingeniería Informática en el Centro Politécnico Superior de la Universidad de Zaragoza.

Los materiales generados por el proyecto Meiiga parten de la transposición y adaptación del material docente preexistente, e incluyen como principal contribución la inclusión de una numerosa colección de ejemplos sobre los que se puede interactuar en tiempo real. También se ha trabajado especialmente el diseño de la interfaz de usuario, cuidando la adecuada exposición conceptual y visual, la ordenación temática, las relaciones entre los conceptos, la coherencia visual y de uso entre las diferentes partes y sobre todo los ejemplos interactivos. Para la validación de la interfaz se han utilizado procedimientos de evaluación con intervención de los usuarios siguiendo los estándares ISO aplicables.

Este proyecto se ha incluído como ejemplo de buena práctica dentro de los proyectos europeos LEONARDO QUICK y QUICK MULTIPLIER. También se relaciona con la generación de materiales docentes para el proyecto europeo-latinoamericano ALFA T-GAME

1. Presentación

Esta comunicación comienza con una introducción que contiene una serie de reflexiones acerca de las técnicas y modalidades de la *Enseñanza Asistida por Ordenador* en el momento actual, la docencia de las materias del bloque *Informática Gráfica* en el CPS de la Universidad de Zaragoza y describe el proyecto MEIIGA. A continuación se exponen los requisitos de partida y las soluciones adoptadas y se entra en la descripción de la interfaz y de algunos ejemplos interactivos. Se sigue con la evaluación de los estudiantes y se finaliza con la exposición de conclusiones y referencias.

1.1. Introducción

La aparición de las técnicas de *Enseñanza Asistida por Ordenador* ha desencadenado una catarata de experiencias docentes. Si se clasifican los materiales generados según sus objetivos, se podría hablar de tres grupos: los dirigidos a la actividad en el aula, a la ayuda del trabajo individual del alumno y a la enseñanza a distancia (EAD). Estos grupos, que en la actualidad se solapan gracias a la generalización del uso de Internet como medio de comunicación, abarcan desde grandes iniciativas, como los campus virtuales donde se han empezado a impartir carreras completas, hasta experiencias más modestas, como la transposición de materiales docentes tradicionales para su exposición mejorada en el aula.

Recordemos el siguiente dicho: *"Si una persona de la Grecia clásica levantara la cabeza, la única actividad que reconocería sería la del aula"*. Esta frase, que se cita con cierta frecuencia para justificar todo tipo de intervenciones más o menos novedosas en la actividad educativa, va quedando obsoleta. Sin embargo, encierra un hecho fundamental: la adquisición de conocimientos dentro del contexto social tradicional en el aula mejora en gran medida el

rendimiento del estudiante. Por esta razón los experimentos de educación de individuos en aislamiento han tenido un éxito limitado en general, y los ensayos tienden actualmente a hablar de educación síncrona (o presencial) y asíncrona, que abarca la semipresencial (70% de tiempo presencial y 30% de estudio a distancia) y la enseñanza totalmente a distancia, asistida parcialmente mediante aplicaciones en red. Esta última modalidad tiende a reservarse a aquellas circunstancias en las que resulta imposible la docencia mediante las dos primeras. Estas líneas (semipresencial y a distancia) son las que exploran el *Anillo Digital Docente de la Universidad de Zaragoza* y el *Grupo de Universidades G9*.

1.2. La Informática Gráfica en la Universidad de Zaragoza

El Centro Politécnico Superior de la Universidad de Zaragoza imparte en la actualidad las materias correspondientes a las titulaciones de Ingeniería Industrial, Telecomunicaciones, Química e Informática. El plan de estudios correspondiente a esta última titulación ofrece el bloque optativo de Informática Gráfica, que se estructura en tres asignaturas: Informática Gráfica, Modelado Geométrico y Modelado Visual y Animación. La carga total de este bloque es de 18 créditos (180 horas) de clases teóricas y prácticas.

La primera asignatura, Informática Gráfica, ofrece los conceptos generales del bloque, tratando la evolución histórica, el hardware gráfico, la visualización 2D y 3D, el proceso de rendering de objetos poligonales y los aceleradores gráficos. La segunda, Modelado Geométrico, estudia la descripción de la forma de los objetos, exponiendo su evolución histórica, los conceptos básicos de Geometría Analítica y Geometría Diferencial, la interpolación y aproximación de funciones 2D y 3D, el modelado geométrico de objetos fractales determinísticos y aleatorios y las técnicas de implementación directa y procedural. La tercera, Modelado Visual y Animación, trata del color y la textura de los objetos y de la generación de secuencias animadas, y estudia los modelos del color, de los modelos heurísticos y fenomenológicos del modelado visual y de la animación tradicional (2D) y tridimensional, cinemática y dinámica [2] [4] [9].

La metodología de enseñanza se basa en el *aprendizaje a través de problemas* [1] [3]. La resolución de problemas es un proceso que requiere el conocimiento de una disciplina y las técnicas y habilidades necesarias para salvar el espacio existente entre el problema y la solución. En este sentido, se concibe la resolución de problemas como un proceso *productivo*, donde el sujeto requiere un periodo de *incubación* seguido de una repentina *intuición* mediante la cual reorganiza mentalmente la estructura del problema.

1.3. El proyecto MEIIGA

Dentro de este marco docente, el objetivo del proyecto MEIIGA es el diseño y desarrollo de un material software que permita la enseñanza y aprendizaje interactivo de los conceptos que se imparten en las tres asignaturas citadas mediante la utilización de diferentes elementos multimedia: texto, imágenes, animaciones y ejemplos interactivos [8][10]. El prototipo inicial desarrolla una presentación para su utilización directa por el profesor durante las clases y por los estudiantes en su trabajo personal, y puede ser consultada directamente desde cedé o mediante conexión a red. Posteriormente se desarrollarán materiales adicionales para uso del alumno, como descripciones textuales detalladas, ejercicios, tests de autoevaluación, etc.

El contenido textual se basa en el material desarrollado con anterioridad y verificado a lo largo de varios años de docencia, convenientemente actualizado.

Las imágenes estáticas, vídeos digitalizados y animaciones sirven para reforzar la comprensión de los conceptos explicados de forma textual, así cómo evitar la monotonía que provoca la exposición exclusiva de texto.

Los ejemplos interactivos constituyen la principal innovación de MEIIGA, y permiten visualizar en tiempo muy corto tanto los conceptos que se están explicando como el efecto del cambio de cualquiera de los parámetros que intervienen, la visualización desde diferentes puntos de vista, etc.

Finalmente, un Motor de Búsqueda permite realizar la búsqueda por palabras clave de determinados temas o conceptos. De esta manera

se ofrece un acceso rápido y sencillo al concepto que se desea presentar o estudiar.

No es frecuente la aparición de experiencias semejantes en la bibliografía. ACM-SIGGRAPH tiene un depósito de ejemplos aislados [11], y D. Nikolic [12] habla de GraphicMentor, una herramienta para el aprendizaje de los conceptos básicos sin ligadura con las correspondientes lecciones.

2. Requisitos de partida y soluciones adoptadas

Los requisitos o requerimientos del proyecto se pueden estructurar en varios apartados que incluyen la plataforma, metodología, velocidad de ejecución, interfaz y modificación de la información. A continuación se exponen los requisitos de cada uno de los apartados y las decisiones elegidas para cada uno de ellos.

2.1. Metodologías de Desarrollo

Para realizar las fases de diseño y desarrollo de las aplicaciones interactivas se ha seguido la Metodología Orientada a Objetos según el modelo Booch. Esta metodología permite el paso eficiente de la fase de diseño a la de implementación, realizada con el lenguaje de programación orientado a objetos Java.

Para el desarrollo de la interfaz de usuario se ha seguido una metodología basada en la generación de prototipos, que incluye cuatro fases bien diferenciadas: análisis, diseño, implementación y evaluación. Se han utilizado los estándares ISO 9241 (*Ergonomic requirement for office work with visual display terminals*) [5], e ISO 13407 (*Human-centred design processes for interactive systems*) [6] y diversas normas de estilo, preferentemente las de la Universidad de Yale [7].

Cabe resaltar la metodología de desarrollo de la interfaz con intervención del usuario. A partir de un primer prototipo, se desarrollaron una serie de entrevistas con los profesores de las asignaturas. De las conclusiones obtenidas salió un segundo prototipo, con el que se impartieron algunas clases a los estudiantes, a los que posteriormente se pidió que rellenaran un cuestionario elaborado conforme a la norma ISO antes citada. El análisis de los resultados condujo a una serie de modificaciones que levaron a la versión final de la interfaz.

2.2. Plataforma

La aplicación se ha diseñado teniendo en cuenta su posible uso local y en red, en cualquier plataforma. Para satisfacer este requisito, el modelo elegido ha sido el de Cliente–Servidor en Internet; los contenidos se visualizan utilizando cualquier navegador (en versiones actuales) y se accede a ellos bien vía Internet o bien en modo local.

El contenido se ha desarrollado en HTML 4.0, y las aplicaciones interactivas en Java Standard. Esta decisión permite la operación desde plataformas mínimas, no necesariamente actualizadas, y también la actualización fácil de contenidos.

2.3. Velocidad de Ejecución

Aunque no se exige el funcionamiento de la aplicación en tiempo real, los componentes utilizados deben ser rápidos en su ejecución y visualización. Tiempos muertos superiores a pocos segundos suponen un corte en la atención del estudiante, que resulta muy difícil recuperar.

Este aspecto tiene vital importancia a la hora de ejecutar e interaccionar con los ejemplos interactivos, sobre todo pensando en el tiempo necesario para la carga de las aplicaciones en red frente a la carga en modo local. Por ello, siempre es preferible esta última para las presentaciones.

Así pues, y debido principalmente a las limitaciones en la velocidad de transmisión por Internet, la única forma de garantizar una adecuada y permanente interactividad durante la clase es disponer del material en un disco local o un CD, y ejecutar las aplicaciones localmente.

3. La interfaz de usuario

Como ya se ha dicho anteriormente, el desarrollo de la interfaz de usuario se ha basado en una metodología basada en la generación de prototipos, que incluye cuatro fases bien diferenciadas: análisis, diseño, implementación y evaluación. Se han utilizado los estándares ISO

9241 [5] e ISO 13407 [6] y diversas normas de estilo, preferentemente las de la Universidad de Yale [7]. Se describe aquí el modelo conceptual, la estructura y los elementos de la interfaz.

3.1. Modelo Conceptual

Los elementos de cada página son los siguientes:

- *La información Asignatura-Parte-Tema-Ítem*, que permite la localización de cada página dentro del temario.
- *El contenido* de cada página.
- *La barra de navegación*.
- *Los ejemplos interactivos*, en su caso.

La estructura de una página se muestra en la figura 1.

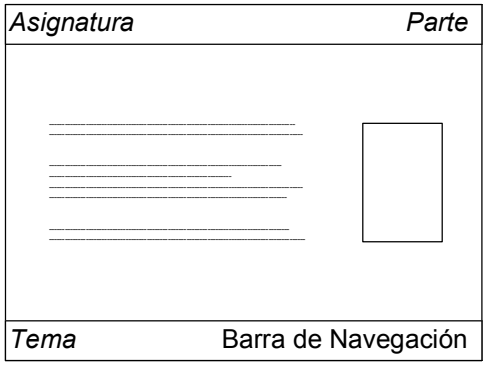


Figura 1. Estructura de una página

El modelo de navegación a través de la información sigue la jerarquía descendente: Asignatura, Parte, Tema e Ítem o lección.

Existen cuatro tipos de ventana: la ventana-índice, donde aparecen las entradas correspondientes al nivel y se despliegan las entradas del nivel siguiente, permitiendo la elección interactiva del que sea de interés; la ventana-exposición, donde aparece el texto e imágenes correspondientes a un determinado ítem; la ventana-ejemplo, que contiene la aplicación interactiva y puede estar encastrado en una página-exposición o desplegarse aparte, y finalmente la ventana de ayuda en línea, que o bien aparece sobre una ventana completa o bien posee estructura interna, accediéndose a cada entrada mediante su grupo de botones.

El mapa de navegación se esquematiza en la figura 2.

3.2. Barra de navegación

Para la barra de navegación se usan imágenes con apariencia de botón, que pueden estar activados o desactivados según el caso y que permiten acceder al índice general de la asignatura, a la página principal del proyecto MEIIGA, a las páginas de ayuda, al nivel superior y a la página anterior o posterior dentro del mismo nivel. También existen botones de acceso al ejemplo interactivo pertinente y al motor de búsqueda.

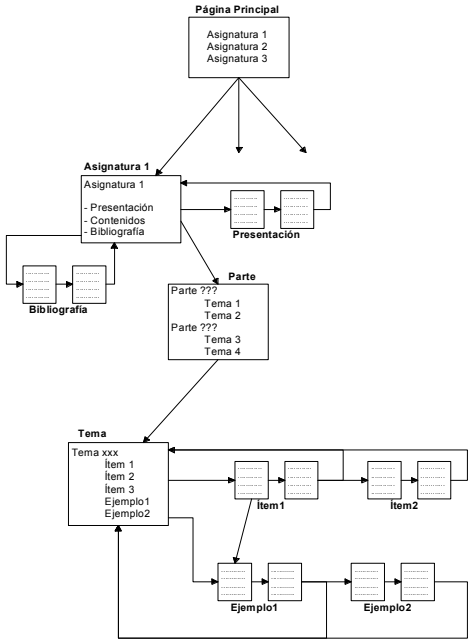


Figura 2. Mapa de navegación

3.3. Aplicaciones interactivas

Los ejemplos interactivos se pueden integrar en la página de dos formas: bien insertándolos dentro de la propia página como si de una imagen interactiva se tratase, bien ejecutándose en ventanas aparte mediante la creación de una nueva ventana independiente del explorador utilizado, definida en Java como un *frame*. En este caso, el formato engloba todo el ejemplo en un solo bloque, con su título y grupo de botones.

La interacción se realiza mediante botones o iconos de tamaño fijo y en blanco y negro. Los botones pueden ser de varios tipos:

- *De acción inmediata*: Al pulsar sobre ellos se produce una acción inmediata, como por ejemplo realizar una ampliación o reducción
- *De estado*: Al pulsar sobre ellos se activan o desactivan mostrando el botón en imagen inversa en caso de estar activado o normal en caso de estar desactivado. Un ejemplo es el botón de visualización de la malla de control en modelado de superficies.
- *De estado o selección múltiple:* Al pulsar sobre ellos las imágenes cambian de manera cíclica mostrando diferentes posibilidades de interacción; por ejemplo, el botón de selección de tipo de superficie cuádrica
- *En exclusión mutua*: Lo forman un conjunto de botones colindantes y relacionados, de los cuales sólo uno puede estar activado (en imagen inversa) al mismo tiempo; por ejemplo, el conjunto de botones de selección de plano de desplazamiento en *Superficies bilineales* y *Superficies de aproximación.*

Todo ejemplo interactivo emergente dispone de una barra de menú, con al menos dos opciones de menú, *Archivo* y *Ayuda*,

La coherencia visual se ha logrado utilizando códigos de color asociados a un significado fijo. Por ejemplo, en las aplicaciones interactivas aparecen en rojo los valores de los parámetros que pueden ser modificables por el usuario –provocando la correspondiente modificación de la escena-, y en negro cuando son fijos.

4. Las aplicaciones interactivas

La contribución didáctica más novedosa de MEIIGA son sus ejemplos interactivos, que no pueden ser descritos fielmente mediante texto e imágenes estáticas. En este artículo se enumeran las aplicaciones desarrolladas y se describen un par de ellas como ejemplo.

4.1. Aplicaciones desarrolladas

Se ha desarrollado un juego de ejemplos interactivos para Informática Gráfica y Modelado Geométrico. Está en desarrollo el conjunto correspondiente a Modelado Visual.

Las aplicaciones disponibles son las siguientes:

- *Informática Gráfica*:

Trazado de Líneas, Transformaciones Geométricas 2D y 3D, Sistemas de Visualización, Proyecciones Geométricas, Recorte de Líneas y de Polígonos, Rendering. Básico.

- *Modelado Geométrico*:

Cicloides Naturales, Generación de Curvas y Superficies por Aproximación, Superficies Bilineales y de barrido, Superficies Regladas, Superficies de Barrido y Regladas (comparación), Superficies de Revolución, Superficies Cuádricas, Fractales de Julia y Mandelbrot.

4.2. Ejemplo: Generación de curvas 2D por aproximación

Se han desarrollado nueve aplicaciones interactivas que permiten generar (en 2D) curvas Bezier y Bspline y BSpline racional utilizando vectores de nudos abiertos, abiertos no uniformes y uniformes, permitiendo observar las diferencias aparentes entre los métodos y la evolución de la curva BSpline hacia la de Bezier al ir aumentando el orden de la primera.

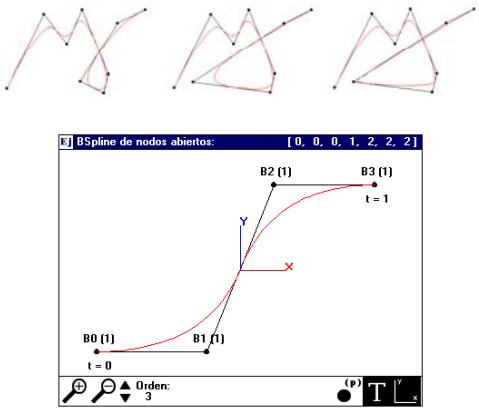


Figura 3. Curvas 2D. Arriba: efecto logrado al desplazar un punto de la malla de control y al variar el orden de la curva. Abajo: Ventana de visualización con sus elementos de control.

Todos los ejemplos se basan en la capacidad de mover de modo interactivo los puntos que

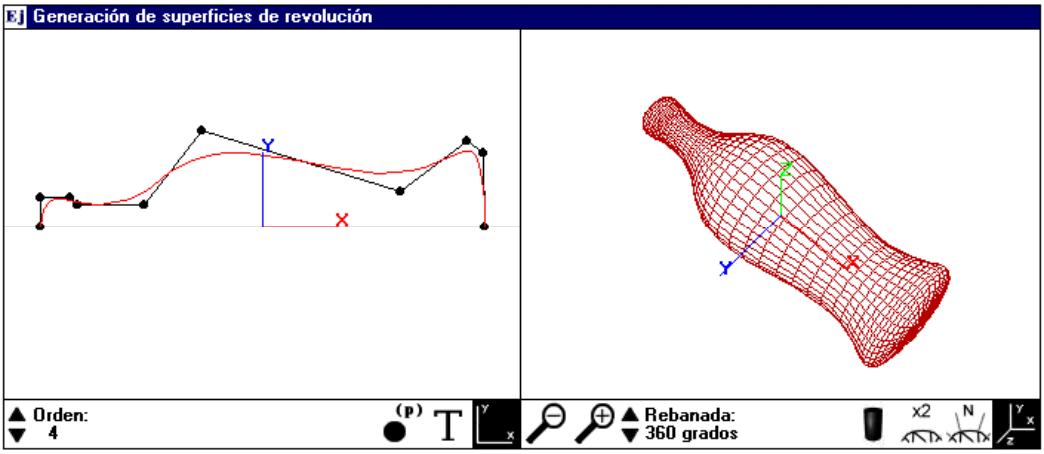


Figura 4. Superficies de revolución

constituyen el polígono de control y de modificar sus pesos (en su caso), con el objetivo de observar cómo afectan estas variaciones a la curva generada.

Para mover un punto sólo es necesario pulsar en él con el botón izquierdo del ratón y sin soltarlo arrastrar el mismo hasta su nueva posición. En la parte inferior de las aplicaciones se muestra una serie de botones, que varían en función del tipo de curva y permiten modificar los parámetros de control y el factor de escala (ver figura 3).

4.3. Ejemplo: Superficies de revolución

El objetivo de esta aplicación es mostrar el comportamiento de las superficies de revolución al variar la forma de la curva generadora o su posición con respecto al eje de rotación.

En la parte izquierda de la ventana se visualiza la curva generatriz, que puede ser modificada estirando de los vértices del polígono de control o modificando éste.

En la parte derecha se visualiza la superficie obtenida en malla de alambre o sombreado de Gouraud.. Su aspecto puede ser observado desde cualquier posición sin más que deslizar el ratón del modo adecuado También se pueden visualizar las normales y variar la escala (ver figura 4).

5. La opinión de los estudiantes

Para disponer de una referencia acerca del grado de satisfacción de los estudiantes se impartió una lección completa mediante los dos sistemas: tradicional (con transparencias) y utilizando MEIIGA. Al finalizar el experimento se pasó una encuesta pidiendo la opinion sobre tres aspectos: la calidad de la interfaz, la calidad del material del módulo impartido (Transformaciones Geométricas) y el grado de aprovechamiento de la aplicación. Las respuestas para cada cuestión incluyeron una calificación (0...10), una estimación (-3...+3) del grado de mejora con respecto a la exposición tradicional y un comentario textual. Los resultados pueden verse en la tabla 1.

En conjunto, los estudiantes mostraron su satisfacción con el uso del nuevo material, en comparación con el método tradicional. Como era de prever, los ejemplos interactivos recibieron las calificaciones más satisfactorias, tanto globalmente como en grado de mejora. Por el contrario, se critica la claridad y visibilidad del texto, debido a la calidad de la proyección, que es mejorable.

La evaluación de la efectividad de MEIIGA en los resultados académicos sería deseble, pero no se ha podido realizar ya que el método de evaluación no lo permite. La evaluación se realiza valorando la implementación de un visualizador,

que se desarrolla en versión básica (objetos 3D poligonales) para Informática Gráfica. A esta aplicación se le añade un módulo para la obtención de las mallas que representan curvas y superficies (Modelado Geométrico) y texturas y modelos de iluminación (Modelado Visual).

6. Conclusiones

Las nuevas formas de presentación electrónica permiten, si se utilizan con rigurosos planteamientos y objetivos, una mejora en la calidad docente, ya que permiten

- incrementar la atención del estudiante si las aplicaciones de apoyo se plantean con la adecuada calidad y claridad en las presentaciones
- aumentar las posibilidades de percepción de las relaciones de cada concepto con el resto, mediante la exhibición continua de los parámetros del contexto y mediante el uso de las posibilidades de navegación entre partes.
- facilitar la transmisión de algunos conocimientos mediante la adecuada presentación de ejemplos interactivos cuidadosamente seleccionados.

Proyecto MEIIGA – Satisfacción del usuario (44 Respuestas)		Media	D.E.
1. - Interface			
Presentación en general	Calificación (0...10)	7,48	1,07
	Grado de mejora (-3…+3)	1,57	0,82
Navegación (facilidad de comprensión)	Calificación (0...10)	8,32	0,96
	Grado de mejora (-3…+3)	1,82	0,79
Tipografía (claridad, visibilidad)	Calificación (0...10)	5,32	2,03
	Grado de mejora (-3…+3)	-0,66	1,61
Ejemplos interactivos (usabilidad)	Calificación (0...10)	8,52	1,28
	Grado de mejora (-3…+3)	2,41	0,82
2. Módulo: Transformaciones Geométricas			
Presentación textual (explicaciones)	Calificación (0...10)	7,26	1,31
	Grado de mejora (-3…+3)	0,68	1,23
Ejemplos interactivos	Calificación (0...10)	8,88	0,82
	Grado de mejora (-3…+3)	2,50	0,66
3. Aprovechamiento de la aplicación			
Como aplicación docente, en clase	Calificación (0...10)	7,48	1,62
	Grado de mejora (-3…+3)	1,50	1,07
Como elemento de estudio, en casa	Calificación (0...10)	8,42	1,38
	Grado de mejora (-3…+3)	2,28	0,93

Tabla 1. La opinión de los usuarios

Este último punto resulta especialmente importante en nuestro dominio de enseñanza, donde una de las dificultades más importantes aparece a la hora de explicar conceptos tridimensionales que son mucho mejor entendidos si se visualizan al instante los cambios que aparecen al cambiar algún parámetro, o si se observa una escena desde diferentes puntos de vista, por ejemplo.

En lo que se refiere al desarrollo de la interfaz, los métodos de diseño y evaluación con intervención del usuario resultan muy valiosos

para asegurar la validez y adecuación de las presentaciones docentes. El uso de estándares ISO es muy útil como guía de diseño y seguimiento de los procesos de desarrollo, asegurando la validación; no obstante, exige un trabajo previo de adopción de metodologías concretas, que puede resultar muy costoso.

Las limitaciones de la velocidad en la transmisión por Internet hacen imprescindible que se disponga de la totalidad del material a exponer en el disco local o en CD cuando se va a impartir una clase. Los tiempos necesarios son tolerables para un entorno individual de trabajo.

El conjunto de materiales ofrecidos al estudiante complementan los tradicionales y mejoran su comprensión, por los motivos expuestos con anterioridad.

Referencias

[1] Chevallard, Y., Boch, M. y Gascón, J. *Estudiar Matemáticas, un eslabón perdido entre enseñanza y aprendizaje*. Cuadernos de Educación. ICE, Universidad de Barcelona. Horsori Editorial, 1997.

[2] Foley, vanDam, Feiner, Hughes, *Computer Graphics, principles and practice*, 2nd edition. Addison-Wesley, noviembre 1996.

[3] Gascón, J. *El papel de la resolución de problemas en la enseñanza de las Matemáticas*. Departamento de Matemáticas de la UAB, 1992.

[4] Hearn, D., Baker, P. *Gráficas por computadora*, segunda edición. Prentice Hall, 1995

[5] ISO 9241 *Ergonomic requirements for office work with visual display terminals (VDTs),* partes 1 y 10 a 17. International Organization for Standardization, 1997-99.

[6] ISO 13407 *Human-centred design processes for interactive systems.* International Organization for Standardization, 1997-99.

[7] Lynch, Horton, *Yale Style Web Guide*. Yale University, 1999. También en http://info.med.yale.edu/caim/manual,

[8] Marqués, Pere. Metodología para la elaboración de software educativo. Comunicación y pedagogía, 1997.

[9] Rogers, Adams, *Mathematical Elements for Computer Graphics*, 2nd edition. MacGraw Hill, 1990.

[10] Roldán, F.J., Ordóñez, J.J. *Material para la Enseñanza Interactiva de la Informática Gráfica Avanzada*. Proyecto Fin de Carrera.. Departamento de Informática e Ingeniería de Sistemas. Universidad de Zaragoza, 1999

[11]Materiales para la enseñanza de la Informática gráfica de la ACM-SIGGRAPH en http://www.siggraph.org/education/materials/C_and_I.htm

[12] Nikolic D. Y Ching-Kuang Shene. *GraphicsMentor: A tool for Learning Graphics Fundamentals*. SIGCSE Bulletin - Inroads. SIGCSE'02 Proceedings, vol. 34 no. 1 pp. 242-246, 2002

Organización curricular y planes de estudio

Reflexiones y experiencias sobre la enseñanza de POO como único paradigma

Daniel Gayo Avello, Agustín Cernuda del Río, Juan Manuel Cueva Lovelle,
Marián Díaz Fondón, Mª Pilar Almudena García Fuente, José Manuel Redondo López
Departamento de Informática
Universidad de Oviedo
C/Calvo Sotelo s/n 33007 Oviedo
e-mail: {dani, guti, cueva, fondon, almu, redondo}@lsi.uniovi.es

Resumen

En la actualidad el paradigma de orientación a objetos (en adelante OO) está ampliamente aceptado y todos los Planes de Estudios modernos consideran necesario impartir una o más asignaturas relacionadas con el análisis, el diseño y la programación orientados a objetos en las titulaciones universitarias en informática.

Sin embargo, existe un candente debate sobre el momento de presentar a los alumnos la programación orientada a objetos (en adelante POO). Algunos docentes consideran problemático empezar directamente con POO y creen necesario enseñar previamente programación estructurada. Otros profesores opinan de manera totalmente opuesta.

En este artículo expondremos nuestros argumentos a favor de una enseñanza temprana de la POO, así como los contenidos teóricos y prácticos impartidos en una asignatura de Introducción a la Programación (OO) de primer curso, además de los resultados y las conclusiones alcanzadas en este primer año de andadura.

1. Introducción

Desde hace algún tiempo se viene debatiendo la forma de introducir la POO en las enseñanzas universitarias en informática. Se trata de un debate fuertemente polarizado. En un lado nos encontramos aquellos que consideramos que es necesario comenzar a explicar a los alumnos OO desde el primer día. En el otro se sitúan los que creen que se debe empezar por la programación estructurada para, después, enseñar OO.

Con este artículo no pretendemos echar más leña al fuego; somos conocedores de los inconvenientes de ambas opciones, pero el hecho es que forzosamente hay que decantarse por una y asumir dichos inconvenientes. Aquí pretendemos simplemente describir por qué y cómo estamos enseñando a alumnos de primer curso de Ingeniería Técnica en Informática POO desde el primer día sin encontrar más dificultades de las que se encontraban al enseñar programación estructurada y con unos resultados provisionales que juzgamos bastante satisfactorios.

En primer lugar partiremos de la hipótesis de que el paradigma OO, sin ser la panacea, es la mejor opción disponible en este momento para el desarrollo de software. Sobre esa base, argumentaremos que si la OO es un objetivo deseable es conveniente afrontarlo desde el primer momento y de la forma más directa posible. A continuación definiremos lo que entendemos por "introducción a la programación" y la forma en que semejante definición encaja en el programa de una asignatura de primer cuatrimestre. Posteriormente, describiremos la manera en que se conducen las clases, se expondrán algunos ejercicios llevados a cabo con los alumnos y se comentarán brevemente los resultados parciales que se han obtenido, así como las impresiones de alumnos que han experimentado en cursos anteriores la enseñanza de programación estructurada como paso previo a la POO.

2. ¿Es útil la orientación a objetos?

Muchos conceptos básicos de la OO se remontan a finales de los años 60 (lenguaje Simula) y 70 (Smalltalk) y es a mediados de los años 80 cuando la OO comienza a despuntar. Hoy en día la OO es

considerada por la inmensa mayoría de las personas involucradas en el desarrollo de software como el mejor paradigma disponible para construir sistemas informáticos.

Sin embargo, el hecho de que un grupo numeroso afirme que algo es cierto no implica que necesariamente lo sea. En el OOPSLA 2002 se celebró un debate en torno al fracaso de los objetos. Apoyaba esta tesis R.P. Gabriel y la refutaba G.L. Steele Jr., ambos de *Sun Microsystems*.

Entrar en detalles sobre tan interesante discusión no es el objeto de este escrito pero puede resultar ilustrativo mencionar algunos puntos allí tratados.

Gabriel argumenta el fracaso de los objetos apoyándose en la incapacidad de la OO para modelar de manera adecuada futuros requisitos (funcionamiento distribuido, ubicuidad y cooperación entre componentes), en la pérdida de la simplicidad de los lenguajes originales, en el escaso índice de reutilización así como en la falsa sensación de facilidad de aprendizaje cuando "en realidad (programar) es tan difícil como siempre" [7]. Sin embargo, la principal crítica que hace a la OO es que "asfixia" la investigación y desarrollo de nuevos paradigmas de programación.

Por su parte, Steele [13] afirma que la OO es un éxito: la mayor parte de los desarrolladores emplean lenguajes OO y los objetos son un buen modelo para la mayoría de entidades del mundo real. Señala que el énfasis en procesar entradas de datos para transformarlas en salidas es una debilidad de la programación procedimental que ha hecho que la OO fuese la respuesta a muchos problemas que la primera difícilmente podía resolver. Naturalmente, acepta que la OO no puede resolver todos los problemas de la programación, que la evolución del paradigma no se ha completado (y tal vez nunca se haga) y que los lenguajes actuales no son los mejores lenguajes OO posibles.

Por nuestra parte, estamos de acuerdo en que es necesario desarrollar más el paradigma OO y otros nuevos; sin embargo, estamos con Steele cuando afirma que "la POO es como el dinero: no lo es todo, pero es mucho mejor que las otras opciones". Por otro lado, también coincidimos con Gabriel en que la POO no es más sencilla, sólo igual de difícil que la programación procedimental.

3. Orientación a objetos en los planes de estudio

En resumen, la OO y los lenguajes, métodos y herramientas asociados no son una solución óptima, pero es lo mejor que hay disponible y el mercado demanda profesionales formados en estas técnicas. Esta necesidad ha afectado finalmente a la forma en que se plantean los planes de estudio de las titulaciones en informática.

El *Computing Curricula 2001* de *ACM* e *IEEE Computer Society* [1] señala la OO como un cambio tecnológico relevante en la enseñanza universitaria de la informática. Considera que la programación, el análisis y el diseño OO deben formar parte de cualquier plan de estudios en informática y plantea varios enfoques para su enseñanza, interesándonos especialmente los denominados "Primero imperativo" y "Primero objetos". En el siguiente apartado argumentaremos por qué consideramos conveniente aplicar el segundo planteamiento.

4. ¿Por qué empezar por objetos?

Hemos establecido que la OO es un paradigma extendido, con un amplio soporte por parte de la industria y que debe ser conocido por cualquier estudiante universitario de informática. Así pues, la cuestión no es si se debe enseñar OO sino en qué momento se debe presentar; básicamente, se trata de elegir entre las dos opciones planteadas en el mencionado informe de *ACM/IEEE*.

Este el centro del debate que mencionábamos en la introducción: "¿Se debe enseñar desde el primer día POO?" Sin embargo, creemos que la pregunta que se formula en semejante debate debería replantearse como: "¿Se debe enseñar desde el primer día POO sabiendo que la OO es un objetivo fundamental de una titulación universitaria en informática?"

Entendemos que la primera pregunta, al estar planteada incorrectamente, ofrece muchas posibilidades de polémica. Sin embargo, si aceptamos que la OO es un paradigma útil y que los alumnos deben comprenderlo y manejarlo con soltura es necesario, en nuestra opinión, comenzar desde el primer momento con la POO (en especial, en titulaciones de ciclo corto como la impartida en nuestra Universidad).

4.1. Posturas opuestas a un enfoque temprano

Muchos autores consideran un error iniciar la enseñanza de la programación comenzando con la OO. Tan sólo en JENUI 2002 se presentaron cuatro ponencias que desaconsejaban tal opción.

Así, Fernández Muñoz et al ([5] y [6]) señalan los siguientes problemas: sobrecarga de conceptos teóricos o sobresimplificación, utilización de bibliotecas de clases, utilización de interfaces gráficos o entornos de desarrollo artificiosos. García Molina et al [9] proponen retrasar las asignaturas de POO hasta el tercer curso de la titulación. Por su parte, Gómez Albarrán [10] propone una asignatura de introducción a la programación que comienza con la descomposición funcional para pasar, posteriormente, a la OO. Todas estas propuestas pueden encuadrarse en la línea de "objetos más tarde".

4.2. ¿Qué paradigma enseñar si la OO es el objetivo?

Las propuestas anteriores plantean algo en apariencia muy sencillo: primero se enseña a los alumnos el paradigma procedimental así como TAD's. Una vez han asimilado los conceptos deberían aprender fácilmente POO puesto que "puede considerarse un paradigma construido sobre la base del paradigma imperativo" [8].

Sin embargo, en la práctica es frecuente que los alumnos se encuentren con el problema del "cambio de paradigma". Stroustrup [14] asegura que un programador experimentado en el paradigma procedimental necesita entre 6 y 18 meses para adaptarse al paradigma OO. Probablemente sea pedir demasiado que nuestros alumnos se adapten mejor que un profesional.

Hay otras razones de peso que aconsejan comenzar por la OO si es el objetivo perseguido en la formación de los estudiantes. Todas ellas se derivan de una premisa muy simple: a pesar de las similitudes no puede decirse que la OO sea una mera extensión del paradigma procedimental.

Bergin [2] señala que el problema de enseñar el paradigma procedimental como paso previo a la POO radica en que las estrategias de resolución de problemas de ambos paradigmas son radicalmente distintas. Así, la programación procedimental fomenta la división de problemas en subproblemas que deben ser resolubles a partir de una serie de datos que serán procesados y transformados, a su vez, en nuevos datos. Así pues, en la programación procedimental el algoritmo es lo que cuenta y los datos son algo secundario, circunstancia poco frecuente en el mundo real.

En parte, esta debilidad de la programación procedimental ha facilitado el camino de la POO. Ésta no busca en primer lugar soluciones a los problemas sino que trata de modelar el sistema como objetos que tendrán un comportamiento, un estado y unas relaciones de colaboración entre los mismos.

Por ello, aun cuando los lenguajes OO sean imperativos y se puedan establecer ciertos símiles entre métodos y subrutinas o entre objetos y TAD, lo cierto es que la forma en que se desarrolla un sistema informático para resolver un problema del mundo real es muy diferente según el paradigma que se emplee; de ahí el problema del cambio de paradigma y la afirmación tajante de Bergin de que el paradigma procedimental es la opción equivocada si lo que se quiere es enseñar OO. No podemos por menos que estar de acuerdo con su postura.

5. Una propuesta para la introducción a la programación

Un aspecto interesante de todas las propuestas mencionadas en el apartado 4.1 es que el citado problema del "cambio de paradigma" no parece producirse. Sería muy interesante estudiar qué aspectos de la actividad docente de los respectivos autores permiten evitar tal problema, pero en nuestro caso lo hemos sufrido de forma clara y reiterada, hasta el punto de que lo consideramos una premisa en las decisiones que tomamos.

Debido en parte a nuestra experiencia y convencidos de la importancia de la OO para los alumnos se decidió aprovechar la entrada del nuevo Plan de Estudios[1] para construir una asignatura de introducción a la programación donde el único paradigma que estudiaría el alumno fuese la OO.

Desde nuestro punto de vista el objetivo primordial de tal asignatura sería proporcionar a los alumnos conocimientos y estrategias básicos para poder enfrentarse a un problema del mundo real susceptible de ser solucionado informáticamente,

[1] http://www.euitio.uniovi.es/nuevoplan/spanish

determinar los objetos que podrían modelar tal sistema, describir con una notación precisa el modelo a construir e implementar dicho modelo mediante un lenguaje de programación OO.

En el caso que nos ocupa, dicha asignatura, denominada Introducción a la Programación, consta de 3 créditos de teoría, 1 crédito de prácticas de tablero y 2 créditos de prácticas de laboratorio y se imparte en el primer cuatrimestre.

A la hora de elaborar el programa y seleccionar las herramientas a emplear se trataron de alcanzar los siguientes objetivos:

1. Seguir, hasta donde fuera posible, las propuestas "objetos primero" de *ACM/IEEE* [1].
2. Lograr que el alumno aprendiese POO y no las peculiaridades del lenguaje de programación elegido.
3. Presentar los contenidos teóricos de manera que se facilitase su comprensión y aprendizaje progresivo y, a la vez, permitiese el desarrollo de ejercicios prácticos desde el primer momento.
4. Minimizar el impacto de la herramienta de desarrollo en la realización de los ejercicios.
5. Presentar a los alumnos técnicas básicas de análisis y diseño orientado a objetos así como los elementos mínimos de la notación UML.
6. Evitar, en la medida de lo posible, problemas habituales en este tipo de iniciativas como la utilización de entornos artificiosos, el énfasis en el desarrollo de aplicaciones gráficas o la presencia excesiva de bibliotecas de clases.

En base a tales objetivos y teniendo en cuenta el tiempo disponible se elaboró el programa de la asignatura cuyos bloques teóricos y prácticos principales se muestran en Figura 1 y Figura 2. En los apartados siguientes se explicará de forma detallada la forma en que se están impartiendo estos contenidos.

5.1. Contenidos teóricos

Los contenidos teóricos de la asignatura se reparten en siete bloques. El primero permite presentar a los alumnos una serie de conceptos fundamentales (en especial el de algoritmo) así como la forma en que la informática ha evolucionado para resolver la antigua necesidad de realizar cálculos y procesar datos de manera automatizada. Además, se explica a los estudiantes el funcionamiento de la arquitectura Von Neumann, la forma en que se representa la información en un ordenador y los distintos tipos de lenguajes de programación existentes.

1. ANTECEDENTES
   - 1.1. Historia de la Informática
   - 1.2. Lenguajes de programación
2. INTRODUCCIÓN A LA PROGRAMACIÓN ORIENTADA A OBJETOS
   - 2.1. Paradigma OO.
   - 2.2. Encapsulación y ocultación de información.
   - 2.3. Abstracción y genericidad.
   - 2.4. Comportamiento de los objetos.
   - 2.5. Constructores y destructores.
   - 2.6. Recolección de basura.
3. ESTADO DE LOS OBJETOS
   - 3.1. Tipos de datos. Tipos primitivos.
   - 3.2. Declaración de atributos
   - 3.3. Expresiones y asignación.
   - 3.4. Comprobación de tipos.
4. COMPORTAMIENTO DE LOS OBJETOS
   - 4.1. Métodos.
   - 4.2. Paso de parámetros.
   - 4.3. Declaración de variables
   - 4.4. Sobrecarga de métodos.
5. ESTRUCTURAS DE CONTROL
   - 5.1. Propiedades de las estructuras de control.
   - 5.2. Estructura secuencial.
   - 5.3. Estructuras alternativas.
   - 5.4. Estructuras repetitivas.
6. CLASIFICACIÓN DE OBJETOS
   - 6.1. Clases y subclases.
   - 6.2. Interfaces.
   - 6.3. Herencia de clases e interfaces. Redefinición de métodos.
   - 6.4. Jerarquías de clases.
7. ESTRUCTURAS DE DATOS FUNDAMENTALES
   - 7.1. Arrays.
   - 7.2. Cadenas de caracteres.

Figura 1. Temario teórico de "Introducción a la Programación"

Prácticas de tablero

1. INTRODUCCIÓN A LA PROGRAMACIÓN ORIENTADA A OBJETOS
2. ESTADO DE LOS OBJETOS
3. COMPORTAMIENTO DE LOS OBJETOS
4. ESTRUCTURAS DE CONTROL
5. CLASES DE OBJETOS
6. ESTRUCTURAS DE DATOS FUNDAMENTALES

Prácticas de laboratorio

1. PRESENTACIÓN DEL ENTORNO DE DESARROLLO.
2. COMPORTAMIENTO DE LOS OBJETOS (I).
3. DOTANDO DE ESTADO A LOS OBJETOS.
4. COMPORTAMIENTO DE LOS OBJETOS (II).
5. JERARQUÍAS DE CLASES.
6. DESARROLLO DE UNA APLICACIÓN COMPLETA.

Figura 2. Temario práctico de "Introducción a la Programación"

Una vez los alumnos han asimilado el concepto de algoritmo es posible plantearles sencillos problemas para los que se puede encontrar un algoritmo sin demasiadas dificultades. Sin embargo, la finalidad de esos ejemplos no es explicar el paradigma procedimental sino exponer sus debilidades.

Para ello, se pide a los estudiantes que traten de imaginar un algoritmo que resuelva un problema muy complejo (por ejemplo, gestionar un sistema de control de tráfico aéreo). Naturalmente, coinciden en que desarrollar semejante algoritmo puede resultar muy arduo.

A continuación se pide a los alumnos que tra-

ten de identificar objetos que sean relevantes en el ámbito de dicho problema, así como el comportamiento que mejor los defina. Esta tarea no les resulta demasiado complicada y sí bastante motivadora.

Tras este contacto intuitivo con el paradigma se dedican al menos tres sesiones a explicar los conceptos básicos del mismo. Aquí se tocan los temas fundamentales de manera breve y tratando de mantener una visión global: diferencias entre clase y objeto, comportamiento y estado, métodos y atributos, constructores, herencia, etc.

La mayoría de los alumnos no comprenden de una forma clara muchos de los conceptos más difíciles; es necesario insistir en que lo fundamental en este momento no es comprenderlo todo sino saber de su existencia y conocer, a grandes rasgos, las relaciones entre los distintos conceptos.

Una vez se ha presentado una panorámica de la POO se puede comenzar a entrar en detalles referentes a estado y comportamiento. Así, al explicar el concepto de estado es posible introducir los conceptos de tipos de datos, tipos primitivos, declaración de variables miembro, etc. Al explicar el comportamiento de los objetos deberían tocarse temas especialmente relevantes como definición de métodos, variables locales, paso de parámetros, sobrecarga, etc. junto con las estructuras de control básicas.

Finalmente se explicarían de manera muy sencilla herencia, interfaces, redefinición de métodos y se introduciría brevemente el polimorfismo.

Por último, se explicarían los principales métodos de la clase `String` y *arrays*. La razón para retrasar hasta el final de la asignatura el concepto de *array* es simple. Mientras los alumnos no manejen adecuadamente los conceptos de clase y objeto además del mecanismo de herencia no es posible desarrollar ejercicios relativamente complejos; y será en dichos ejercicios donde los *arrays* sean verdaderamente útiles para poder implementar de una forma básica relaciones 1..n.

Respecto a la forma de representar los conceptos de cara a su explicación hemos optado por emplear la notación UML. Sin embargo, no hemos hecho ningún esfuerzo específico por enseñar la notación a los alumnos; simplemente hemos ido introduciendo símbolos a medida que eran necesarios. En estos momentos en los que la asignatura está prácticamente finalizada los alumnos son capaces de desarrollar por sí solos modelos con varias clases, empleando adecuadamente relaciones de herencia y asociaciones con multiplicidad.

5.2. Lenguaje de programación elegido

Para el desarrollo de las prácticas era posible elegir entre muchos lenguajes: Smalltalk, Eiffel, C++, Java, etc. Todos presentan inconvenientes a la hora de ser empleados como un lenguaje para enseñanza, resultando la opción "menos mala" Java [11].

Como argumentos a favor del uso de Java podemos citar los siguientes:

1. Es un lenguaje OO prácticamente puro.
2. La sintaxis es sencilla y legible, y permite expresar la mayor parte de conceptos OO de una manera simple.
3. Es un lenguaje relativamente pequeño.
4. Los alumnos pueden trabajar con él sin tener que conocer un número excesivo de clases de la biblioteca.
5. Sólo permite herencia simple.
6. El alumno no tiene que preocuparse de la reserva y liberación de memoria.
7. Es ampliamente utilizado por la industria y facilita la transición a otros lenguajes OO.

Obviamente, Java presenta una serie de inconvenientes, entre los que se pueden mencionar:

1. La existencia de tipos simples puede confundir al principio a los estudiantes.
2. La entrada/salida requiere la utilización de clases de la biblioteca del lenguaje y resulta, cuando menos, engorrosa.
3. Hasta que los alumnos se acostumbran a que es un lenguaje sensible a mayúsculas las clases contenedoras de tipos simples como `Boolean`, `Integer` o `Float` pueden causar más de un problema.
4. El método `main` es totalmente ajeno a los conceptos OO y precisa entre una y dos clases para ser correctamente explicado obligando, además, a introducir el concepto de métodos estáticos muy pronto.

De los problemas anteriores los más graves son el segundo y el cuarto; a continuación se describirá la forma en que hemos tratado de solucionarlos en la asignatura que impartimos.

Para llevar a cabo entrada y salida en Java sería necesario, en principio, explicar a los alumnos en un momento muy temprano la existencia de la biblioteca de clases de Java para

poder emplear los métodos de las clases `System.out` y `System.in`. Para evitar esto optamos por construir un paquete `Consola` que implementa las clases `Pantalla` y `Teclado` y explicamos los métodos para dichas clases (por ejemplo, `escribir` o `leerFlotante`).

Las diferencias pueden parecer triviales pero son importantes para los alumnos:

1. El alumno percibe perfectamente que `Pantalla` y `Teclado` son elementos ajenos al lenguaje; después de todo fueron escritas por los profesores y tienen "nombres en español". Esta tarea no es tan sencilla al emplear clases aportadas por la biblioteca de Java.
2. La sentencia `import` es relativamente sencilla de entender si se explica en términos de directorios y subdirectorios.
3. Es posible presentar las clases `Pantalla` y `Teclado` en una lección de teoría para que los alumnos vean qué es exactamente un objeto, la forma en que proporciona un comportamiento para cumplir con sus responsabilidades así como la forma en que se invocan mensajes.

Explicar el método `main` de una manera razonable es algo más complicado. En primer lugar, requiere que el alumno comprenda perfectamente el concepto de algoritmo, la arquitectura Von Neumann y la forma en que ésta permite ejecutar algoritmos.

Una vez estos conceptos están claros es necesario explicar las diferencias entre lenguajes de bajo y de alto nivel así como el papel jugado por los distintos tipos de traductores.

En ese momento se puede explicar de un modo muy sencillo la forma en que Java depende de una máquina virtual (JVM) y cómo el código fuente Java es traducido a *bytecode* que será ejecutado por la JVM que, a su vez, se ejecutará sobre una arquitectura Von Neumann.

Si todo esto queda claro a los alumnos (lo cual puede llevar toda una sesión de teoría) pueden entender que, de algún modo, aunque Java sea un lenguaje OO tiene que acabar siendo traducido a código de bajo nivel que se ejecutará en una arquitectura Von Neumann. Puesto que dicha arquitectura requiere que un algoritmo tenga un punto de inicio y las instrucciones se ejecutan secuencialmente no les resulta difícil aceptar que de alguna forma es necesario indicar en el propio código Java qué método de qué clase será el primero en ejecutarse.

Por supuesto, es necesario indicar a los alumnos que el método `main` es ajeno al paradigma OO y que el código que se coloque en su interior debería ser siempre el menor posible.

5.3. Herramientas de desarrollo

Un problema común a la hora de impartir una asignatura de programación se plantea en el momento de seleccionar el entorno de desarrollo, especialmente si se trata de un lenguaje no orientado al ámbito académico.

En nuestro caso debíamos seleccionar un entorno[2] que permitiese el desarrollo de aplicaciones OO en Java y satisficiese, en la medida de lo posible, los siguientes criterios [12]: facilidad de uso, entorno integrado (editor, compilador y depurador), IGU con soporte para manipulación de objetos y libre disponibilidad para los estudiantes.

Las opciones que se plantearon fueron las siguientes: Kawa, BlueJ, Borland JBuilder Personal y Visual J++.

Kawa[3] era un sencillo IDE proporcionado por *Macromedia*; sin embargo, actualmente carece de soporte y no tenía sentido animar a los estudiantes a utilizar una herramienta sin futuro.

BlueJ[4] es la única de las herramientas anteriores específicamente pensada para su utilización académica. Es gratuita, muy fácil de usar y permite trabajar en un modo "visual" empleando UML. Además, el alumno puede instanciar objetos y enviarles mensajes directamente, con lo cual dos problemas de Java se solucionan de una forma sencilla: la entrada/salida y el método `main`.

Borland JBuilder Personal[5] es una versión muy reducida de la herramienta Borland JBuilder

[2] ¿Por qué no utilizar simplemente el JDK? Puede resultar muy instructivo enfrentarse a compiladores y depuradores en línea de órdenes. Sin embargo, la finalidad de la asignatura es aprender POO empleando Java y no aprender Java *per se*; por ello, el uso didáctico del JDK carecía de sentido. Además, requeriría enseñar a la mayoría de los alumnos MS-DOS consumiendo un tiempo precioso. Por último, aun cuando fuesen capaces de emplear el JDK, la ausencia de entorno les distraería de lo realmente importante: la POO.

[3] http://www.macromedia.com/software/kawa

[4] http://www.bluej.org

[5] http://www.borland.com/jbuilder/personal

siendo, por tanto, muy similar a otros IDE's comerciales. La principal ventaja de esta herramienta es que cualquier usuario puede obtener una licencia de uso personal de manera gratuita. Visual J++ es similar aunque más complicado, razón por la que fue descartado.

Así pues, había dos opciones: BlueJ y JBuilder. Por motivos didácticos la opción más deseable era BlueJ. No obstante, su utilización planteaba dudas sobre posibles problemas por parte de los alumnos al tener que cambiar a un IDE "real".

Si dicho cambio se produjese en un nuevo curso habríamos elegido BlueJ; sin embargo, por diversas razones la asignatura "Interacción Persona-Ordenador" (IPO) se imparte en el segundo cuatrimestre del primer curso. Al no encontrarse ningún entorno sencillo para el diseño de IGU's que pudiese ser integrado con BlueJ nos vimos obligados a emplear Borland JBuilder Personal para no obligar a los alumnos a aprender una herramienta distinta en cada cuatrimestre. Fueron necesarias dos sesiones prácticas para presentar el entorno y muchos alumnos no comenzaron a manejarlo de manera autónoma hasta la tercera o la cuarta sesión.

5.4. Ejercicios prácticos

Para superar la asignatura los estudiantes deben aprobar un examen práctico o bien superar con éxito dos ejercicios que son desarrollados en gran parte durante las sesiones prácticas y finalizados fuera del horario docente.

El objetivo del primer ejercicio es consolidar dos conceptos básicos de la POO como son estado y comportamiento. Para ello el alumno parte de una clase básica denominada `Inichi` que implementa una sencilla mascota virtual [3].

Para superar el ejercicio el alumno debe añadir a su mascota una serie de métodos y atributos que la doten de un comportamiento más complejo. Una parte importante de la evaluación de este ejercicio obliga al alumno a realizar, bajo condiciones controladas, una pequeña ampliación que requiere algunos atributos y métodos nuevos.

En el segundo ejercicio [4] el alumno debe desarrollar una aplicación completa que requiere el uso de varias clases. Para ello puede optar por desarrollar su propio modelo de clases o bien ampliar uno de los distintos modelos propuestos por los profesores. Este ejercicio se entregará el mismo día del examen escrito.

5.5. Resultados parciales

En estos momentos la asignatura está a punto de finalizar; los alumnos ya han realizado un control teórico y se ha evaluado el primer ejercicio práctico, estando pendientes el examen escrito, la entrega del segundo ejercicio práctico y el examen práctico para quienes no sean evaluables mediante el sistema de evaluación continua.

El citado control se llevó a cabo tras apenas 6 semanas de clase y abarcó los tres primeros bloques del temario. El 85% de los matriculados se presentaron al examen y se obtuvieron los resultados que se muestran en Figura 3. En cuanto a la evaluación de la primera práctica se han obtenido los siguientes resultados: la han presentado el 81% de los matriculados, el 9% de los ejercicios eran "fraudulentos" (copias, prácticas en equipo, etc), de las prácticas válidas un 14% suspendieron, un 33% aprobaron, un 28% obtuvieron notable y un 25% sobresaliente.

Además, tras ocho semanas de clase se realizó una encuesta entre el alumnado que había cursado Metodología de la Programación I (una asignatura del anterior Plan de Estudios que comenzaba con programación estructurada y finalizaba con algunos conceptos de POO). Los resultados se muestran en Figura 4.

Figura 3. Resultados del control teórico

Paradigma preferido	¿Dificultad teoría POO?	¿Dificultad práctica POO?	Factores influyen percepción dificultad
78% OO	30% Más difícil	19% Más difícil	37% Cambio de Pascal a Java
6% Indiferente	22%Similar	23% Similar	49% Comenzar con POO
16% Progr. Estructurada	48% Más fácil	58% Más fácil	40% Haber estudiado antes programación

Figura 4. Resultados de la encuesta entre repetidores

6. Conclusión

Esta experiencia de enseñanza de la OO como único paradigma de programación está resultando muy positiva. Creemos que no resulta más difícil para el alumno que la programación estructurada y, además, el grado de satisfacción de los alumnos con el enfoque, el lenguaje y la herramienta de desarrollo son altos. Creemos que esto se debe a una serie de factores motivadores: el alumno aprende un lenguaje que se emplea en la industria, utiliza una versión de una herramienta profesional y se siente especialmente motivado por el propio paradigma. No pensamos que esto se deba a que la OO sea más intuitiva o sencilla que la programación estructurada; simplemente es más potente a la hora de enfrentarse a problemas reales y creemos que esta percepción de poder por parte de alumnos de primer curso es lo que les alienta.

Referencias

[1] ACM/IEEE. *Computing Curricula 2001*. http://www.computer.org/education/cc2001/report

[2] Bergin, J. *Why Procedural is the Wrong First Paradigm if OOP is the Goal*. http://csis.pace.edu/~bergin/papers/Whynotproceduralfirst.html

[3] Díaz Fondón, M., García Fuente, A., Gayo Avello, D. *Desarrollo de una mascota virtual.* http://www.euitio.uniovi.es/~ip/practica09.pdf

[4] Díaz Fondón, M., García Fuente, A., Gayo Avello, D., Redondo López, J.M. *Evaluación de conductores con dispositivos PDA*. http://www.euitio.uniovi.es/~ip/practica10.pdf

[5] Fernández Muñoz, L., Peña, R., Nava, F., Velázquez Iturbide, A. *Análisis de las propuestas de la enseñanza de la programación orientada a objetos en los primeros cursos*. JENUI 2002. Cáceres, España.

[6] Fernández Muñoz, L., Peña, R., Nava, F., Velázquez Iturbide, A. *Enfoque diacrónico para la enseñanza de la programación imperativa*. JENUI 2002. Cáceres, España.

[7] Gabriel, R.P. *Objects have failed*. http://www.dreamsongs.com/NewFiles/ObjectsHaveFailed.pdf

[8] García Molina, J. *¿Es conveniente la Orientación a Objetos en un primer curso de programación?* Novática 154, 2001.

[9] García Molina, J., Menárguez Tortosa, M., Moros Valle, B. *Una propuesta para organizar la enseñanza de la Orientación a Objetos*. JENUI 2002. Cáceres, España.

[10] Gómez Albarrán, M. *Metodología basada en descomposición funcional y orientación a objetos en la introducción a la programación*. JENUI 2002. Cáceres, España.

[11] Kölling, M. *The Problem of Teaching Object-Oriented Programming, Part 1: Languages*. Journal of Object-Oriented Programming, Vol. 11 No. 8, 8-15, 1999. http://www.mip.sdu.dk/~mik/papers/oo-languages.pdf

[12] Kölling, M. *The Problem of Teaching Object-Oriented Programming, Part 2: Environments*. Journal of Object-Oriented Programming, Vol. 11 No. 9, 6-12, 1999. http://www.mip.sdu.dk/~mik/papers/oo-environments.pdf

[13] Steele Jr., G.L. *Objects have not failed*. http://www.dreamsongs.com/ObjectsHaveNotFailedNarr.html

[14] Stroustrup, B. *The Design and Evolution of C++*. Addison-Wesley, 1994.

Las asignaturas de programación en las universidades españolas

Julia González, Alberto Gómez
Departamento de Informática
Universidad de Extremadura
10071 Cáceres
e-mail: {juliagon, agomez}@unex.es

Resumen

En este trabajo se presenta un estudio comparativo de las asignaturas de programación impartidas en los dos primeros cursos de las carreras de Informática en las Universidades españolas, junto con algunos comentarios sobre la última versión de una de las recomendaciones curriculares internacionales más citadas, la de ACM e IEEE. El objetivo es comprobar si hay demasiadas diferencias en los contenidos mínimos exigidos y en el enfoque utilizado en las Universidades de nuestro entorno.

Los datos que se han utilizado son los disponibles en las diversas páginas Web, tanto institucionales como propias de las asignaturas. Por este motivo, quizás la información no sea completa, aunque pensamos que sí es significativa.

1. Introducción

El espacio único de educación europeo propuesto en la Declaración de Sorbona en 1998, y refrendado en 1999 con la Declaración de Bolonia, provoca una revisión y adaptación de todos los planes de estudio de las titulaciones impartidas actualmente en las Universidades españolas [1].

Las titulaciones deberán acomodarse a los nuevos títulos de pregrado y grado, el primero con valor específico en el mercado de trabajo y el segundo que conducirá a la obtención de un master y/o doctorado.

Además se establece un sistema global para el cómputo de la carga lectiva, los créditos ECTS. Este sistema computa el volumen total del trabajo que el estudiante ha de realizar para superar una materia, incluyendo tanto su asistencia a clase como el tiempo empleado en su evaluación o el trabajo personal realizado. Este nuevo sistema busca la comparación y homologación de títulos entre las Universidades europeas.

De acuerdo al nuevo reto de convergencia, la Conferencia de Decanos y Directores de Centros de Informática, CODDI, ha elaborado un informe preliminar [2] en el que se ratifica la necesidad de un único título de pregrado que de acceso a la profesión y a un segundo ciclo de especialización. La titulación propuesta tendría una duración de 4 años y una carga lectiva comprendida entre 224 y 260 créditos ECTS.

Según este informe, nuestras titulaciones, divididas actualmente en primer y segundo ciclo, han de modificarse, pasando a un único título de pregrado. La convergencia hacia un único título implica una reubicación de recursos, una nueva planificación y una adaptación de los contenidos, que han de verse alterados por las nuevas necesidades y carga lectiva asignada.

Este nuevo planteamiento supondrá un gran esfuerzo de las Universidades españolas que, actualmente y de forma mayoritaria, imparten más de una titulación de Informática. Tal y como refleja la figura 1, el 78% de las Universidades españolas, datos obtenidos del Ministerio de Educación, Cultura y Deporte, imparten actualmente más de un título en Informática [3].

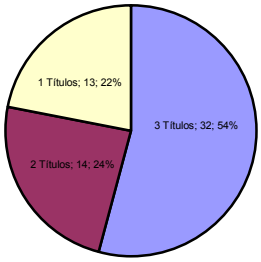


Figura 1. Universidades clasificadas por el número de titulaciones que imparten.

En el informe de la CODDI se propone un reparto porcentual de los créditos, donde el 55% corresponderá a materias troncales. El 7% del total estará destinado a la materia Programación y Estructuras de datos, lo que nos lleva a 18 créditos ECTS troncales en esta materia. Actualmente esta materia está reflejada en los Reales Decretos 1459/1999, 1460/1990 y 1461/1990, como las materias Metodología de Programación y Estructura de Datos y de la Información, con un total de 27 créditos y una carga aproximada del 10% del total de la titulación.

Esta nueva situación, nos lleva a marcar como objetivo de este trabajo el estudio de la situación actual de las materias relacionadas directamente con la enseñanza de la programación en las Universidades españolas, que nos sirva como reflexión sobre la situación actual y punto de partida para los cambios que se avecinan.

En el siguiente apartado se presentan brevemente las últimas recomendaciones relacionadas con la enseñanza de la programación de ACM e IEEE. En el apartado 3 se revisa la situación actual de las asignaturas en las Universidades españolas, según la información encontrada en las páginas web. Para terminar, realizamos algunos comentarios sobre el estudio y presentamos las conclusiones.

2. La programación en el Computing Curricula 2001 de ACM e IEEE

El objetivo principal del Computing Curricula'2001 de ACM e IEEE [4] es revisar el anterior modelo de 1991 e incorporar los desarrollos e innovaciones producidos en la última década.

Una de las principales novedades de este currículum es la división del informe en varios volúmenes. El primero de ellos y objeto de este estudio, Computing Curricula, se corresponde con una visión global de los estudios de Informáticas. Además, están desarrollándose cuatro volúmenes más pertenecientes a diferentes disciplinas: Computer Science, Computer Engineering, Software Engineering e Information System.

En la elaboración del informe se aseguró la representación de un amplio rango de expertos en áreas distintas, para lo que se establecieron 20 grupos de trabajo, cada uno destinado a un área, divididos en dos categorías distintas: 14 áreas de conocimiento y 6 áreas de pedagogía

Cada una de las áreas de conocimiento propuestas, dividen sus contenidos en unidades, y cada unidad en temas. Además, se selecciona un conjunto de unidades básicas o troncales, ampliamente consensuadas como indispensables en cualquier currículum. El resto de temas se definen como no troncal. De forma explícita, el informe asegura que estas materias troncales no conforman el currículum en sí mismo, aunque son imprescindibles, y no es necesario que estén concentradas en los primeros cursos.

Con respecto a los cursos introductorios, se proponen diversas implementaciones, dependiendo de si vamos a optar por una aproximación de programación o bien basada en hardware. Eligiendo la primera opción, que es la

más extendida y la que nos interesa en este trabajo, las implementaciones posibles son:

- Primero imperativo.
- Primero objetos.
- Primero funcional.

Temas	Horas mínimas
Fundamentos de programación	
PF1. Constructores de programación fundamentales	9
PF2: Algoritmos y resolución de problemas	6
PF3: Estructuras de datos fundamentales	14
PF4: Recursividad	5
PF5: Programación orientada a eventos	4
Algoritmos y complejidad	
AL1: Análisis de algoritmos	4
AL2: Estrategias de algoritmos	6
AL3: Algoritmos fundamentales de computación	12
Lenguajes de programación	
PL1: Introducción a los lenguajes de programación	3
PL4: Declaraciones y tipos	3
PL5: Mecanismos de abstracción	3
PL6: Programación orientada a objetos	10
Total	**79**

Tabla 1. Temas cubiertos en las tres aproximaciones de programación propuestos por el Computing Curricula '01

En las tres aproximaciones las unidades tratadas, ver tabla 1, coinciden. Se diferencian en la temporalización en cursos y en el orden de presentación de estos conceptos.

Este documento es uno de los más citados y consultados a nivel internacional; resulta adecuado que sea tenido en cuenta en los futuros planes de estudio.

Según el documento de la CODDI citado anteriormente, son 18 los créditos ECTS troncales destinados a programación y estructuras de datos, con una conversión de 25 horas de trabajo por crédito, da un total de 450 horas, que pueden dividirse entre los dos primeros años de carrera. El CC'01 propone un mínimo de 79 horas de clases magistrales, para implementar el currículo básico en esta materia, por lo que su planificación está en consonancia con nuestras necesidades temporales, dando un margen de libertad a las Universidades y al profesorado para incluir materias no recogidas en este informe.

3. Programación y estructuras de datos en las Universidades españolas

En 1990 se publican los Reales Decretos que actualmente regulan las titulaciones en Informática [5] [6] [7]. Estos decretos establecen los descriptores básicos de las materias troncales "Metodología y tecnología de la programación" y "Estructuras de datos y de la información", donde se incluyen todos los conceptos de programación básicos para las titulaciones de Informática. Para

las titulaciones técnicas se establece una carga lectiva de 12 créditos para cada una, y para la titulación de Ingeniería Informática de 15 créditos.

En las tres titulaciones los descriptores son los mismos. Para "Metodología y tecnología de la programación":

- Diseño de algoritmos
- Análisis de algoritmos
- Lenguajes de programación
- Diseño de programas: descomposición modular y documentación
- Técnicas de verificación y prueba de programas

Para la materia troncal "Estructuras de datos y de la Información", que incluye además conceptos de bases de datos:

- Tipos abstractos de datos
- Estructura de datos y algoritmos de manipulación
- Estructuras de la información: Ficheros y bases de datos

Los descriptores de las materias, pese a su brevedad, coinciden con los temas propuestos en el CC'01, excepto en lo referente a las bases de datos, que al igual que en el informe de la CODDI, se trata como área independiente, y la documentación de programas. Si prevé en cambio la inclusión de la programación orientada a objetos y la programación orientada a eventos, no recogida en las directrices del Ministerio, probablemente por los años pasados desde el momento de su especificación, aunque estos paradigmas se imparten de forma habitual en las Universidades españolas.

El estudio se ha realizado a través de la información disponible en las páginas web institucionales y de profesores de las Universidades españolas, tanto privadas como públicas. Se han considerado las tres titulaciones, teniendo en cuenta que las materias objeto de estudio en este trabajo poseen los mismos descriptores.

A continuación se presentan los datos obtenidos, clasificados en cuatro bloques: paradigma de programación y lenguaje utilizado, temario, prácticas y evaluación, y bibliografía.

3.1. Paradigma y lenguaje

La elección del paradigma de programación y el lenguaje utilizado para realizar las prácticas es uno de los temas de debate continuos en nuestra comunidad. Tras realizar un análisis de todas las Universidades públicas españolas, y algunas Universidades privadas, se puede concluir que no existe consenso en este tema, tal y como muestra la tabla 2.

Según los datos obtenidos, el lenguaje C/C++ es el más utilizado, seguido de Pascal y Java. Debemos tener en cuenta que el lenguaje Java es uno de los más recientes, y que debido a su popularidad en el mundo profesional y a sus innegables características, en unos años probablemente será uno de los más utilizados. Apoyados en el lenguaje se van a desarrollar los contenidos teórico-prácticos de la materia, por lo que resulta interesante conocer cuál es el paradigma de programación más utilizado actualmente.

Los datos muestran que el 7% de las Universidades utiliza desde el primer momento la programación orientada a objetos y un 79% el paradigma de programación imperativa. Existe un 14% para el que no poseemos datos.

	Nº Asignaturas	Nº Universidades
C/C++	26	16
Pascal	21	11
Java	11	8
Ada	9	5
Modula 2	2	1
Haskell	1	1

Tabla 2. Primer lenguaje de programación en las titulaciones de Informática españolas

De este 79% que empieza con programación imperativa, el 56% presenta también en el primer curso, aunque con posterioridad, la programación orientada a objetos y el 44% restante mantiene el paradigma de programación imperativa.

Según estos mismos datos, al término de los dos primeros cursos, el 98% de las Universidades presentan los dos paradigmas de programación, tratando la programación orientada a eventos y la programación funcional y lógica en asignaturas no obligatorias, en caso de ser consideradas.

3.2. Temario

A continuación realizaremos algunos comentarios sobre aspectos relevantes de los temarios de las asignaturas que constituyen las materias de programación en las titulaciones de Informática; como son el tratamiento de la recursividad, los tipos abstractos de datos, la documentación, etc. Hemos elegido estos temas porque suon en los que más difiere la implementación de los planes de estudio. Además hemos recogido la documentación de programas por no estar reflejado en el curricula CC'01 y la programación orientada a eventos porque no está definida dentro de los descriptores de las asignaturas.

- Recursividad

25 de las Universidades estudiadas tratan la recursividad como un tema específico dentro de los contenidos de las asignaturas de primer año, en 12 de estas 25 Universidades se retoma este tópico en asignaturas de segundo año y en 5 Universidades se introduce por primera vez en segundo curso.

- Programación orientada a eventos

Las Universidades de Almería, Málaga, Oviedo y Vigo son las únicas que incluyen este paradigma en las asignaturas de programación, troncales u obligatorias, de los primeros años. Las dos primeras utilizan como lenguaje de programación Java, la universidad de Oviedo implementa sobre Pascal y la universidad de Vigo utiliza C++.

- Tipos abstractos de datos

El concepto de Tipo Abstracto de Datos, TAD, se presenta en el 90% de las Universidades. Además del concepto, se recoge la implementación de las estructuras Pila, Cola, Lista y Conjunto como tipos abstractos de datos y se utilizan en diversos algoritmos.

- Documentación

Sorprendentemente, la documentación no se trata como tema exclusivo en nuestras universidades, y de estar incluido en los temarios como tema independiente, es uno de los últimos. Aunque suponemos que aparecerá como contenido transversal a lo largo de los cursos, intuimos que primero se les enseña a crear programas y después a documentarlos, y no a hacerlo de forma paralela. Creemos que esta no es una buena práctica y que los temas de documentación, principalmente la interna deben realizarse de forma anterior al diseño y paralela a la implementación.

- Especificación algebraica y Verificación formal.

Los métodos formales están incluidos como contenidos teóricos, preferentemente en asignaturas de segundo curso relacionadas con las estructuras de datos. El 50% de las universidades estudiadas incluyen temas específicos destinados a la verificación formal, pero solamente 8 universidades destinan al menos un tema a la especificación algebraica.

3.3. Practicas y evaluación

Mayoritariamente, en todas las titulaciones observadas, se combinan las clases magistrales (horas de contenido teórico) con clases en los laboratorios, de contenido práctico. Ambas están directamente relacionadas, dependiendo el contenido práctico de la explicación teórica y raramente separado, aunque existen Universidades que poseen asignaturas independientes para cada una de estas modalidades, como las Universidades de Alcalá, Complutense de Madrid, Deusto, Europea de Madrid, Extremadura y Vigo. Suponemos que al igual que ocurre en nuestra Universidad, esta división es formal y la dependencia es estrecha entre las asignaturas teóricas y prácticas.

Debido a la fuente de información, es difícil determinar el sistema de evaluación seguido mayoritariamente, aunque sí observamos que el alumno trabaja en proyectos de programación, y suponemos que éstos son evaluados de forma

independiente a los contenidos teóricos, aunque no podemos determinar la forma ni el peso que tienen.

3.4. Bibliografía

Existe una amplia bibliografía en temas relacionados con la programación. No obstante y sorprendentemente, la bibliografía utilizada en las diferentes Universidades es bastante homogénea y puede ser clasificada en dos bloques:

- Bibliografía genérica
- Bibliografía dependiente del lenguaje de implementación utilizado.

Existe un grupo de libros básicos o clásicos recomendada mayoritariamente. En este grupo los autores que más destacan son:

1. Aho [8]
2. Horowith [9]
3. Wirth [10][11]
4. Brassard [12][13]
5. Weiss[14]
6. Joyanes [15]
7. Castro[16]
8. Peña [17]
9. Balcázar [18]

Los libros y autores citados anteriormente se utilizan en los dos primeros cursos de las titulaciones. Los libros [15] y [16] se destinan a cursos de iniciación a la programación, mientras que [9], [12], [13], [14], [17] y [18] son destinados a contenidos avanzados, normalmente en segundo curso.

Se ha observado que a través del servicio de publicaciones de las distintas Universidades se elaboran libros que rara vez consiguen salir del ámbito de la universidad en la que son creados, ejemplos de ello existen en la Universidad de Extremadura, Las Palmas de Gran Canaria, Málaga, Murcia. Cabe destacar el libro [19], publicado por la Universidad Politécnica de Cataluña, muy referenciado por el resto de Universidades.

El resto de bibliografía utilizada está muy influida por el lenguaje de programación elegido como soporte a los conocimientos teóricos. Como se señaló en la sección anterior, los lenguajes más utilizados son C++, Java y Pascal, seguidos de Ada y Modula-2 y Cobol. En estos casos, la bibliografía utilizada no es tan homogénea, debido principalmente al elevado número de volúmenes dedicados a estos lenguajes.

De la bibliografía revisada destacar que no hemos encontrado libros destinados a la documentación de programas, y que a menudo este aspecto no está tratado en la bibliografía fundamental, aunque el tema sí posea gran relevancia.

4. Comentarios al estudio

Aunque la información con la que se ha realizado este estudio no es completa ni homogénea, ya que no se pueden encontrar los mismos datos de esta materia en las páginas Web de todas las Universidades, se pueden extraer algunas conclusiones útiles sobre la materia de programación impartida en los estudios de Informática.

En general, las características académicas de las asignaturas son similares. Suele ser una asignatura anual, o dividida en dos cuatrimestres, entre 12 y 18 créditos por año, impartida en los dos primeros cursos de carrera, que normalmente están complementadas con asignaturas obligatorias y optativas con el objetivo de profundizar en algunos de los aspectos tratados en las troncales.

Los temarios de las asignaturas consideradas son bastante parecidos en cuanto a las materias que se tratan. En la mayoría de los casos, los primeros temas se dedican a los fundamentos de programación, construyendo poco a poco estructuras de datos cada vez más complejas, viendo por último los esquemas algorítmicos más utilizados y algoritmos clásicos.

La mayor diferencia se detecta en el nivel de formalismo utilizado y el nivel de abstracción tratado.

En general una deficiencia encontrada es la poca atención prestada a la documentación, prueba y mantenimiento de programas. A menudo estos temas quedan pendientes de materias relacionadas con Ingeniería del Software, tratadas en cursos posteriores, aunque tenemos el convencimiento de que son temas fundamentales que deben presentarse desde el principio.

La bibliografía recomendada es bastante similar en todos los casos. Los libros que más se repiten son los considerados claves en la literatura.

Como trabajo futuro, es recomendable completar el estudio con datos homogéneos no encontrados en las páginas web de las Universidades, para recopilar información de los pasos que cada Universidad está llevando a cabo en su adaptación al nuevo espacio europeo.

5. Conclusiones

A la vista del estudio comparativo de las materias de programación en las Universidades españolas, resulta ser bastante similar en cuanto a características académicas, temario, lenguajes de programación utilizados y bibliografía.

Por este motivo, la adaptación de los temarios al nuevo espacio de educación europeo no debe ser complicada. El principal problema puede ser la adaptación al nuevo sistema de cómputo de carga lectiva. Según la propuesta elaborada por la CODDI, se rebaja en un 3% la carga dedicada a programación y estructura de datos. Esta modificación, junto con la cuantificación del trabajo realizado por el alumno individualmente, quizás suponga un recorte en horas lectivas, lo que implicará un recorte en los temarios o en la profundización de los contenidos o en las horas dedicadas a las prácticas tuteladas.

La situación de cambio en la que estamos embarcados, debe ayudarnos a sentar las bases de la definición del perfil profesional buscado, y según esto, adaptar los contenidos, modernizándolos y ajustándolos a las necesidades de nuestra sociedad.

Agradecimientos

Queremos agradecer el esfuerzo de las distintas Universidades y de los profesores de las asignaturas que han puesto esta valiosa información al alcance de todos en las páginas web, y que se encargan de mantenerla actualizada.

También pedimos disculpas por no haber sabido encontrar todos los datos que las Universidades pueden tener accesibles a través de Internet.

Referencias

[1] www.crue.org/espaeuro
[2] http:// webepcc.unex.es/jenui2002
[3] Ministerio de educación, cultura y deporte. http: //www.mec.es
[4] ACM/IEEE. Computing Curricula 2001. Computer Science. Final Report. Decempber 15th. The Joint Task Force on Computing Curricula, 2001.
[5] RD 1459/1999. Real decreto de regulación de la titulación Ingeniero Informático.
[6] RD 1460/1990. Real decreto de regulación de la titulación Ingeniero Técnico en Informática de Gestión.
[7] RD 1461/1990. Real decreto de regulación de la titulación Ingeniero Técnico en Informática de Sistemas.
[8] Aho, Alfred; E. Hopcroft, John y D. Ullman, Jeffrey. "Estructuras de datos y algoritmos". Wilmington,"Addison-Wesley Iberoamericana 1988.
[9] Horowitz Ellis & Sahni, Sartaj. "Fundamentals of Computer Algorithms". Pitman, 1978.
[10] Wirth, N. "Algoritmos + estructuras de datos = programas". Ediciones del Castillo,1980.
[11] Wirth, N. "Algoritmos y Estructuras de datos". Prentice-Hall, 1986.
[12] Brassard y P. Bratley. "Algorítmica. Concepción y análisis". G. Ed. Masson, 1990.
[13] Brassard, Gilles & Bratley, Paul. "Fundamentos de Algoritmia". Prentice-Hall, 1997.
[14] Weiss, Mark. "Estructuras de datos y algoritmos". Wilmington, Addison-Wesley, 1995.
[15] Joyanes, L. "Fundamentos de programación. Algoritmos y estructuras de datos". Segunda edición. McGraw-Hill, 1996.
[16] Jorge Castro, Felipe Cucker, Xavier Messeguer, Albert Rubio, Lluis Solano y Borja Valles. "Curso de Programación". McGraw-Hill, 1994.
[17] Peña, R. "Diseño de Programas. Formalismo y Abstracción. 2ª edición." Prentice Hall Hispanoamericana, S.A., 1997.
[18] Balcázar J.L. "Programación Metódica". Ed. McGraw-HilI, 1993.
[19] Franch, X., "Estructuras de datos. Especificación, diseño e implementación". Ediciones UPC. 1994.

Web de Universidades

[20] A Coruña http://www.udc.es

[21] Alcalá http://www.uah.es
[22] Alfonso X El Sabio http://www.uax.es
[23] Alicante http://www.ua.es
[24] Almería http://www.ual.es/
[25] Antonio de Nebrija http://www.unnet.es
[26] Autónoma de Barcelona http://www.uab.es
[27] Autónoma de Madrid http://www.uam.es
[28] Barcelona http://www.ub.es
[29] Burgos http://www.ubu.es
[30] Cádiz http://www.uca.es
[31] Camilo José Cela http://www.ucjc.edu
[32] Cardenal Herrera-Ceu http://www.ceu.es
[33] Carlos III de Madrid http://www.uc3m.es
[34] Castilla-La Mancha http://www.uclm.es
[35] Católica de Ávila http://www.ucavila.es
[36] Católica San Antonio http://www.ucam.es
[37] Complutense de Madrid http://www.ucm.es
[38] Córdoba http://www.uco.es
[39] Deusto http://www.deusto.es
[40] Europea de Madrid http://www.uem.es
[41] Extremadura http://www.unex.es
[42] Girona http://www.udg.es
[43] Granada http://www.ugr.es
[44] Huelva http://www.uhu.es
[45] Illes Balears http://www.uib.es
[46] Jaén http://www.ujaen.es
[47] Jaume I de Castellón http://www.uji.es
[48] La Laguna http://www.ull.es
[49] Las Palmas de Gran Canaria http://www.ulpgc.es/index0.html
[50] León http://www.unileon.es
[51] Lleida http://www.udl.es
[52] Málaga http://www.uma.es
[53] Miguel Hernández http://www.umh.es
[54] Mondragón Unibertsitatea http://www.muni.es
[55] Murcia http://www.um.es
[56] Nacional de Educación A Distancia http://www.uned.es
[57] Oberta de Catalunya http://www.uoc.es
[58] Oviedo http://www.uniovi.es
[59] Politécnica de Catalunya http://www.upc.es
[60] Politécnica de Madrid http://www.upm.es
[61] Politécnica de Valencia http://www.upv.es
[62] Pompeu Fabra http://www.upf.es
[63] Pontificia Comillas http://www.upco.es
[64] Pontificia de Salamanca http://www.upsa.es
[65] Pública de Navarra http://www.unavarra.es
[66] Ramon Llull http://www.url.es
[67] Rey Juan Carlos http://www.urjc.es
[68] Rovira I Virgili http://www.urv.es
[69] Salamanca http://www.usal.es
[70] San Pablo C.E.U. http://www.ceu.es
[71] Sevilla http://www.us.es
[72] Valencia Estudi General http://www.uv.es
[73] Valladolid http://www.uva.es
[74] Vic http://www.uvic.es
[75] Vigo http://www.uvigo.es
[76] Zaragoza http://www.unizar.es

La Universidad Española: Contenidos sobre Programación en los primeros cursos de las Titulaciones en Informática

Pedro J. Clemente y Pedro L. Pérez
Dpto. de Informática. Universidad de Extremadura.
Quercus Software Engineering Group. http://quercusseg.unex.es
e_mail: {jclemente, plperez}@unex.es

Resumen
Los planes de estudio establecen los conocimientos y habilidades que los alumnos deben desarrollar. En muchos casos, estos planes de estudio no están suficientemente adecuados a las necesidades actuales, y deben actualizarse y afrontar los cambios sufridos, no sólo a cambios tecnológicos, sino también cambios sociales y docentes. Sin embargo, no cabe duda que mantener actualizada la planificación de las asignaturas de informática resulta toda una tarea de investigación. El profesor debe documentarse sobre las últimas currícula internacionales, así como mantener una visión global de los contenidos que se imparten en otras universidades, nueva bibliografía, etc. Con el fin de ayudar en esta revisión de contenidos, en este artículo se presenta un estudio sobre los contenidos de programación que se imparten en la universidad española. Para ello se tiene en cuenta el número de créditos aplicados, los paradigmas y lenguajes de programación utilizados, así como los contenidos impartidos. Por último, se presenta una propuesta para la distribución de estos contenidos durante los primeros cursos de las Titulaciones de Informática.

1. Introducción

Cuando un profesor se plantea organizar los contenidos de una determinada materia, normalmente necesitaría más créditos de los que dispone. Por tanto, se debe ajustar los contenidos a los créditos de que dispone, y por tanto, se debe buscar un equilibrio entre contenidos y profundidad con que estos se imparten. En la toma de decisiones se deben tener en cuenta las recomendaciones curriculares más representativas (IS'97[1], CC91[2], CC2001[3], MSIS'2000[9], IFIP'94[11], ICF'2000[12], etc.), las directrices que ofrece el MEC y aproximaciones curriculares nacionales, así como la situación en la universidad española de las asignaturas relacionadas con la materia.

La recopilación de información sobre los contenidos que se imparten en otras universidades y que pueden servir como referencia la hora de realizar una nueva planificación o plan de estudio resulta una labor tediosa. Sin embargo, es necesaria para establecer los contenidos que otros compañeros imparten en sus universidades.

En este trabajo se realiza un estudio sobre los contenidos en programación de las universidades españolas en los primeros cursos de las Titulaciones de Informática (sección 2). Posteriormente se presenta una propuesta que establece los contenidos a impartir en los primeros cursos de estas Titulaciones (sección 3). Esta propuesta incluye temporización, paradigmas y lenguajes de programación a utilizar para su desarrollo. Dicha propuesta está basada en el estudio de los planes de estudio y contenidos impartidos en las universidades españolas, las aproximaciones realizadas por diferentes currículas internacionales [1][9] [11][12], centrándonos en las recomendaciones para los primeros cursos realizada por *Computing Currícula 2001* de ACM/IEEE [3], experiencia propia y de otros profesores en la materia.

2. La situación actual en la Universidad Española

En esta sección se presenta un estudio sobre los contenidos que se imparten sobre programación en la universidad española. Para la elaboración de este resumen se han tenido en cuenta las

asignaturas tanto troncales como obligatorias destinadas a impartir las directrices *Metodología y Tecnología de la Programación* y *Estructura de Datos y de la Información,* (esta última desde el punto de vista de *Estructuras de Datos y Algoritmos,* es decir, sin entrar en el tópico *Bases de Datos*)[10], en las universidades que imparten la titulación de Ingeniería Informática.

En primer lugar describiremos cuál es la distribución de créditos para los contenidos de primer y segundo curso, así como su distribución en asignaturas. Posteriormente, se analizará cuál es el primer paradigma de programación utilizado en las Universidades españolas, así como los lenguajes de programación utilizados. Por último, se presentan los contenidos impartidos tanto en primer y como en segundo curso[1].

2.1. Los Créditos destinados a programación

La primera situación que nos encontramos es la existencia de aproximadamente el mismo número de créditos en primer y segundo curso relacionados con programación. Concretamente, de media se ofertan 16,6 créditos para asignaturas de primer curso, y una media de 15 créditos para asignaturas de segundo curso. En ambos casos, suele existir una distribución pareja: dos asignaturas cuatrimestrales en primer curso y dos asignaturas cuatrimestrales en segundo curso.

Sin embargo, existen picos sobre estos datos, como los proporcionados por la Universidad de Oviedo que oferta 15 créditos en primer curso y 24 créditos en segundo curso, con lo cual su oferta global de contenidos incluye contenidos de forma más extensa (Programación dirigida por eventos o programación en entorno Windows en primer curso, o por ejemplo, estudio detallado de metodología orientado a objetos y programación funcional en segundo curso).

En cuanto a la distribución de contenidos teóricos y prácticos: En primer curso, de los 16,6 créditos disponibles 9,2 corresponden a créditos teóricos y 7,4 a créditos prácticos. En cuanto a segundo curso, de los 15 créditos se distribuyen en 9 teóricos y 6 prácticos.

Desde nuestro punto de vista, el número de créditos es insuficiente para proveer al alumno de los contenidos requeridos actualmente[3]. Por ello, se debería incrementar el número de créditos medio para hacer frente a los nuevos contenidos que demanda la sociedad, y que no estaban recogidos en los reales decretos que regulan las Titulaciones en Informática[10], como pueden ser los contenidos sobre Programación Orientada a Objetos.

2.2. El paradigma de programación

Desde el punto de vista del paradigma de programación utilizado en las diferentes asignaturas podemos apreciar en la Figura 1 la distribución correspondiente a las asignaturas de primer curso y en la Figura 2 segundo curso.

Vamos a centrar nuestra atención en los paradigmas impartidos en primer curso, y en este sentido, es el paradigma imperativo el predominante (66%), aunque también se aprecia que el paradigma orientado a objetos tiene una amplia aceptación (27%). Es importante recordar que como media en primer curso las universidades ofertan dos asignaturas, y es bastante común que en la primera de ellas se imparta el paradigma imperativo y en la segunda una aproximación orientada a objetos. Pocas universidades imparten como primer paradigma el orientado a objetos, tal y como ocurre, por ejemplo, en la Universidad Carlos III de Madrid.

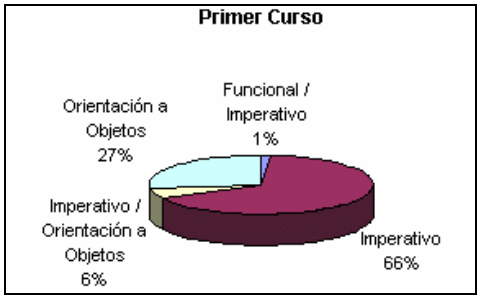


Figura 1. Distribución de los paradigmas de programación en Primer Curso

En el primer curso, el paradigma imperativo se impone en el primer cuatrimestre. En el segundo cuatrimestre cada vez más

[1] Los datos que forman este estudio han sido recopilados a finales de 2002.

universidades adoptan una aproximación orientada a objetos.

Analizando la situación para el segundo curso se observa un avance importante del paradigma orientado a objetos (en el 60% de las asignaturas). Casi el total de las universidades imparte este paradigma en alguna de las asignaturas de segundo curso.

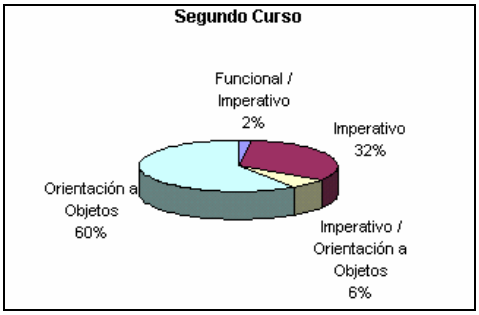


Figura 2. Distribución de los paradigmas de programación en Segundo Curso

Como conclusión destaca que en la Universidad Española el primer paradigma que se introduce es el paradigma imperativo. Posteriormente, en muchos casos en el segundo cuatrimestre, se introduce el paradigma orientado a objetos. En segundo curso el paradigma predominante es el orientado a objetos, siguiendo la estructura que se propone desde el enfoque *Primero Imperativo* en CC2001[3], donde los alumnos reciben formación sobre el paradigma imperativo inicialmente para pasar a estudiar el paradigma orientado a objetos posteriormente.

2.3. El Lenguaje de Programación

Un elemento de controversia en la comunidad docente en *Metodología de la Programación* y *Estructuras de Datos y Algoritmos* es el lenguaje de programación utilizado. Por ello, hemos analizado los lenguajes de programación utilizados tanto en primer, como en segundo curso.

En la Figura 3 podemos observar la distribución de los lenguajes de programación utilizados en primer curso. Podemos observar que aparecen destacados los lenguajes de programación *híbridos* (C++, Pascal). Esta situación se debe a las decisiones en cuanto al paradigma de programación utilizado. Hemos comentado en la sección anterior que se impone el paradigma imperativo en el primer cuatrimestre de primer curso, y que comienza a introducirse el paradigma orientado a objetos en el segundo cuatrimestre. Por tanto, es habitual utilizar un lenguaje híbrido para el desarrollo de las asignaturas, ya que el alumno no necesita cambiar de lenguaje para pasar del paradigma imperativo al orientado a objetos[5]. ¿Por qué C++ es el lenguaje más utilizado? Entendemos que además de tratarse un lenguaje híbrido, que podemos utilizar para ilustrar el paradigma imperativo y orientado a objetos, se trata de uno de los lenguajes de programación más utilizados en la empresa y esta situación motiva también a los alumnos a aprenderlo.

Las universidades que apuestan por impartir como primer paradigma el paradigma orientado a objetos, por ejemplo, las Universidades de Almería y Carlos III de Madrid, imparten Java (un lenguaje orientado a objetos calificado como *puro*). Como se ha comentado, esta no es la situación habitual. Por tanto, como podemos ver en la Figura 3, el porcentaje de asignaturas que imparten Java en primer curso está motivado por el sesgo orientado a objetos que sufren las asignaturas en el segundo cuatrimestre.

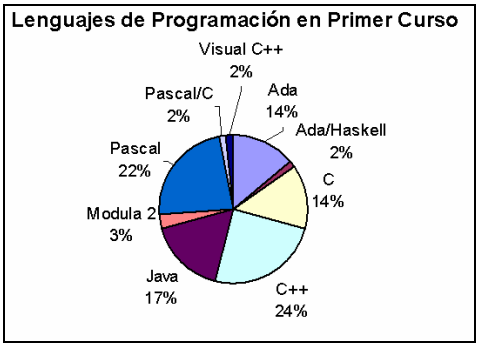


Figura 3. Distribución de los lenguajes de programación en Primer Curso

Los lenguajes de programación predominantes utilizados en segundo curso (ver Figura 4), donde se imparte el paradigma orientado a objetos, son C++(32%) y Java (28%). Podemos observar cómo C++ sigue manteniendo la hegemonía, seguido por Java, un

lenguaje creado desde sus orígenes orientado a objetos e implantado fuertemente en Internet, lo cual ofrece un gran atractivo al alumno.

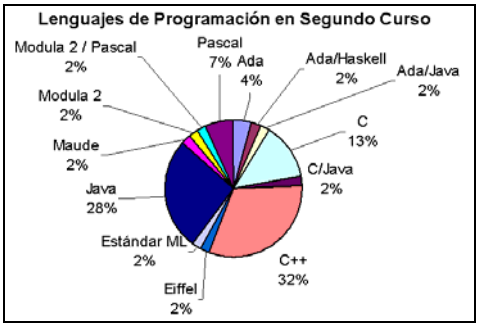


Figura 4. Distribución de los lenguajes de programación en Segundo Curso

2.4. Contenidos Primer Curso

En cuanto a los contenidos de primer curso, la primera consideración importante es que ambas directrices, Metodología y Tecnología de la Programación y Estructura de Datos y de la Información (esta última desde el punto de vista de Estructuras de Datos y Algoritmos)[10] y sus tópicos se imparten normalmente tanto en primer curso como en segundo curso, es decir, las asignaturas de primer curso imparten nociones de metodología de la programación y estructuras de datos, y las asignaturas de segundo curso profundizan y amplían los contenidos tanto de metodología de programación como de estructuras de datos y algoritmos.

Vamos a describir cuales con los contenidos que habitualmente se desarrollan, según la información obtenida sobre las asignaturas que se imparten en las universidades españolas. Los contenidos incluyen su correspondiente referencia en Computing Currícula 2001[3], lo cual muestra inicialmente la adaptación de contenidos a está currícula, considerada como la más completa.

Los principales temas que los alumnos encuentran en las asignaturas de primer curso relacionadas con las directrices MEC estudiadas son:

- *Conceptos Fundamentales:* Introducción a los conceptos fundamentales de las metodologías de programación, breve historia de los lenguajes de programación (*SP1 en CC2001*). Recorrido por los diferentes paradigmas de programación: procedurales, orientados a objetos, funcionales, declarativos. Proceso de compilación, interpretación. Diferencias coste/portabilidad. *(PL1 de CC2001).*
- *Algoritmos y Resolución básica de problemas:* Estrategias de resolución de problemas, algoritmos y su representación. Metodologías de diseño de algoritmos, diseño descendente. *(PF1 y PF2 de CC2001).*
- *Programación Estructurada:* **S**e imparten las diferentes estructuras de control de flujo: secuenciales, de selección e iteración *(PF1 de CC2001).*
- *Tipos de Datos y Acciones Elementales:* Elementos básicos y tipos elementales: variables, tipos, expresiones, asignaciones, acciones elementales. *(PL4 de CC2001).*
- *Programación Modular:* Mecanismos de abstracción básicos: Funciones, procedimientos, módulos de los lenguajes de programación. Recursividad. Paso de parámetros tanto por valor como por referencia. Variables y ámbito de las mismas *(PL5 de CC2001).*
- *Análisis de algoritmos:* El estudio de la eficiencia de los algoritmos se estudia inicialmente en las asignaturas de primero. Por ejemplo, en la Universidad Autónoma de Barcelona o en la Universidad de Extremadura se estudia sólo complejidad en algoritmos iterativos. Se pospone para segundo curso el estudio profundo del análisis de algoritmos. *(AL1 de CC2001)***.**
- *Tipos de Datos definidos por el usuario: Vectores, Matrices, Registros.* En la totalidad de los planes de estudio se imparten estos conceptos y estructuras fundamentales. *(PF3 de CC2001)*
- *Algoritmos de ordenación y búsqueda:* Se trata de algoritmos de manipulación de estructuras de datos lineales (búsqueda secuencial y binaria) *(AL3 de CC2001)*
- *Tipos abstractos de datos.* En la gran parte de los planes de estudio consultados se

imparten los Tipos Abstractos de Datos fundamentales: TAD Pila, TAD Cola, TAD Lista, e incluso TAD Árbol, aunque no es lo habitual. *(AL2 en CC1991)*

- *Punteros, gestión dinámica de memoria, estructuras dinámicas, listas enlazadas.* Conceptos ampliamente extendidos en los planes de estudio y en los contenidos que se imparten en primer curso. *(PF3 en CC2001)*
- *Introducción a la Programación Orientada a Objetos:* La introducción del paradigma orientado a objetos es significativa. Sin embargo, sólo en algunas universidades se introduce los conceptos de herencia y polimorfismo, tal y como ocurre en la Universidad de Castilla la Mancha donde se ofrece una asignatura específica de Programación Orientada a Objetos en el segundo cuatrimestre, donde se imparte el lenguaje Java. Lo mismo ocurre en las aproximaciones de la Universidad Carlos III y en la Universidad de Almería donde el primer paradigma que se estudia es el orientado a objetos. Sin embargo, es bastante frecuente encontrarnos aproximaciones *centrada en objetos*[4], es decir, donde se realiza una aproximación a la programación orientada a objetos, describiendo, clases, objetos, métodos, atributos, características de encapsulación, pero dejando los conceptos de herencia y polimorfismo para profundizar en las asignaturas de segundo curso. *(PL6 de CC2001)*
- *Ficheros.* Introducción al almacenamiento en memoria secundaria. Se tratan los conceptos introductorios, justificación, acciones básicas sobre ficheros, clasificación (secuenciales & acceso directo), algoritmos de ordenación (mergesort y quicksort) y fusión. *(PF3 en CC2001)*
- *El proceso de desarrollo software.* Desde las asignaturas de primer curso se introduce la necesidad de una metodología para el desarrollo software. Este proceso se introduce en función del paradigma utilizado (Ciclo de vida del software estructurado, orientado a objetos, etc.). *(SE1 en CC2001)*
- *Depuración, Prueba de programas y Documentación. (SE6 en CC2001)*
- *Especificación Formal.* Sólo en algunas universidades. Aunque se introduce la necesidad de especificar los TADs, sin embargo, se utiliza una notación no formal. *(DS3 en CC2001)*
- *Algoritmos numéricos:* El estudio de algoritmos numéricos se estudia en algunas universidades, como la universidad de Vigo, donde el número de créditos para primer curso es superior a la media (Vigo 27 Créditos para primer curso) *(AL3 en CC2001).*

2.5. Contenidos Segundo Curso

En este apartado vamos a destacar los principales temas que los alumnos encuentran en las asignaturas dedicadas a programación de segundo curso:

- *Análisis de Algoritmos:* Se trata de uno de los aspectos fundamentales para segundo curso en las universidades españolas. Se estudia la notación asintótica, cálculo de la eficiencia de algoritmos tanto en iterativos y recursivos. Este estudio se plasma con el cálculo de la eficiencia de diferentes algoritmos de búsqueda y ordenación. *(AL1 en CC2001)*
- *Especificación y Verificación formal:* Se imparte en profundidad en varias universidades. Por ejemplo en las asignaturas *Programación metódica* en la Politécnica de Cataluña, *Metodología de la programación* en las Universidades del País Vasco y Zaragoza, o *Programación II* en la Universidad Nacional de Educación a Distancia. *(DS3 en CC2001)*
- *Tipos Abstractos de Datos:* La especificación de tipos abstractos de datos, bien con especificación formal e informal es un contenido típico en la universidad española, posiblemente influenciada por CC1991[2]. Por ejemplo, encontramos Tipos Abstractos de Datos en la asignatura Algoritmos y Estructuras de Datos en la Universidad de Murcia, o incluso como asignatura específica, véase la asignatura Tipos Abstractos de Datos en la Universidad de Málaga. *(AL2 en CC1991)*

- *Estructuras de Datos:* Estructuras de datos lineales (Conjuntos, Pilas, Colas y Listas)*(PF3 en CC2001)*, Árboles y Grafos *(DS5 en CC2001)(PF3 en CC2001)*. Por ejemplo, en la Universidad Autónoma de Barcelona se imparte una asignatura completa para el estudio de Grafos (Grafs i complexitat). Para estas estructuras se estudia su especificación, representación, diferentes implementaciones, algoritmos de ordenación y búsqueda (pila, colas y listas), algoritmos de coste mínimo (grafos), árboles de expansión mínima (grafos). *(AL3 de CC2001)*
- *Tablas:* Estudio de Tablas, su especificación e implementación. Estudio de Tablas Hash y los técnicas de resolución de colisiones *(PF3 en CC2001)*
- *Técnicas de Diseño de Algoritmos:* El estudio de las principales técnicas de diseño de algoritmos se encuentra en la mayoría de las universidades. Los tópicos incluidos en este estudio son: Programación Dinámica, Algoritmos Voraces, Algoritmos Probabilísticas, Algoritmos heurísticos, así como Ramificación y Poda. En algunas universidades disponen de asignaturas específicas donde este tema es el núcleo de la misma. Por ejemplo, la asignatura *Teoría de Algoritmos* de la Universidad de Granada, la asignatura *Análisis y Diseño de Algoritmos* en la Universidad de Sevilla o *Metodología de Programación* en la Universidad de Castilla la Mancha. *(AL2 en CC2001)*
- *Programación Orientada a Objetos:* En segundo curso se imponen el paradigma orientado a objetos, tal y como hemos visto en las secciones anteriores. Este paradigma se estudia en profundidad en las asignaturas de segundo curso, completando en su caso los conceptos de orientación a objetos que los alumnos han recibido en primer curso *(PL6 en CC2001)*.
- *Metodología de Programación Orientada a Objetos:* En algunos casos, se ofertan asignaturas como *Laboratorio de Tecnología Orientada a Objetos* de la Universidad de Málaga o *Metodología de Programación II* de la Universidad de Oviedo donde se incluyen metodologías de análisis y diseño orientado a objetos, técnicas de depuración y documentación, utilización de librerías y API y programación dirigida por eventos *(SE1, PL6, SE2 en CC2001)*.

En esta sección hemos realizado un estudio sobre los contenidos que comúnmente se imparten en las asignaturas de programación dentro de la Universidad Española. En la siguiente sección se presenta una distribución de estos contenidos para 1er y 2º curso.

3. Propuesta de distribución de contenidos en 1er y 2º curso

A la hora de realizar la planificación de los planes de estudio es importante establecer de forma adecuada requisitos y correquisitos. Con este fin, las asignaturas donde se imparten los contenidos sobre programación deben formar bloques muy bien diferenciados, donde se determine adecuadamente sus requisitos, y los conocimientos que se proveen tanto desde el punto de vista teórico como práctico. En este sentido deben evitarse situaciones como la siguiente: planificar asignaturas que comparten temario durante un mismo cuatrimestre, donde una de ellas imparte los temas teóricos que deben ser aplicados en las clases prácticas de una segunda asignatura. Esta situación, aparece en algunos los planes de estudio de la universidad española, donde se mezclan asignaturas anuales con cuatrimestrales, asignaturas eminentemente teóricas con asignaturas totalmente prácticas, lo cual dificulta una clara división de contenidos y plantea problemas de planificación.

Desde nuestro punto de vista, una distribución de asignaturas cuatrimestral, donde sea factible realizar una clasificación de requisitos y correquisitos, unido a una relación teoría/práctica equilibrada en las asignaturas permitirá a los alumnos asimilar paulatinamente los contenidos sobre la materia. En esta relación se deben evitar los temarios de contenidos excesivos donde únicamente se ofrezcan *pinceladas* y no se abordan los temas en profundidad.

La propuesta que a continuación se presenta está desarrollada teniendo en cuenta las Directrices del MECD, currículas

internacionales y así como recomendaciones nacionales en la materia[5][6][7][8], análisis de los planes de estudio y estudio de la materia en otras universidades (ver sección 2), experiencia propia de la materia y de otros profesores.

Para conseguir los objetivos planteados se presentan las siguientes asignaturas troncales: *Programación I y Estructuras de Datos I* para primer curso, y *Programación II y Estructuras de Datos II* para segundo curso. En segundo curso se añade una obligatoria *Software Orientado a Objetos*. Esta nueva obligatoria responde a la necesidad de ajustar los contenidos y créditos disponibles para el bloque de programación, donde la POO requiere un espacio para su estudio específico.

La distribución de contenidos y créditos puede apreciarse en Tabla 1. Debemos notar que cada asignatura dispone de un conjunto de créditos prácticos mediante los cuales el alumno debe familiarizarse y asimilar los contenidos impartidos en las clases teóricas.

Curso	*Cuat.*	*Asignatura*	*Créditos*
1	1	Programación I	7,5 (4,5T+3P)
1	2	Estructuras de Datos I	7,5(4,5T+3P)
2	1	Programación II	7,5(4,5T+3P)
2	1	Software Orientado a Objetos	6(4,5T+1,5P)
2	2	Estructuras de Datos II	7,5(4,5T+3P)

Tabla 1. Distribución de asignaturas en 1er y 2º Curso

En esta propuesta se introduce al alumno en programación mediante el paradigma imperativo, para introducir posteriormente los conceptos iniciales de programación orientada a objetos, que será el paradigma que se utilice a partir del segundo cuatrimestre de primer curso. En cuanto al lenguaje de programación utilizado, se opta por C++ por los motivos descritos en la sección 2, y que llevan a este lenguaje a ser el más utilizado en la universidad española. Sin embargo, en segundo curso, se propone un segundo lenguaje de programación: Java, que nos permite una aproximación más completa de la POO, al menos, en cuanto a características específicas de POO como pueden ser las interfaces. A continuación se detalla brevemente los contenidos a impartir en cada una de las asignaturas:

3.1. Programación I

Se trata de la primera asignatura sobre programación que estudiarán los futuros titulados. Para ello se realiza una aproximación a la programación imperativa, eficiencia de algoritmos, estructuras fundamentales (registros y vectores), para terminar con una introducción a la programación orientada a objetos. Podemos establecer tres módulos bien diferenciados:

Un primer módulo de *introducción a la programación imperativa.* Este módulo pretende ofrecer una visión global sobre la programación imperativa al alumno. Los contenidos de este módulo corresponden a: Conceptos Fundamentales, Algoritmos y Resolución básica de Problemas, Programación Estructurada, Tipos de Datos y Acciones Elementales, Programación Modular, Depuración y Prueba de Programas, Documentación. Conceptos de eficiencia de los algoritmos, donde se estudia el cálculo de la eficiencia en algoritmos iterativos.

Un segundo módulo donde se presenta el concepto de *Tipo Abstracto de Datos* como herramienta de abstracción. En este módulo se describe el tratamiento de *vectores* introduciendo los aspectos básicos de su manejo, así como los algoritmos típicos de ordenación y búsqueda, que pueden ser utilizados como banco de prueba de los conceptos sobre eficiencia.

Un tercer módulo donde se presenta la *programación orientada a objetos*. Donde los alumnos estudiarán los conceptos básicos y las principales características de la programación orientada a objetos, sin entrar en detalles sobre herencia y polimorfismo. Se debe incidir sobre la capacidad de abstracción del mundo real mediante objetos.

3.2. Estructuras de Datos I

Esta asignatura de segundo cuatrimestre permite introducir al alumno en las primeras estructuras

de datos, concretamente estructuras lineales, implementadas de forma estática y dinámica, así como la gestión de datos persistentes en ficheros.

Inicialmente debe realizarse un breve resumen sobre conceptos básicos de POO introducidos en la asignatura requisito (Programación I). A continuación, se estudiarían *Estructuras Lineales.* En este módulo se estudia la implementación de las estructuras lineales más usuales (Pilas, Colas, Listas, Conjuntos).

El estudio de la *gestión dinámica de memoria* (punteros, variables dinámicas, reserva y liberación de memoria) nos permitirá introducir al alumno las implementaciones dinámicas de las estructuras de datos lineales estudiadas. Por último es conveniente introducir al alumno en el almacenamiento persistente de datos (módulo sobre *ficheros*). Posiblemente este tema sea de los que más interés despierta en los alumnos, ya que abre todo un mundo de posibilidades para sus aplicaciones. Desde el punto de vista teórico, se deben explicar la necesidad de almacenamiento persistente de datos, los conceptos y operaciones básicas, clasificación de ficheros (secuencial y acceso directo) (texto y binario), algoritmos básicos de inserción y borrado, así como fusión y ordenación de ficheros.

3.3. Programación II

Esta asignatura tiene como objetivo introducir al alumno en el análisis de algoritmos, las técnicas más usuales de diseño de algoritmos, así como introducir la especificación y verificación formal (esta asignatura tiene como requisito Programación I).

En un primer módulo sobre *análisis de algoritmos* se presentan los conceptos necesarios para analizar los algoritmos desde el punto de vista de la eficiencia. Dichos conceptos son los que posteriormente deben ser utilizados para razonar sobre la complejidad computacional de los algoritmos construidos. En un segundo módulo presentan las principales *técnicas de diseño de algoritmos* (Divide y vencerás, Algoritmos Voraces, Vuelta atrás o backtracking, Ramificación y poda, Programación dinámica). Se debe razonar sobre su aplicabilidad y adecuación a la hora de resolver un problema. Las diferentes estrategias pueden presentarse con su esquema general y un conjunto de ejemplos de aplicación de cada una de las técnicas. Por último, un módulo sobre *especificación algebraica y verificación formal* se ofrece una visión de la programación desde el punto de vista diferente, que permita a los alumnos familiarizarse con la verificación y derivación formal de programas.

3.4. Software Orientado a Objetos

En esta asignatura se extienden los conceptos sobre programación orientada a objetos presentando las características de herencia y polimorfismo, y los conceptos asociados tipos de herencia, ligadura tardía, métodos virtuales, etc. En esta asignatura se debe presentar una introducción a alguna metodología para el desarrollo de aplicaciones software orientado a objetos (diagramas de clases, diagramas de interacción, etc.)[2].

En un curso de estas características es importante que el alumno no adquiera la visión de OO ofrecida por un lenguaje en particular, sino que comprendan los conceptos que subyacen a la OO de una forma independiente del lenguaje[7].

Como complemento a las clases teóricas se introduce el lenguaje de programación Java. Que será el lenguaje de programación utilizado en las clases prácticas, si embargo, y como se ha comentado se recomienda la exposición de los conceptos sobre orientación a objetos mediante ejemplos en varios lenguajes, y no centrar los conceptos en la implementación concreta realizada por un lenguaje de programación.

3.5. Estructuras de Datos II

En este módulo se parte de los conceptos estudiados sobre eficiencia de algoritmos y sobre especificación y verificación formal impartidos en Programación II, así como los conceptos sobre orientación a objetos impartidos en Software Orientado a Objetos, y se presentan

[2] Sería recomendable que la metodología seleccionada se desarrollarse completamente en una asignatura de tercer curso

las estructuras de datos básicas: Pilas, Colas, Listas, Conjuntos, Tablas, Árboles y Grafos. La exposición de cada estructura de datos puede comenzar con una introducción sobre la estructura de datos de estudio y su especificación algebraica, a continuación, presentar diversas alternativas de implementación, analizando la eficiencia de cada uno de los algoritmos de manipulación implementados Especial interés requieren los temas dedicados a las estructuras arbóreas y grafos, dada la utilidad de las mismas y el amplio conjunto de algoritmos susceptibles de ser estudiados.

4. Conclusión

Se ha presentado un estudio sobre los contenidos relacionados con programación que se imparten en la universidad española. De este estudio destaca que en la universidad española el paradigma de programación que se presenta al alumno en primer curso es el imperativo, sin embargo, se observa una mayor implantación del paradigma orientado a objetos en el segundo cuatrimestre de primer curso y así como en el segundo curso. Dada estas características de cambio de paradigma de programación, los lenguajes de programación híbridos son los más utilizados. Por último, cabe destacar en cuanto a los contenidos impartidos (en primer y segundo curso) la coincidencia con la propuesta *Primero Imperativo* de Computing Currícula 2001[3] para estos cursos iniciales.

Por otro lado, se ha presentado una propuesta de plan de estudios para los contenidos sobre programación en los primeros cursos de las Titulaciones en Informática. Establecer un equilibrio entre los contenidos teóricos y prácticos de las asignaturas, dando la oportunidad a los alumnos de afrontar dichos contenidos bajo una constante componente práctica. Esta componente práctica ayuda a los alumnos en el desarrollo de las habilidades, aptitudes y conocimientos requeridos.

Referencias

[1] ACM-AIS-AITP Joint Task Force. *IS'97 Model Curriculum and Guidelines for Undergraduate Degree Programs in Information Systems, 1997*.

[2] ACM/IEEE. *Computing Curricula 1991*. Report of the ACM/IEEE-CS Joint Curriculum Task Force. IEEE Computer Society Press, 1991.

[3] ACM/IEEE. *Computing Curricula 2001*. Computer Science. Final Report December, 15. The Joint Task Force on Computing Curricula, 2001.

[4] Arnow D. y Weiss G. *Introduction to Programming using Java.* 2ª edición, Addison-Wesley, 2000.

[5] Fernández Muñoz L., Peña R., Nava F. y Velásquez Iturbide A. *Análisis de las propuestas de la enseñanza en la programación orientada a objetos en los primeros cursos*. VIII JENUI, Cáceres, 2002.

[6] García Molina, J.J. *¿Es conveniente la Orientación a Objetos en un primer curso de programación?* Novatica, noviembre-diciembre 2001.

[7] García Molina, J.J.; Menánguez Tortosa, Marcos; Moros Valle, Begoña. *Una propuesta para organizar la Programación Orientada a Objetos*. VIII JENUI, Cáceres, 2002.

[8] Gómez Albarrán, Mercedes. *Metodología basada en descomposición funcional y orientación a objetos en la introducción a la programación*. VIII JENUI, Cáceres, 2002.

[9] *Model Curriculum and Guidelines for Graduate Degree Programs in Information Systems* (MSIS'2000). Association for Computing Machinery (ACM) y Association for Information Systems (AIS). 2000.

[10] Reales Decretos 1459/1990, 1460/1990 y 1461/1990 (B.O.E. de 20 de noviembre de 1990). Establecen la creación de los títulos universitarios oficiales de II, ITIS e ITIG.

[11] Unesco-IFIP. *A modular Curriculum in Computer Science*. The United Nations Educational Scientific and Cultural Organization, 1994.

[12] UNESCO-IFIP (2000). ICF-2000. *Informatics Curriculum Framework 2000 for Higher Education*. International Federation for Information Processing. UNESCO, París.

Los estudios universitarios en la sociedad de la información y el conocimiento: una propuesta de verificación de cobertura de contenidos curriculares "*ad-hoc*" mínimos y metodología docente asociada

Ferran Virgos Bel, Antoni Pérez Poch
Dept. de Lenguajes y Sistemas Informáticos
Universitat Politècnica de Catalunya (UPC)
EUETIB Urgell 187
08036 Barcelona
e-mail:
Ferran.Virgos@upc.es , Antoni.Perez-Poch@upc.es

Resumen

En este trabajo se recoge la realidad de que muchos planes de estudios de cualquier titulación se diseñan con un fuerte componente de mimetismo. Si bien todo el mundo suele reconocer en público que los diseños curriculares deberían mantener un equilibrio "base teórica" vs. "componente práctico" basado en la "realidad social-industrial-empresarial", lo cierto es que a las empresas les sigue interesando que les resuelvan "su problema", mientras a los miembros del mundo académico nos encanta continuar explicando "lo que sabemos". Si este planteamiento lo mantenemos en la nueva realidad de la sociedad de la información y el conocimiento, los resultados pueden conducir a un notable alejamiento del entorno actual y, sobre todo, del futuro. Nuestros estudiantes hallarían dificultades en su integración profesional, y sus empresas no alcanzarían el nivel de competitividad deseado.

Y en cambio, resolver el problema no es difícil: basta con un poco de examen de conciencia, dolor de los pecados y propósito de enmienda de los responsables correspondientes. O sea: admitir que no sólo es importante lo que sabemos cada uno, ser honestos y aceptar que la estructura del diseño curricular debe actualizarse y decidirnos a actuar en consecuencia. Habrá que renunciar a alguna de las materias históricas y eso será doloroso. Las características y dinamismo de las nuevas materias aconsejarán, también, cambios en la metodología docente, pero ambos redundarán en una notable mejoría de la formación de los alumnos, sus perspectivas profesionales y la competitividad de sus empresas.

Jenui 2003 puede ser un buen momento para el nuevo viaje. De hecho, ya empezamos a ir justos de tiempo si no queremos perder el tren de las XXI.....

Desearíamos apresurarnos a aclarar, antes de proseguir, dos aspectos que me sugieren los revisores: el primero, que el contexto del trabajo no se sitúa en ninguna titulación y pretende crear un marco de uso universal, aunque, lógicamente, los huecos en algunas titulaciones puedan resultar más chocantes; el segundo, que el énfasis se pone en el mismo marco más que en la experiencia docente, donde sólo se hacen apuntes, y que sería objeto, en su caso, de otro trabajo.

1. Situando el tema

Todo el mundo admite que nuestro entorno social, empresarial y, en consecuencia, laboral, se ha visto modificado por la nueva realidad de la sociedad de la información y el conocimiento. Si buscamos el fundamento, digamos tecnológico,

del cambio, podemos hablar de tecnologías de la información (TI). Hace algún tiempo se hablaba de nuevas tecnologías de la información (NTI) pero los historiadores ya saben que llamarle a algo "nuevo" supone crear una etiqueta con pocas posibilidades de persistencia. Otra descripción muy usada es la de Tecnologías de la información y comunicaciones (TIC, o ICT), pero se me antoja que, en la actualidad, sólo puede leerse como un innecesario deseo de enfatizar lo evidente. En efecto, ¿quién imagina actualmente las tecnologías de la información sin comunicaciones?. Y siendo así, ¿qué sentido puede tener explicitar una parte junto al todo?.

Preferiría el concepto TI/SI para recoger la necesaria relación Tecnología de la Información-Sistema de Información. Con posterioridad, no obstante, la redundancia de la "I" y la aceptación de que la Tecnología es relevante para el sistema, puesto que hay una bidireccionalidad en el diseño (dos caras integradas de la misma moneda), llevó a proponer el término TSI (Tecnologías y Sistemas de Información) que ya aparece en [3].

Más recientemente, la diferenciación entre información y conocimiento llevó a algunos a realizar propuestas que incorporaran este término, tales como "Gestión del conocimiento" (*Knowledge Management*). Este es un "nuevo" nivel muy claro en lo fundamental (la relación datos-información-conocimiento es, como concepto, una pregunta de "examen" en cualquier curso del tema), sin duda necesario, pero todavía poco asentado y que, en cualquier caso, pensamos se basa en el anterior (TSI).

Dicho de otro modo, si nos preguntamos por la sociedad de la información y el conocimiento, veremos que lo que subyace es la aplicación de las tecnologías y sistemas de información a nuestras actividades cotidianas (términos como el propio *Knowledge Management* (KM), *Supply Chain Management* (SCM), *Customer-Relationship Management* (CRM), etc., no dejan de ser "casos" específicos. Incluso otros que "arrasan" en la bibliografía como *e-Commerce* o *e-Business* no añaden nada nuevo (excepto el marketing), aunque debemos aceptar que insinúan una orientación del planteamiento. En consecuencia, para este trabajo (y su proyección), siempre que podamos, utilizaremos el término TSI que es el que nos parece más apropiado.

2. Evolución de la visión de las TSI desde la empresa

Para continuar avanzando, propondremos un marco descriptivo de la evolución en la visión de las TSI por parte de las empresas, basado en una actualización del descrito en [1]. Así, podríamos diferenciar cinco fases: aplicación, MIS, dual, red y global (que se resume en la tabla 1):

a) *Aplicación*. No existe propiamente el concepto de Sistema de información. La informática se aplica exclusivamente a la mecanización de tareas repetitivas de nivel operacional. La incidencia de las TSI en los recursos humanos de la empresa es muy baja, excepto para el personal específico.

b) *MIS.* Integración de los subsistemas, sumarización e información por excepción. Es el concepto de Management Information System (MIS). Tecnológicamente se soporta en el acceso remoto y los Sistemas de Gestión de Base de Datos (SGBD) apareciendo la necesidad de planificar los sistemas empresariales desde una perspectiva general. Estamos en un nivel superior de Planificación y control operativo según la pirámide de Anthony [5]. También se inicia la importancia del papel del usuario. La informática (mejor dicho, el sistema informático o MIS) se convierte en el subsistema de control del sistema físico. Pero la incidencia en número de personas es todavía notablemente bajo. También se inician los sistemas organizacionales, si bien en forma limitada y para aplicaciones específicas.

c) *Dual.* Basado en el modelo propuesto por Le Moigne ([10], [11]), consistente en diferenciar un sistema transaccional (único para toda la organización) y un sistema decisional múltiple cuyos subsistemas estarían encargados de elaborar las informaciones que demandan los órganos de gestión (nivel de Planificación táctica y control de gestión en la pirámide de Anthony). El soporte tecnológico hay que buscarlo en los sistemas de compartición de tiempo sobre un *host* (*time sharing*), extendiéndose, más tarde, con los minis y los PCs. En estos casos, la arquitectura suele ser en estrella. Con esta incorporación, las TSI ascienden a niveles superiores pero siguen afectando a un reducido número de personas.

d) *Red.* Con perspectiva de futuro, en el momento de escribir el trabajo de Abril et al. [1], se referenciaba como *cibernético* y tomaba el modelo de Stafford Beer [6] cuyo paradigma sería nuestro sistema nervioso. Tecnológicamente coincide con la difusión de los PCs, las redes locales y la arquitectura cliente/servidor. Esta arquitectura permite que el puesto de trabajo se convierta en un nodo de la estructura empresarial, y a través de pasarelas, permite también, acceder al mundo exterior. Se produce un despegue en el número de usuarios, tanto en sentido vertical (pirámide jerárquica) como horizontal (generalización de uso) pero fundamentalmente limitado a las fronteras de las empresas (sistemas inter-organizacionales).

e) *Global-social.* La aplicación de las TSI trasciende no sólo las fronteras de las empresas y las relaciones entre ellas, sino que se extiende a las comunicaciones entre éstas y sus clientes finales llegando a dar cobertura habitual a las relaciones entre personas, a nivel individual. Se tiende a que cada individuo sea un nodo de la red. Desde el punto de vista del marketing, el viejo sueño de pasar de "1" mercado de "n" individuos a "n" mercados de "1" individuo es, ahora, posible. Muchos modelos empresariales entrarán en crisis y otros nuevos verán la luz. Las TSI tienen, en este momento, plena capacidad para reestructurar ciertos sectores, potenciar algunos más e incluso inventar otros nuevos.

fase	**Modelo**	**PERSONAS IMPLICADAS**	**NIVEL empresarial**	**Relación con la estrategia**	**Fronteras**
F1	APLICACIÓN	Pocas en la empresa	Operativo	(I) nula	Cerrado (sección)
F2	MIS	Algunas más en la empresa	Operativo + control de gestión	(I),(II) pasiva	La empresa
F3	DUAL	Se incrementa. Incluye soporte decisional	Idem + Decisional	(II) y (III) explícita	La empresa
F4	RED (limitada)	Generalización de uso en la empresa más algunas relaciones externas	Todos en la empresa	(III) patente	B2B Clientes y proveedores
F5	GLOBAL-SOCIAL	Prácticamente todas. Uso universal	Todos (incluyendo el personal individual)	(III) esencial	No hay ninguna

Tabla 1. Evolución de la visión de las TSI en la empresa

Desde el punto de vista empresarial, el principal elemento de cambio ha sido la relación con la estrategia corporativa. En este sentido, podríamos tipificar tres fases esenciales:

I. Desconexión (prácticamente no existe relación)
II. Planificación de TSI alineada con la estrategia (relación pasiva)
III. Contribución de TSI a la definición de la estrategia (relación activa)

Las implicaciones organizativas de las TSI ya se analizaban en un número monográfico de la revista Novática [17], mientras la visión estratégica de las TSI tiene una amplia bibliografía (véase, por ejemplo [2], [3, [4] y [12]). La cadena de valor propuesta por Porter (ver [13] y [14]), se convirtió, en su momento, en referencia obligada cuando se trata de "identificar" oportunidades de aplicación de las TSI.

El otro gran elemento de cambio ha sido la generalización de uso a nivel interno (en los dos ejes mencionados, tanto el vertical como el horizontal).

Pero, sobre todo, el gran *boom* se produce como consecuencia de la generalización externa (fase global-social), que abre infinitas posibilidades a nuevos modelos de negocio, con reestructuración total de algunos sectores y la práctica reinvención de otros (como ha quedado apuntado).

3. Nuestra relación personal/laboral con las TSI

En cuanto al nivel personal, la incidencia de la "e" (de electrónico), "d" (de digital) o (mejor aún!) "n" (de *network*), está presente en un día cualquiera de nuestra vida:

- *Anoche me acosté tarde. Estuve viendo, en mi PC, una película en DVD que me habían prestado en la biblioteca de mi universidad.*
- *No obstante, hoy me levanto pronto. Quiero adelantar un artículo que tengo que entregar el viernes, y estoy un buen rato trabajando con mi portátil, antes de desayunar. Avanzo bastante. Durante el desayuno, me conecto a INTERNET para leer las noticias del día.*
- *Después del desayuno, accedo al campus virtual de mi universidad para ver las consultas de mis alumnos. Hoy no tengo clase presencial y me voy a quedar trabajando en casa. Reviso como avanzan los trabajos de no presencialidad y sigo la evolución del forum de mi asignatura.*
- *Después, me dedico a revisar unas transparencias Powerpoint a las que estoy añadiendo voz. Yo creo que la formación presencial y no presencial tienen muchos puntos en común y quiero demostrarlo. Así continúo el resto de la mañana.*
- *Sobre las 13:30 doy por terminado ese trabajo y dedico unos minutos a seleccionar unas fotos digitales. Los reyes se acordaron de mi este año y he amortizado el regalo. Pero debo elegir las mejores, porque con la compra se incluía un vale para pasar 40 a copia en papel y caduca mañana.*
- *Recibo un mail de mi esposa con una cita para comer. Los SMS van bien pero con tarifa plana, INTERNET tiene coste marginal cero. Sería absurdo no usarlo.*
- *Durante la comida comento con mi esposa que trabaja en una inmobiliaria como podrían utilizar terminales de mano para hacer las valoraciones de pisos y evitarse las transcripciones y los errores. También como podrían ofrecer visitas virtuales a sus clientes y ahorrarse desplazamientos.*
- *Después de comer, compro dos libros en la red y entro en la web de RENFE para ver las comunicaciones con Cádiz. Creo que iré a JENUI 2003 en tren.*
- *A última hora de la tarde me han invitado a una escuela de turismo para dar una charla de las aplicaciones de las TSI en el sector. Les preocupa la desaparición de las agencias de viaje. Les animo y les digo que si entienden el concepto de "añadir valor al cliente" y usan ellos la red para ese fin, no deben temer nada.*

Por la noche, medito ¿Quién puede dudar de la incidencia de las TSI en, virtualmente, cualquier profesión, o lo que es lo mismo, en la vida o el trabajo de una inmensa mayoría de la población?. ¿Afecta sólo a los ingenieros en informática, o es esencial a ingenieros industriales, licenciados en empresariales, directores de hospitales o diplomados en turismo, por igual?. Si es así, ¿Porqué no se estudia?.

4. Diseño metódico de planes de estudios

Algunas referencias bibliográficas ([7], [19]) realizan propuestas concretas de contenidos curriculares para estudios específicos.

En forma alternativa/complementaria, en [20] se examinan algunos planes de estudio de diferentes universidades, detectando algunos núcleos claramente diferenciados en los planes correspondientes:

- Informática básica (conceptos y fundamentos de programación)
- Herramientas de productividad personal (singularmente entorno, tratamiento de textos y hoja de cálculo; más raramente gestión de datos)
- Utilizaciones de programas singulares (contabilidad, *workflow*, gestión de producción, gestión de proyectos), normalmente presentados como prácticas de alguna asignatura básica clásica.
- Visión de diseñador de sistemas informáticos.
- *Business-games*

La misma referencia [20], expone una metodología genérica para diseñar planes de estudios a partir de "objetivos" docentes. Pero el problema radica, en este caso, en saber definir los objetivos docentes. En efecto, ¿quién puede exigir a un jefe de estudios de una facultad de ciencias económicas que defienda que un economista debe ser capaz de diseñar un modelo con hoja de cálculo para evaluar "escenarios" alternativos y no es tan importante saber historia de la economía?, ¿quién puede asegurar que un jefe de estudios de una escuela de turismo tendrá claro que para un diplomado en turismo es tan o más importante conocer las posibilidades de la Web que ser un experto en geografía o en arte?, ¿quién le puede pedir a un profesor de medicina que intuya que para un traumatólogo que realiza intervenciones de artroscopia de rodilla puede ser más importante conocer la estructura de la señal de vídeo que la anatomía del aparato digestivo?.

La ventaja de [20] es que utiliza la metodología propuesta para deducir unas orientaciones curriculares en los estudios de empresariales. Pero nuestro objetivo en este trabajo sería definir un marco que pudiera aplicarse por taxonomía en cualquier tipo de estudios, incluyendo las ingenierías, desde luego, también, la de informática.

Para ello, será conveniente, buscar algunas fuentes autorizadas.

5. Un vistazo a la revisión de la recomendación curricular en Sistemas de Información (IS 2002)

La Association for Computing Machinery (ACM), desde 1972 viene proponiendo unas recomendaciones curriculares en relación a la formación en Sistemas de Información. En 1997 hubo un esfuerzo conjunto de AIS (*Association for Information Systems*), AITP *(Association of Information Technology Professionals*) e IFIP (*Internacional Federation for Information*). El resultado fue IS'97 Recientemente, coincidiendo con un congreso de AIS en Barcelona en enero del 2002, se presentó un documento con la puesta al día de estas recomendaciones, con la denominación IS'2002 [9].

En el documento citado se justifica la necesaria, actualización de la formación para poder responder a las nuevas necesidades de la profesión: el elemento básico detectado es la explosión de INTERNET. Es decir, alcanzar la situación tipificada en la fase 5 de nuestro modelo evolutivo de la tabla 1.

En el mismo trabajo se expone que el modelo curricular propuesto representa un "consenso entre los expertos y deberá ser usado como base para diseños curriculares específicos, pero no de modo prescriptivo sino flexible y adaptable", debiendo complementarse con las metodologías docentes más apropiadas en cada caso.

Para aceptar la capacidad de extensión de las recomendaciones anteriores a cualquier tipo de estudio, no hay más que fijarse en las características identificadoras de la disciplina que el documento engloba en cuatro áreas o ejes:

1. *Alcanzar una perspectiva genérica del mundo real y de las reglas del negocio*. Se trata, en consecuencia, de entender IS (nomenclatura del documento: nosotros diríamos TSI) como un facilitador (elemento activo) del éxito de la organización. Se trata de ver a las TSI como un elemento integrador de todos los niveles y funciones. Se trata, en definitiva, de ver a las

TSI como un elemento esencial en la definición de la estrategia corporativa.

2. *Alcanzar una fuerte capacidad analítica y habilidades creativas*. Se incluyen aspectos como saber encontrar soluciones a problemas, capacidad de usar conceptos de sistemas para entender y modelizar estos problemas, ser capaz de aplicar tanto los métodos tradicionales como las soluciones aportadas por los nuevos conceptos y capacidades. En definitiva, utilizando sus mismos términos, entender que "un sistema está formado por personas, procedimientos, hardware, software y datos".
3. *Tener elevados principios éticos así como capacidad de comunicación interpersonal y trabajo de equipo*.
4. *Ser capaces de diseñar e implantar soluciones basadas en las TSI que mejoren las capacidades de la organización*. En particular, se trataría de poseer la capacidad y las habilidades necesarias para entender y modelar los procesos y los datos propios de la organización, definiendo soluciones tecnológicas, gestionando los proyectos e integrando sistemas. En concreto, se trataría de "partir de una actitud de visión de las TSI como una herramienta al servicio. de los individuos y las organizaciones, en el logro de sus objetivos"

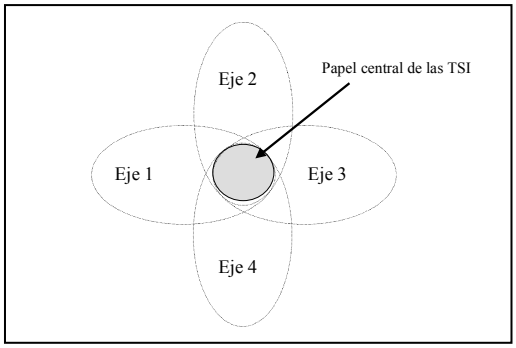


Fig 1: TSI como elemento facilitador de los procesos de negocio

El resumen de los cuatro ejes anteriores nos llevaría a identificar el papel central de las TSI como elemento impulsor-facilitador-catalizador de la mejora de los procesos de negocio en, virtualmente, cualquier tipo de organización y puesto de trabajo.

En el análisis comparativo con la recomendación anterior (IS'97), aparte otros ajustes menores, el principal déficit detectado era la inexistencia de un módulo orientado a Internet, Comercio electrónico y su incidencia empresarial. En consecuencia, La recomendación, añade un nuevo curso *IS 2002.2 Electronic Business Strategy, Architecture and Design* para resolver este déficit, reorganizando el resto de módulos.

La lista completa incluye un total de 10 módulos más el prerrequisito (P0):

- IS 2002.P0. *Personal Productivity with IS Technology.*
- IS 2002.1. *Fundamentals of Information Systems.*
- IS 2002.2. *E-Business Strategy, Architecture & Design.*
- IS 2002.3. *Information Systems Theory & Practice.*
- IS 2002.4. *IT Hardware & System Software.*
- IS 2002.6 *Networks & Telecommunications*
- IS 2002.5 *Programming, Data, File & Object Structures.*
- IS 2002.7 *Analysis & Logical Design.*
- IS 2002.8 *Physical Design & Implementation with DBMS.*

- IS 2002.9 *Physical Design & Implementation in Emerging Environments.*
- IS 2002.10 *Project Management & Practice*

De todos ellos, a nuestros efectos, nos interesan especialmente los 6 primeros (como aparecen en la relación anterior)**.**

6. Identificación de materias esenciales a modo de "check-list"

Combinando la metodología descrita en [20], así como sus conclusiones preliminares, con otros trabajos (como [7] y [19] citados previamente) y a la vista de las recomendaciones de IS 2002 del epígrafe anterior, hemos identificado cuatro orientaciones esenciales en cualquier formación:

- *A nivel individual*, un módulo orientado hacia las *herramientas de productividad, información y comunicación personal*. Naturalmente deberá incluir tratamiento de textos, correo y navegación en Internet pero sobre todo deberá basarse el uso de la hoja de cálculo en entornos "decisionales".
- *A nivel corporativo, e*s clara la necesidad de un módulo de *Sistemas de Información para las organizaciones* que debería incorporar, a su vez (1) Teoría de sistemas, (2) Sistemas de Información, (3) Implicaciones organizativas, (4) ERPs, (5) De tareas a procesos - reingeniería y calidad - , (6) Visión estratégica, (7) Sistemas Interorganizacionales, (8) Sociedad Red, e-Commerce y e-Business, (9) Teletrabajo y organizaciones virtuales, pudiéndose completar con seminarios más especializados (*datawarehouse, datamining, datamarts*, modelos específicos de intermediación, etc).
- *Sistemas comunicacionales y de gestión del conocimiento.*
- *Tecnologías de la Información (Soporte tecnológico).*

De estos cuatro, hemos desarrollado la docencia de los dos iniciales (el primero en la EUETIB-UPC, y el segundo en la FIB-UPC) que son los que consideramos de tipo "universal". Los dos finales resultan más "específicos", no hemos tenido ocasión de impartirlos y deberán ser objeto de un trabajo posterior.

7. Consideraciones de"orientación"de los módulos y "metodología docente"

El módulo de *herramientas de productividad, información y comunicación personal* debe ir orientado a la generación de *modelos* de problemas reales que permitan su análisis a partir de la formulación de hipótesis. Este módulo deberá prescindir de detalles "operativos" y poner el "énfasis" en: la hoja como herramienta de modelización (*input*, modelo-paramétrico, output), análisis inverso (objetivo), optimización (*solver*), tablas y gráficos dinámicos, análisis de sensibilidad del modelo (*what if* ?) y análisis de escenarios desde la perspectiva de la prospectiva,. Es muy conveniente coordinar este módulo con "*workshops*" de aplicación específica a materias concretas. De hecho, como metodología proponemos una combinación de

- Ejercicio guiado (base del aprendizaje inicial).
- Trabajo de consolidación (en grupo).
- Casos abiertos (con directrices).
- *Workshop* final, con presentación.

En cuanto al módulo de *Sistemas de Información para las organizaciones*, proponemos

- Temas de presentación del profesor
- Exposición de artículos por parte de los alumnos (trabajo cooperativo guiado a partir de material facilitado por el profesor). Acceso a material Web.
- Discusión de casos, con existencia de grupo antagónico.
- Pero sobre todo, hemos experimentado unos resultados extraordinarios (en la FIB, UPC) con el concepto de CONGRESO como base del trabajo cooperativo. El Congreso se centra en las 3 últimas semanas del curso y en él los grupos de alumnos presentan una ponencia que es discutida en clase y valorada por ellos mismos.

La valoración de los alumnos, en este caso, ha sido de 6,0 en una escala Likert de 7.

Resumen y conclusiones

Las TSI constituyen un elemento de uso generalizado que se introduce en nuestras vidas y se convierte en herramienta imprescindible que permite modificar la estructura de los sectores industriales y la manera de realizar nuestro trabajo. La innovación es necesaria. La tecnología se convierte en arma esencial de esta innovación, pero a diferencia de las épocas de nuestros antepasados la tecnología actual no se incorpora de modo automático. La formación en TSI es necesaria, independientemente de la profesión.

La universidad tiene el deber de estar al frente de este movimiento, no sea que hagamos buena aquella frase de *que los profesores universitarios, de tanto enseñar, se olvidaron de aprender.*

Es curioso constatar que los dos módulos comunes identificados no existen (a veces, ni como materias optativas), en muchos planes de estudios. A veces, ni siquiera en ingenierías informáticas. Un mal principio para comenzar a dibujar un espacio común para los equipos multidisciplinares del mundo real.

El presente trabajo pretende constituir un impulso (y una primera guía) para comenzar a movernos en la dirección de incorporar estos módulos con sus materias, orientaciones, e incluso planteamientos didácticos.

Referencias

[1] Raúl Mario Abril, Manuel Costa, Josep Rucabado, Ferran Virgós. *Modelos y métodos: una evolución paralela*. Revista Novática. Vol XI, num 65, octubre 1985

[2] Raúl Mario Abril & Rafael Macau *Visión estratégica de los Sistemas de Información.* Datamation. Enero 1988

[3] Rafael Andreu, Joan E. Ricart & Josep Valor. *Planificación estratégica de tecnologías y sistemas de información en la empresa*. IESE, 1990

[4]] R. Andreu, J.E. Ricart, J. Valor. *La organización en la era de la información.* McGraw-Hill / IESE. Barcelona, 1996.

[5] R. N. Anthony. Planning and control systems: a framework for analysis. Harvard University, 1965

[6] Stafford Beer. The brain of the firm. John Wiley & sons. 1981.

[7] Patricia Compañ Rosique, Rafael Molina y Lorenzo Carbonell.. *Informática para economistas y juristas: enfoque docente e nla universidad de Alicante*. Jenui 98.

[8] H.M Deitel, P.J. Deitel & K. Steinbuhler. e-Business & e-Commerce for managers. Prentice Hall. 2001

[9] John T. Gorgone et al. *IS 2002-Final report of the undergraduate information systems model curriculum. Twenty third International Conference on Information Systems. Barcelona, 2002*.

[10] J. L. Le Moigne. *Les systèmes de information dans les organisations*. Presses Universitaires de France. 1973.

[11] J. L. Le Moigne. *Les systèmes de décision dans les organisation*s. Presses universitaires de France. 1974.

[12] José Antonio Ortega Martinez. *Ventaja competitiva y sistemas de información: un enfoque estratégico*. Harvard Deusto.

[13] Porter, M.E *Competitive strategy* The Free Press. 1980.

[14] M.E. Porter. *Competitive advantage*. The free Press. 1985.

[15] Jeffrey F. Rayport & John J. Sviokla. *Exploiting the virtual value chain*. Harvard Business Review. Nov-dec 1995.

[16] John F. Rockart. *Chief executive define their own data needs.* Harvard Business Review. Marzo-abril 1979. (versión castellana en Harvard-Deusto Business Review).

[17] Número monográfico de la revista Novática. *El impacto organizativo de las TI* (varios artículos). Novática, n 104. Julio-agosto 1993

[18] Michael J. Earl & Jeanne W. Ross. *Eigth imperatives for new IT organization*. Original Sloan Management Review, reproducido en Harvard Deusto.

[19] José Manuel Rodríguez. *Los estudios de informática en los currícula de los economistas. Una propuesta alternativa*. Jenui, 1998.

[20] Ferran Virgós Bel y Joan Segura Casanovas. Propuesta metodológica para la definición de contenidos de formación en TI/SI para titulaciones del área de empresa. Aplicación práctica. Actas Jenui, 1999.

Del propósito de la materia de compiladores en la formación del ingeniero informático

JosuKa Díaz Labrador, José Mª Sáenz Ruiz de Velasco
Dpto. de Ingeniería del Software
Universidad de Deusto
Apartado 1 - 48080 Bilbao
e-mail: josuka@eside.deusto.es, jmsaenz@eside.deusto.es

Resumen

Se presenta una reflexión sobre el papel que puede jugar la materia de compiladores en el plan de estudios de la ingeniería informática. Se argumenta que, además del objetivo obvio de enseñar a construir un compilador de un lenguaje de programación, pueden extraerse otras aportaciones. Se ha identificado que una de ellas es completar la formación que hasta el momento ha adquirido el estudiante sobre la tarea de programar y su herramienta (el lenguaje de programación), y que se trata de una aportación relevante y necesaria para este propósito. En segundo lugar, se considera que la aplicación de los conocimientos de compiladores no se restringe al problema de implementar un compilador en el sentido clásico, sino que al extender su concepto, pueden también aparecer nuevas aplicaciones, que incluso se dan con más frecuencia que la original.

1. Introducción

La materia de *Compiladores* es una de las troncales de 2º ciclo, con una carga lectiva de 9 créditos, en el Plan de Estudios de Ingeniero en Informática fijado por el Ministerio de Educación y Ciencia [10]. A pesar del título («*Procesadores de lenguaje*»), la descripción que aparece a continuación («*Compiladores, Traductores e Intérpretes. Fases de Compilación. Optimización de código. Macroprocesadores*») permite suponer que se pretendía un enfoque clásico (según el texto de Aho, Sethi y Ullman [1], referencia aún obligada pese a su antigüedad), en el que el objetivo prácticamente único del compilador es procesar lenguajes de programación.

Sin embargo, a la luz de los ya más de 10 años transcurridos desde el establecimiento de dicho plan de estudios, parece necesario (si no lo es en todo momento) volver a preguntarse sobre el papel que cumple la materia de compiladores en la formación del ingeniero en informática, sobre su orientación, y sobre sus contenidos teóricos y prácticos.

Muchas de las ideas que se recogen en este trabajo se desarrollaron en el año 2000 como parte del proyecto docente del primer autor para las asignaturas de compiladores en la Universidad de Deusto. Sin embargo, hemos constatado en otros docentes preocupación por estas cuestiones, y ha sido para nosotros una grata sorpresa encontrar un trabajo muy reciente de Debray [4], que realiza un análisis similar, y llega a conclusiones en parte comunes: sus ideas se irán desgranando en las sucesivas secciones.

En la sección 2 se presentan los resultados de un estudio (no exhaustivo) que se ha realizado de la implementación de la asignatura en varias universidades españolas. En la sección siguiente se plantean las cuestiones que (además de la razón obvia de mostrar qué es y cómo se construye un compilador) pueden justificar el interés de la materia.

En la cuarta, se argumenta que la misma aporta adicionalmente una serie de conocimientos y perspectivas que completan o afianzan los que el estudiante ha adquirido en relación con el concepto de programar. Después, la sección quinta muestra que la aplicación de los conocimientos de compiladores se extiende más allá del problema concreto de construir un traductor de un lenguaje de programación, y que la clase de estos problemas afines es bastante más común (al menos desde hace unos años) que el

problema anterior. Las conclusiones recogen una redacción tentativa de los objetivos de la asignatura, a la luz de lo discutido, y se analiza su efecto sobre los contenidos.

2. Situación actual

Una revisión de los planes de estudios de algo más de la mitad (30 sobre 44) de las universidades españolas que ofrecen la asignatura (como troncal de 2º ciclo), permite alcanzar varias conclusiones.

1. Aunque existen algunos casos, no se suele aumentar la carga lectiva fijada por la troncalidad (23 de las 30 analizadas mantienen los 9 créditos, y solo en un caso se llega a 15 créditos). Esto puede explicarse porque muchos centros deciden seguir la troncalidad, o porque el segundo ciclo en que se encuentra la materia no permite fácilmente la adición de créditos extra, pero también porque no se percibe necesidad de una mayor carga. Esto contrasta, sin embargo, con otras materias, por ejemplo, sistemas operativos (aunque es de primer ciclo), en que solo dos universidades siguen los 6 créditos troncales, siendo al menos 17 los centros analizados que dan más de 12 créditos, entre troncales y obligatorias.
2. Salvo muy contadas excepciones, la implementación de la materia sigue el enfoque que hemos llamado clásico, de compilación de los lenguajes de programación. Entre los centros de los que se ha podido recabar información, solo se han encontrado dos aspectos afines adicionales: procesamiento de lenguajes naturales (Camacho [3]) y lenguajes de la tecnología *web* (Barrutieta [2]).
3. Existe un enfoque de diseño prácticamente unánime en la presentación de la materia: el objetivo final es que el estudiante *sea capaz de construir* un compilador para un lenguaje de programación. Este enfoque, que se contrapone a una perspectiva analítica o descriptiva de la asignatura, es totalmente adecuado en la medida que se están formando ingenieros.
4. También es casi unánime incluir unas prácticas más o menos extensas, en que se hace uso de las conocidas herramientas de generación de compiladores en sus diversas versiones, algunas de las cuales han sido mostradas en ediciones anteriores de las JENUI (por ejemplo Mascaró y Orrell [9]).

Sin embargo, es interesante añadir algo más (impresiones subjetivas esta vez) a los datos precedentes. Pensamos que los estudiantes (lamentablemente, también algunos docentes) observan esta asignatura como una especie de "isla" en su formación. La idea de fondo es similar a la que se tiene respecto a la teoría de autómatas y lenguajes formales o la metodología de la programación: todo el mundo tiene claro que "parecen necesarias", pero pocos (quitando a los propios docentes y honrosas excepciones) les conceden mayor relevancia. La materia de compiladores parece verse como necesaria por el mero hecho de ser el sistema informático que permite en la práctica la tarea de programar, pero no se aporta mucho más acerca de su papel, ni se pone en relación con otras materias anteriores o posteriores de los estudios.

Por lo tanto, pensamos que resulta preciso profundizar en la razón de ser de los compiladores, no con el objetivo de una posible actualización de la materia (que creemos que en principio no se necesita), sino más bien para ser capaces de motivar a nuestros estudiantes, de situar la materia en el edificio de conocimientos que están construyendo, y prepararles para el aprovechamiento futuro de los mismos.

3. Interés de los compiladores

A primera vista, parece obvio pensar que las razones de que haya que enseñar compiladores a los estudiantes de informática son dos: primero, el compilador es un sistema informático suficientemente importante como para que sea conocido en detalle, y segundo, una de las hipotéticas tareas del profesional informático puede ser construir un compilador.

Adoptar un enfoque de diseño en la presentación de la asignatura es entonces doblemente adecuado, pues permite contestar el *qué* y el *cómo*, que son los objetivos planteados antes.

Ahora bien, si profundizamos en las razones anteriores, se plantean inmediatamente dos cuestiones. Primera: ¿por qué es importante lo que debe conocerse del compilador? Es decir, ya sabemos que casi todas las materias aportan una formación, un "poso de conocimientos", que justifican por sí mismos su interés, pero ¿podemos ser más explícitos? ¿Qué aportan los procesadores

de lenguajes concretamente a *la formación que hasta ahora* ha tenido el estudiante?

La segunda cuestión, expresada llanamente, es la siguiente: en verdad, ¿quién se dedica (sin necesidad de restringirse a España, incluso) a construir compiladores? Es decir, ¿en algún momento de su vida profesional van nuestros alumnos a poner en juego esos conocimientos adquiridos? Este argumento "utilitarista" suele repelernos a los académicos (nosotros al menos compartimos las ideas de Giner [6]), y desde luego, no es siquiera necesaria una respuesta si somos capaces de contestar adecuadamente a la primera pregunta. Pero hemos de recordar que estamos formando ingenieros (quizás también a algunos científicos), por lo que esta cuestión, bien entendida, se convierte en relevante.

Además, identificamos en ella una de las posibles razones por las que el estudiante no se encuentra suficientemente motivado ante asignaturas como compiladores (u otras similares como sistemas operativos). El alumno percibe (a partir de su ya amplia experiencia como usuario de esos sistemas) que son enormemente complejos, que muy pocos se dedican a ellos, y por tanto, no se conciben a sí mismos como posibles futuros participantes en su desarrollo.

Debray [4] hace exactamente este análisis (como muchos otros trabajos de mejora docente en compiladores, como el de Vegdahl [13], por citar uno de los más recientes que hemos conocido), y veremos que sus conclusiones son ciertamente similares a las nuestras.

Las dos cuestiones planteadas se desarrollan en sendas secciones a continuación.

4. Importancia del conocimiento de los compiladores

Un compilador es, en sentido estricto, un sistema intrínsecamente ligado a la programación, que a su vez, es una de las tareas fundamentales del informático. Por lo tanto, el conocimiento de los compiladores influirá en la maestría con que se lleve a cabo esta tarea.

En efecto, durante el primer ciclo de la carrera de informática se dedica un considerable número de créditos a los aspectos teóricos y prácticos relacionados con el concepto de programar: fundamentos de la programación y de los lenguajes, metodología, paradigmas, estructuras de datos y algoritmia, acompañados del conocimiento de (normalmente) varios lenguajes de programación. Se supone que el estudiante que termina el primer ciclo (o la carrera técnica, por poner el caso) es diestro al menos en programar y usar las herramientas de programación (entornos de desarrollo y, precisamente, compiladores).

Sin embargo, nuestra conclusión a este respecto es que todavía *falta* algo: un cierto conjunto de conocimientos que han de añadirse a los ya disponibles para que la maestría en programar pueda considerarse completa.

4.1. Comprensión de la definición de los lenguajes

Como primera aportación, destacamos la *sistematización* de la definición de los lenguajes de programación. El uso de modelos matemáticos formales (autómatas, expresiones regulares, gramáticas independientes del contexto, gramáticas de atributos, etc.) para definir rigurosamente el léxico, la sintaxis y la semántica de los lenguajes de programación aporta un sustento único al que afianzarse a la hora de comprender el lenguaje de programación.

Digamos que hasta ahora el conocimiento de los lenguajes era "intuitivo": el estudiante domina la escritura en ellos como un niño aprende a hablar, pero carece del edificio necesario para comprender la estructura de la herramienta. Después de ver cómo los modelos matemáticos formales ayudan a expresar la definición de un lenguaje, el futuro informático estará preparado para comprenderlo de forma organizada.

Esta idea es importante en relación con otro aspecto, que es la evolución de los lenguajes de programación. Hoy enseñamos Pascal, C, Ada, Modula, C++, Java, Eiffel, Lisp, Scheme, ML, Haskell... pero el día de mañana estas serán "lenguas muertas", y surgirán nuevos lenguajes, por lo que es importante que los conocimientos que transmitimos no sean caducos, y puedan aplicarse igualmente en los retos venideros. Cuando un ingeniero informático se enfrente a la tarea de aprender un nuevo lenguaje, el sustento de los compiladores le ayudará a sistematizar la mirada (elementos léxicos, estructuras sintácticas, aspectos semánticos), comprender rápidamente sus entidades (con ayuda de los modelos formales

mencionados), y fijarse en los detalles que puedan ser importantes.

Hay incluso una tercera ventaja, que es el mejor aprovechamiento de los mensajes de error. Los estudiantes han tenido ya una dilatada experiencia de uso de los compiladores, y probablemente están unánimemente de acuerdo en que los mensajes de error resultan muchas veces crípticos, cuando no equivocados, y no acaban de comprender la razón de que se señale cierto error, cuando claramente (según su percepción) es otro. El conocimiento de las fases de análisis, su funcionamiento y la gestión de errores resulta determinante para que se puedan comprender mejor los mensajes que emite el compilador, con el resultado indudable de favorecer el ciclo de composición de programas.

4.2. Comprensión de la implementación de los lenguajes de programación

En segundo lugar, encontramos una serie de aspectos relacionados con la generación de código. Todos ellos se dirigen a que el estudiante sea consciente de qué es un programa traducido, cuando se ejecuta (objetivo final de programar), y cómo puede mejorarse su calidad.

De nuevo, o quizá más que en el caso anterior, los estudiantes tienen una serie de ideas más bien difusas sobre el asunto: alcanzan a saber como mucho que hay una pila (que tiene direcciones de retorno) y un montículo (para la memoria dinámica), que se dice que "los datos están en memoria" (sin ser plenamente conscientes de lo que esto significa o relacionarlo con lo anterior), que se usan los registros de la máquina, que se pueden aplicar optimaciones, etc.

Mientras, es indudable que solo un conocimiento pleno de estos aspectos puede ayudar a explicar lo bueno o lo malo que es un programa (queremos decir, una vez que se han asegurado todas las cuestiones algorítmicas, metodológicas y de estructuras de datos involucradas, evidentemente).

La estructura del programa objeto, el uso de la pila y el montículo, el concepto de registro de activación, las secuencias de llamada y retorno, la asignación estática de datos locales en la pila, la importancia de la existencia o no de anidamiento de bloques en el lenguaje, la inicialización de datos globales y locales, el paso de parámetros, las estrategias de asignación de registros para la evaluación de expresiones y sentencias, las optimaciones que pueden practicarse, etc. forman un bloque de aspectos ineludibles para comprender finalmente los programas.

4.3. Comentarios adicionales

En definitiva, nuestra postura es que la enseñanza de los compiladores aporta al futuro informático una serie de conocimientos que afianzan, o terminan de fijar, una de las tareas que se supone que este ha de realizar, como es la de programar, y pensamos que ello viene dado por dos aspectos principales: primero, la sistematización o estructuración del concepto de lenguaje de programación, y segundo, la comprensión de la implementación de los mismos.

Ambos aspectos pueden hacerse aparentes sin necesidad de cambiar el temario o enfoque tradicionales de la materia. El primer aspecto puede surgir al presentar las fases del *front-end* del compilador, mientras que al segundo se le suele dedicar una lección (capítulo 7 de Aho, Sethi y Ullman [1]).

Una de nuestras estrategias docentes al respecto es plantear durante el curso múltiples cuestiones sobre los lenguajes de programación conocidos por los estudiantes, del tipo:

- ¿es `integer` una palabra reservada de Pascal?
- ¿cuesta lo mismo inicializar variables locales y variables globales en C?

cuyas respuestas (negativas en estos casos) sorprenden a muchos, lo que les lleva a reflexionar y darse cuenta que es precisa otra visión (a la vez sistemática y de implementación) para comprender *plenamente* la programación y los lenguajes que la permiten.

Hay que tener en cuenta que en algunos planes de estudio se encuentra (normalmente en tercer curso) una asignatura que, entre otras cosas, trata justamente de algunos de estos temas (principalmente el segundo, esto es, la implementación de los lenguajes): lo que podría llamarse quizá "lingüística de programación", tal como se refleja, entre muchos otros, en los textos de Sethi [12] o MacLennan [8].

Su presencia en esos planes de estudios responde indudablemente a los motivos

expresados antes respecto a compiladores, y se supone que, entonces, en dichos planes esta última asignatura se dedica a profundizar más en los aspectos complejos propios del problema.

5. Aplicación de las técnicas de compilación

Recordemos la segunda cuestión planteada al final de la sección 3: ¿tiene utilidad práctica, en cuanto a desarrollo de la profesión, el conocimiento de los compiladores? Desde luego, puede decirse que un porcentaje realmente pequeño de los informáticos construye alguna vez en su vida un compilador (en sentido estricto, es decir, para procesar un lenguaje de programación), pero de la misma forma, aún es menor el número de quienes desarrollan un sistema operativo o una base de datos, así que este argumento debe ser rebatido a quienes (normalmente alumnos poco motivados) lo exponen.

Nos referimos a este aspecto incluso recordando que normalmente enseñamos a *diseñar e implementar* un compilador, es decir, se completa la cuestión anterior con esta otra: ¿por qué es importante este enfoque de diseño?

Debray [4], ante este problema, sostiene una idea básica que apoyamos plenamente: «*the principles, techniques, and tools discussed in compiler design courses are nevertheless applicable to a wide variety of situations that would generally not be considered to be compiler design*». Sin embargo, reconsideramos este argumento a través de dos facetas: primero, los métodos y herramientas de compiladores resultan relevantes en sí mismos como nuevos utensilios para la resolución de problemas (tarea final de la informática), y segundo, su aplicación a problemas que a primera vista no parecen de compilación es más habitual de lo que parece.

5.1. Nuevos métodos y herramientas de solución de problemas

La informática busca la solución de problemas. Los estudiantes llegan al segundo ciclo de la carrera con un notable repertorio de teorías, métodos y herramientas dirigidos a este propósito, pero es evidente que aún existen muchas más. Nuestra postura es que el diseño de compiladores aporta un conjunto de ellas que son relevantes y de aplicación en un amplio rango de problemas: se ejercitarán en la asignatura de compiladores bajo el objetivo concreto de construir un compilador, pero debemos recalcar a los alumnos que su aplicación excede ampliamente este problema concreto, y que esa es precisamente una de las razones por las que resulta importante la materia.

En primer lugar, encontramos los modelos matemáticos formales típicos (expresiones regulares, autómatas finitos y con pila, máquinas de Moore y de Mealy, gramáticas independientes del contexto). No suele ser habitual, creemos, que se dedique mucho tiempo, en la asignatura correspondiente del primer ciclo, a la implementación práctica de estos modelos, por lo que hacerlo en compiladores (al explicar las fases de análisis léxico y sintáctico) aporta ya de por sí una nueva herramienta de trabajo al informático.

En segundo lugar, aparecen los algoritmos propios de análisis sintáctico (sobre todo descendente, factible sin disponer de herramienta de generación), y los mecanismos de procesamiento semántico (esquemas de traducción y gramáticas de atributos), que se convierten en útiles adicionales.

Finalmente, pero como aspecto más importante, disponemos de los archiconocidos programas de generación de compiladores como lex y yacc (o sus variantes; ver Levine *et al.* [7]), ANTLR/PCCTS (ver Parr y Quong [11]), y similares. Nosotros, una vez más, los enseñamos en el contexto concreto de implementación de compiladores, pero nunca se insistirá lo suficiente en la capacidad que tienen dichos programas para resolver problemas cuya "apariencia" no es la de un compilador.

Hay un aspecto que muchas veces se olvida al plantear la bondad de estas herramientas de generación de compiladores. Desde un punto de vista reduccionista, no hacen realmente nada nuevo aparte de implementar un autómata o una gramática (de la misma forma que los programadores de verdad no necesitan más que el ensamblador). Sin embargo, la *rapidez* con la que podemos resolver cada problema y, sobre todo, la *confianza* (casi) plena que podemos depositar en los programas que generan hacen que las consideremos herramientas insustituibles entre los utensilios del informático.

5.2. Problemas que no son de compilación

Todos los textos de compiladores hablan de las aplicaciones de las técnicas de compilación, pero la inmensa mayoría está relacionada de una u otra forma con lenguajes de programación. Se citan algunos otros ejemplos de lenguajes que tienen otros propósitos, principalmente TEX y similares (aunque es difícil discernir si TEX *no* es un lenguaje de programación), pero poco más. Además, como la presentación de la materia de compiladores se realiza tomando el procesamiento de los lenguajes de programación como ejemplo único, la mayor parte de los estudiantes saca la conclusión de que los conocimientos de la asignatura solo se aplicarán si alguien les manda construir un compilador de un lenguaje de programación.

Como dijimos al principio, es claro que este problema concreto se presenta a un número bien reducido de profesionales. Sin embargo, pensamos que existen más problemas en que pueden aplicarse las técnicas de compilación.

De nuevo, Debray [4] presenta dos ejemplos concretos de problemas (una descripción textual de grafos que genera imágenes Postscript y un conversor de LATEX a HTML) que no son estrictamente de compilación, pero que se resuelven claramente con las técnicas de compilación, y en particular, con las herramientas mencionadas.

Sin embargo, Debray aboga como conclusión por «*consider compilers as just one instance of translators, broadly, from (almost) any arbitrary source language to (almost) any arbitrary target language*». Nosotros planteamos que esta generalización de la compilación como traducción, primero, siendo relevante, es casi un lugar común en la presentación inicial de la materia (véase cualquier apunte de la asignatura, como el de Eguíluz y Díaz [5]), y segundo, se encuentra todavía demasiado "cerca" del concepto original, apartada entonces de muchas de sus aplicaciones.

Nuestra postura es que la condición para plantearse una posible aplicación de las técnicas de compilación a un problema es únicamente la intervención de un *lenguaje (formal) artificial* en su especificación (normalmente como datos de entrada del problema), y que esta no necesita estar elaborada específicamente como una traducción (aunque en el fondo lo sea).

El ejemplo típico que solemos comentar es el navegador de Internet, en su faceta de presentación textual y/o gráfica de contenido HTML. A pesar de que nosotros identificamos formalmente que este programa es un *traductor* de HTML a secuencias de posicionamiento de elementos gráficos en la pantalla, es difícil a primera vista hacer esta interpretación del problema. Realmente, sin embargo, bastaría con darse cuenta que interviene HTML como entrada: desde el momento en que es un lenguaje artificial suficientemente rico, debería sospecharse que se pueden aplicar los conocimientos de compiladores, y ello sea lo que sea lo que se quiere hacer con esa entrada.

Esta idea, además, cobra gran relevancia desde hace pocos años, no solo por el advenimiento de Internet y el conjunto de lenguajes asociados, sino en la medida que la información se almacena cada vez menos en forma binaria, y cada vez más de forma textual. El siguiente ejemplo se nos presentó hace pocos meses en un contexto no relacionado con nuestra docencia de compiladores.

Imaginemos que nos proporcionan una (enorme) base de datos supuestamente codificada en XML, con la salvedad de que los datos (obtenidos de una fuente no estructurada) no han sido bien codificados, es decir, el documento XML resultante no está bien formado. Concretamente, entre otras dificultades, ciertas etiquetas no tienen su cierre y otras no están correctamente anidadas, como se ve en el siguiente ejemplo:

```
<a> xxx <b> yyy </a> zzz </b>
```

Lamentablemente, ocurre que no es posible modificar el programa de codificación para obtener un documento bien formado (o aunque se pudiera, la fuente de la información original no estructurada ya no se encuentra disponible), es decir, solo puede tratarse el documento pseudo-XML original para corregir los errores.

Este ejemplo es, creemos, paradigmático del *procesamiento de información textual* no relacionado *a priori* con los compiladores, ni siquiera desde la perspectiva generalizada de traducción: «oye, es que tengo un documento

XML mal formado que quiero corregir» es lo que nos plantearon. De nuevo, la mera aparición de XML como entrada es lo que debe encender la bombilla de las técnicas de compilación.

Pero incluso el ejemplo nos parece interesante porque muestra que no es necesaria siquiera una traducción estricta:

- entrada: XML mal formado;
- salida: XML bien formado;
- método: hacer el *parser* de XML mal formado y generar código de XML bien formado.

como solución.

Nuestra estrategia fue analizar someramente las modificaciones necesarias, y observar que no requerían la construcción de un *parser* completo de XML mal formado (lo que llevaría un considerable esfuerzo, incluso con técnicas de compilación). Al contrario, las partes problemáticas seguían patrones muy claros, y se podían identificar por el contexto. Con ello, nos fue suficiente recurrir a lex, particularmente haciendo uso de los estados de análisis que permite esta herramienta.

Por tanto, el ejemplo muestra que ni en la especificación ni en la solución resulta muy útil un enfoque de traducción, por mucho que lo que hagamos en el fondo con el programa lex no sea más que eso exactamente. El enfoque que funciona es el de procesamiento: si la entrada es un lenguaje formal, habrá que procesarla (para lo que sea), y ello lleva a considerar las técnicas o herramientas de compilación como posible vehículo de la solución.

5.3. Comentarios adicionales

Esta perspectiva hace que cobren nueva importancia los aspectos del *front-end* del compilador, y en particular, las herramientas de generación de compiladores, percibidas más bien como programas de ayuda al diseño rápido, efectivo y seguro de soluciones para problemas de procesamiento de información. Ello es una razón más para insistir como docentes en la parte práctica de la materia, tal como ya se ha visto (en la sección 2) que se lleva a cabo en la mayor parte de los centros. Quizá sería conveniente incluir como práctica adicional algún pequeño problema del tipo presentado antes, además de los consabidos lenguajes de programación simples.

Por otro lado, es cierto que bajo este enfoque resultan de menor aplicabilidad los aspectos del *back-end* del compilador clásico (generación de código intermedio y objeto, optimación). Ello no justifica en ningún modo (al menos debido al interés planteado en la sección 4) su posible marginación en el temario.

Ahora bien, según la importancia que cada docente quiera conceder a las ideas aquí expuestas, pueden considerarse dos presentaciones alternativas de la materia de compiladores a este respecto (nuestro interés sería realmente no tener que perder ningún aspecto, pero ya se ha visto lo difícil que resulta superar el límite de los 9 créditos):

- la presentación tradicional, en que prevalece el interés de la implementación de los lenguajes de programación, con fuerte profundización en los aspectos señalados del *back-end*;
- la presentación alternativa, en que prevalece la aplicación a problemas de procesamiento de información textual, más sencilla (descriptiva y no tanto sintética) en los aspectos del *back-end*, y focalizada en la resolución de problemas como los mostrados.

6. Conclusión: objetivos de la materia de compiladores

La discusión precedente nos permite elaborar de una forma profunda los objetivos de la asignatura de compiladores.

- El objetivo genuino (y preponderante) de la materia es ciertamente *ser capaz de diseñar y desarrollar un compilador* de un lenguaje de programación: esto incluye la teoría, métodos e implementación de cada fase del compilador, y en particular, las herramientas de generación de compiladores.
- El segundo objetivo es *afianzar la tarea de programar*, es decir, añadir una serie de perspectivas y conocimientos (quizá no fundamentales, pero sí relevantes y necesarios) al edificio que ya tiene el estudiante. Se enfatizan dos factores: primero, la sistematización de la definición de los lenguajes de programación (y artificiales), y segundo, la comprensión de los modelos de

implementación de los lenguajes de programación.

- El tercer objetivo es *ser capaz de identificar y resolver problemas que no son de compilación* (de lenguajes de programación). Aquí, será importante, primero, comprender que la aplicación de los conocimientos de compiladores incluye el concepto general de lenguaje formal artificial, segundo, ser capaz de identificar problemas que parecen alejados de la compilación como problemas de procesamiento de lenguajes, y tercero, comprender que los algoritmos y las herramientas de generación de compiladores se convierten entonces en utensilios relevantes para el profesional informático.

Se ha visto que, dados estos objetivos, los contenidos de la asignatura (tal como se explican de forma casi unánime en nuestras universidades) no tienen que experimentar ninguna modificación significativa, en la medida que siguen siendo adecuados al primero y principal, y que mediante sencillas técnicas docentes se puede asegurar el cumplimiento de los otros dos. En todo caso, si se desea una orientación más fuerte al tercer objetivo, se puede simplificar ligeramente la explicación de la fase de generación de código.

Referencias

[1] Aho, A.V.; Sethi, R.; Ullman, J.D. *Compilers. Principles, Techniques, and Tools.* Addison-Wesley, 1986 (traducción al castellano: *Compiladores. Principios, técnicas y herramientas*. Addison-Wesley Iberoamericana, 1990)

[2] Barrutieta Anduiza, G. "Procesadores de lenguaje". Escuela Politécnica Superior, Universidad de Mondragón, sin fecha (`http://www.mondragon.edu/bin/tusestudios/pdfs/asignaturas/IF2403.pdf`).

[3] Camacho Fernández, D. "Procesadores de Lenguaje II (13-10830)". Universidad Carlos III de Madrid, revisión 04/04/2002 (`http://www.uc3m.es/uc3m/gral/ES/ESCU/1310830.html`).

[4] Debray, S. "Making Compiler Design Relevant for Students who will (Most Likely) Never Design a Compiler". *Proceedings of the 33rd SIGCSE technical symposium on Computer Science Education*, pp. 341-345, Covington, Kentucky, EE.UU., 2002.

[5] Eguíluz Morán, A.; Díaz Labrador, J. *Teoría de compiladores. Primer Parcial.* Universidad de Deusto, 1995.

[6] Giner Sanjulián, S. "La Universidad y la falacia utilitarista". *Conferencia: Los objetivos de la Universidad ante el nuevo siglo*, Universidad de Salamanca, 17 y 18 de noviembre de 1997 (`http://www.crue.org/pginersa.htm`)

[7] Levine, J.R.; Mason, T.; Brown, D. *lex & yacc (second edition).* O'Reilly & Associates, 1992.

[8] MacLennan, B.J. *Principles of Programming Languages. Design, Evaluation, and Implementation (second edition).* Harcourt Brace Jovanovich College Publishers, 1987.

[9] Mascaró Portells, M.; Orell Más, C. "Entorno de programación Java para la docencia y desarrollo de procesadores de lenguajes". En José Miró Julià, editor, *Actas de las VII Jornadas de Enseñanza Universitaria de Informática, Jenui 2001*, pp. 204-209, Palma de Mallorca, julio 2001.

[10] Ministerio de Educación Cultura y Deporte *Troncales de Ingeniero en Informática.* 1990, revisión 06/05/2002 (`http://www.mec.es/consejou/titulos/troncal/inginfor.html`).

[11] Parr, T.J.; Quong, R.W. "ANTLR: A Predicated-LL(k) Parser Generator". *Software Practice & Experience*, 25, 7, 1995, pp. 789-810.

[12] Sethi, R. *Programming Languages. Concepts and Constructs.* Addison-Wesley, 1989 (traducción al castellano: *Lenguajes de programación. Conceptos y constructores*. Addison-Wesley Iberoamericana, 1992).

[13] Vegdahl, S.R. "Using visualization tools to teach compiler design". *Proceedings of the second annual CCSC on Computing in Small Colleges Northwestern conference*, pp. 72-83, Beaverton, Oregon, EE.UU., 2000.

Programación, algoritmos y estructuras de datos

Representaciones gráficas y Mundos Virtuales infinitos en las Prácticas de *Programación Lógica y Funcional*

Jose Emilio Labra Gayo
Dpto. de Informática
Universidad de Oviedo
CP. 33007 Oviedo, Spain
e-mail: labra@lsi.uniovi.es

Resumen

En el esquema de prácticas de la asignatura de *programación lógica y funcional* se ha incorporado la generación de representaciones gráficas y de *quadtrees* y *octrees* para la enseñanza de las diferentes características de este tipo de lenguajes: funciones de orden superior, evaluación perezosa, polimorfismo, variables lógicas, satisfacción de restricciones, etc. La utilización de vocabularios XML estándar: SVG para gráficos y X3D para realidad virtual permite disponer de numerosas herramientas de visualización. La incorporación de este tipo de ejercicios puede facilitar la enseñanza y motivación de los estudiantes.

1. Introducción

La asignatura *Programación Lógica y Funcional* se imparte como optativa de 6 créditos en las titulaciones de Ingeniero Técnico de Informática de Gestión y de Sistemas en la Universidad de Oviedo. El resto de asignaturas de programación de dicha titulación se centra en lenguajes imperativos: Pascal, C, C++ y Java. Esta asignatura es, por tanto, el primer y en muchas ocasiones, único contacto que los estudiantes de estas titulaciones tienen con los lenguajes declarativos. Dado su carácter optativo, la supervivencia de la asignatura depende del número de alumnos matriculados. Aparte de la dificultad de una asignatura de programación, la elección de los estudiantes se ve condicionada por otros aspectos relacionados con este tipo de lenguajes: entornos de programación rudimentarios, escasez de sistemas gráficos de depuración y traza, carencia de librerías, dificultad para enlazar con librerías escritas en otros lenguajes, etc. Con el fin de ampliar la motivación de los estudiantes y el alcance de la asignatura, desde el curso 2001/02 se ha incorporado al proyecto IDEFIX [14,15] que persigue la enseñanza a través de Internet de lenguajes de programación.

La evaluación de la asignatura se realiza fundamentalmente mediante la realización de trabajos de programación. Los enunciados de estos trabajos siguen un esquema de presentación gradual basado en la taxonomía de objetivos cognitivos [12]: en primer lugar se presenta un programa correcto que los estudiantes deben compilar y ejecutar. Seguidamente, se les pide la realización de modificaciones para que demuestren que comprenden el programa y adquieran por imitación una habilidad básica de construcción de programas. Finalmente, se les solicita a los estudiantes la creación de programas que deben construir por su cuenta.

En este artículo se describe el esquema de prácticas utilizado y que se basa principalmente en la generación de ficheros XML que incluyen gráficos bidimensionales en SVG y mundos virtuales en X3D. El artículo comienza con un repaso a los vocabularios XML utilizados. A continuación se presenta el ejercicio básico que consiste en representaciones gráficas de funciones y permite manipular el concepto de funciones de orden superior. En la sección 4 se estudia la estructura recursiva de *quadtree*. En la sección 5 se presentan los *octrees* y la sección 6 se presenta la generación de mundos virtuales infinitos mediante evaluación perezosa. En la sección 7 se describe la creación de tipos de datos polimórficos. La sección 8 describe la utilización de características propias de la programación lógica (especialmente la utilización de variables lógicas y *no-determinismo*) y la siguiente sección se centra en la utilización de programación lógica con restricciones. Finalmente se resumen los principales trabajos relacionados y se detallan las principales conclusiones y líneas de investigación futuras.

Notación. A lo largo del artículo se utilizan fragmentos de código *Haskell* y *Prolog*. Se supone que el lector tiene ligeros conocimientos de la sintaxis de ambos lenguajes. También se supone cierto conocimiento de vocabularios XML.

2. Vocabularios XML

El lenguaje XML [3] se está convirtiendo en un estándar de intercambio y representación de información en Internet. Puesto que en el actual plan de estudios no se imparte dicho lenguaje en ninguna asignatura, se ha considerado que la utilización de vocabularios XML puede ser una buena oportunidad para que los estudiantes manejen dicho estándar.

Uno de los primeros ejercicios planteados es la construcción de una librería de funciones que permita generar ficheros XML. Esta librería, que los estudiantes construyen, permitirá utilizar una base común en el resto de prácticas que facilita la generación de ficheros con vocabularios XML específicos.

La librería contiene las siguientes funciones básicas:

- *vacio e as* genera un elemento vacío e con atributos as
- *gen e es* genera un elemento *e* con subelementos *es*
- *genAs e as es* genera un elemento e con atributos *as* y subelementos *es*

Uno de los vocabularios XML específicos que se utilizará es el lenguaje SVG (Scalable Vector Graphics) que se está convirtiendo en el principal estándar de representación de gráficos bidimensionales vectoriales en Internet.

Asimismo, el lenguaje VRML (*Virtual Reality Modelling Language*) se puede considerar el principal estándar de representación de mundos virtuales en Internet. Sin embargo, este lenguaje es anterior a XML y no sigue, por tanto, su sintaxis. Desde el año 2000, el consorcio Web3D ha diseñado el vocabulario X3D que puede considerarse una nueva versión de VRML adaptada a XML.

3. Funciones de orden superior: Representaciones gráficas

El segundo trabajo práctico que se plantea consiste en la representación gráfica de funciones. A los estudiantes se les presenta la función plotF que permite almacenar en un fichero SVG la representación gráfica de una función.

```
plotF ::(Double→Double)→String→IO ()
plotF f fn = writeFile fn (plot f)

plot::(Double→Double)→String
plot f = gen "svg" (plotPs ps)
 where
  plotPs       = concat . map mkline
  mkline (x,x')= line (p x) (p x') c
  ps           = zip ls (tail ls)
  ls           = [0..sizeX]
  p x          = (x0+x, sizeY - f x)

line (x,y) (x',y') c =
  vacio "line"
    [("x1",show x) ,("y1",show y),
     ("x2",show x'),("y2",show y')]

sizeX = 500; sizeY = 500; x0 = 10
```

Obsérvese que `plotF` realiza acciones de Entrada/Salida. Tradicionalmente, los cursos que enseñan el lenguaje *Haskell* tendían a retrasar la presentación del sistema de Entrada/Salida mediante mónadas. Creemos que una presentación gradual permite evitar malentendidos posteriores, ya que en caso contrario, muchos estudiantes consideraban extraña la posterior introducción de efectos laterales en un lenguaje que consideraban *puro*.

El código de la función `plot` sirve como ejemplo de utilización de los combinadores recursivos `zip`, `map`, `foldr`, etc. característicos del lenguaje *Haskell*.

En este trabajo práctico, los ejercicios que se proponen a los estudiantes utilizan el concepto de funciones de orden superior, clave del paradigma funcional. Así, por ejemplo, se solicita al estudiante que construya una función `plotMedia` que escriba en un fichero la representación de dos funciones y la función media de ambas. Por ejemplo, en la figura 1 se representa la media de las funciones `(\x -> 10 * sin x)` y `(\x -> 10 + sqrt x)`.

La solución del ejercicio en *Haskell* puede ser:

```
media f g = plotF (\x→(f x + g x)/2)
```

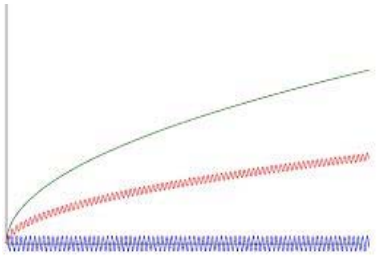


Figura 1. Representación de dos funciones y su media

4. Tipos de datos recursivos: *Quadtrees*

La siguiente unidad didáctica se centra en la presentación de técnicas de definición de tipos recursivos y su manipulación. Tradicionalmente, los trabajos prácticos en esta sección se realizaban con el tipo predefinido lista y con un tipo de datos definido por los alumnos para representar árboles binarios. Los estudiantes tienen serias dificultades para comprender los procesos recursivos y se sienten habitualmente poco motivados por este tipo de ejercicios [9,18]. Para intentar aumentar su motivación se trabajará con *quadtrees* [17] que son estructuras recursivas que permiten representar imágenes y tienen numerosas aplicaciones prácticas. En un *quadtree* las figuras se representan mediante un único color o la subdivisión de cuatro cuadrantes que son a su vez *quadtrees*. La generalización de los *quadtrees* al espacio tridimensional se denomina *octree* ya que cada subdivisión se realiza en ocho *octrees*. Estas estructuras son ampliamente utilizadas en el campo de la informática gráfica ya que permiten optimizar la representación interna de escenas y las bases de datos tridimensionales para sistemas de información geográfica.

En Haskell, un quadtree puede representarse mediante el siguiente tipo de datos:

```
data Color = RGB Int Int Int
data QT = B Color
        | D QT QT QT QT
```

Un ejemplo de quadtree sería

```
ejQT = D r g g (D r g g r)
  where r = B (RGB 255 0 0)
        g = B (RGB 0 255 0)
```

El ejemplo anterior define un *quadtree* formado por la subdivisión en cuatro cuadrantes, dos rojos y dos verdes dispuestos en diagonal. En la figura 2 puede observarse una representación del quadtree `ejQT`.

La visualización de *quadtrees* se realiza mediante una sencilla función que genera un fichero en formato SVG utilizando las funciones *vacio*, *gen,* y *genAs* implementadas por los estudiantes para la generación de documentos XML.

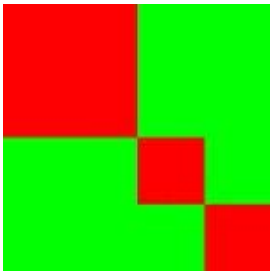


Figura 2. Ejemplo de *Quadtree*

```
type Punto = (Int,Int)
type Dim   = (Punto,Int)

verqt::QT→IO ()
verqt q = writeFile “qtree.svg”
      (gen “svg” (ver ((0,0),500) q))

ver::Dim→QT→String
ver ((x,y),d) (B c) =
 rect (x,y) (x+d+1,y+d+1) c

ver ((x,y),d) (D ul ur dl dr) =
  let d2 = d `div` 2
  in
   if d <= 0 then “”
   else ver ((x,y),d2)       ul ++
        ver ((x+d2,y),d2)    ur ++
        ver ((x,y+d2),d2)    dl ++
        ver ((x+d2,y+d2),d2) dr

rect::Punto→Punto→Color→String
rect (x,y) (x',y') (RGB r g b) =
 vacio "rect"
     [(“x”,show x),(“y”,show y),
      (“height”,show (abs (x - x'))),
      (“width”, show (abs (y - y'))),
      (“fill”, “rgb(”++show r++ “,”++
                       show g ++“,”++
                       show b ++“,”)]
```

5. *Octrees* y mundos virtuales

Un *octrree* es una generalización de un *quadtree* para representaciones tridimensionales: al subdividir cada cara de un cubo en cuatro partes se obtienen ocho cubos. La representación de *octrees* en *Haskell* podría ser la siguiente:

```
data OT = Vacio
        | Cubo Color
        | Esfera Color
        | D OT OT OT OT OT OT OT OT
```

La representación anterior indica que un *octree* puede estar vacío, ser un cubo o una esfera con un determinado color, o una división en ocho *octrees*.

Un ejemplo de *octree* sería:

```
ejOT :: OT
ejOT = D v e e v r v v g
 where v = Vacio
       e = Esfera (RGB 0 0 255)
       r = Cubo   (RGB 255 0 0)
       g = Cubo   (RGB 0 255 0)
```

A los estudiantes se les presenta la función

```
wOT :: OT → FileName → IO ()
```

que toma como argumentos un *octree,* un nombre de fichero y escribe en dicho fichero una representación en X3D del *octree*. En la figura 3 se presenta una pantalla capturada de la representación del oc*tree* ejOT en realidad virtual. Aunque en la figura se representa una versión impresa, el sistema genera un modelo virtual en el que los estudiantes pueden navegar.

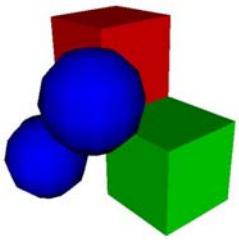


Figura 3. Ejemplo de *Octree*

6. Evaluación *Just-in time:* Mundos infinitos

Una de las principales características del lenguaje Haskell es la evaluación perezosa que permite definir algoritmos que manipulan estructuras potencialmente infinitas. La evaluación perezosa puede también considerarse un tipo de evaluación *Just-in time* en la que el sistema no evalúa los argumentos de una función hasta que realmente necesita su valor. El programador puede definir y manipular *quadtrees* infinitos. Por ejemplo, es posible definir:

```
inf :: OT
inf = D inf v s v v v r inf
 where v = Vacio
       s = Esfera (RGB 0 0 255)
       r = Cubo   (RGB 255 0 0)
```

Obsérvese que el *octree* se define en función de sí mismo. Su visualización se presenta en la figura 4.

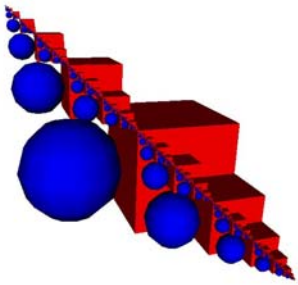


Figura 4. *Octree* infinito

Gracias a la evaluación perezosa, es posible definir funciones que manipulen mundos virtuales infinitos. Por ejemplo, la función `repite` toma como argumento un *octree* y genera un nuevo *octree* x repitiendo en cada cuadrante el *octree* x.

```
repite :: OT → OT
repite x = D x x x x x x x x
```

Al aplicar la función repite al *octree* inf se obtendría el resultado de la figura 5.

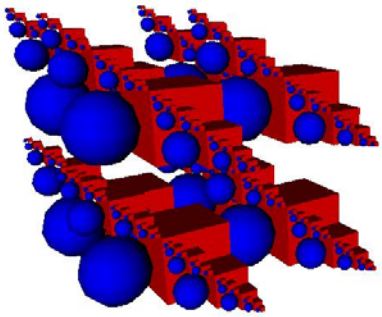


Figura 5. Repetición de un *Octree* infinito

7. Polimorfismo paramétrico: *Quadtrees* de alturas

El sistema de tipos del lenguaje *Haskell* admite la utilización de polimorfismo paramétrico. Las listas son el ejemplo tradicional de tipos de datos polimórfico. Aunque los *quadtrees* tradicionales contienen en cada cuadrante información del color, podría estudiarse una generalización que contuviese en cada cuadrante información de un tipo *a* que se pasa como parámetro. La nueva definición sería:

```
data QT a = B a
          | D (QT a) (QT a)
              (QT a) (QT a)
```

Los *quadtrees* con información de color serían valores de tipo `QT Color`.

La generalización anterior permite definir otros ejemplos de *quadtrees,* como los que contienen en cada cuadrante información de la altura (un valor de tipo `Float`). Estos *quadtrees* pueden utilizarse para la representación de alturas de terrenos. Por ejemplo el siguiente *quadtree*:

```
qta :: QT Float
qta = D x x x x
  where x = D b c a b
        a = B 0
        b = B 10
        c = B 20
```

se representa en la figura 6.

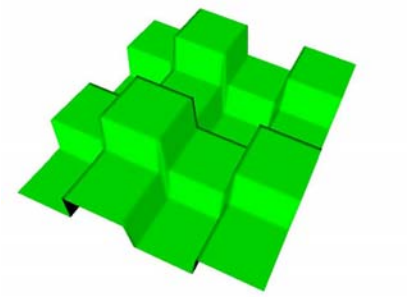


Figura 6. *Quadtree* de alturas

El lenguaje *Haskell* facilita y promueve la utilización de funciones genéricas. La función *foldr* es ejemplo de función genérica predefinida para el caso de las listas. Esta función puede también definirse para el tipo de datos *quadtree:*

```
foldQT::
  (b→b→b→b→b)→(a→b)→QT a→ b
foldQT f g (B x) = g x
foldQT f g (D a b c d) =
 f (foldQT f g a) (foldQT f b)
   (foldQT f g c) (foldQT f d)
```

Es posible definir múltiples funciones a partir de *foldQT*. Por ejemplo, para calcular la lista de valores de un *quadtree* puede definirse:

```
valores :: QT a → [a]
valores = foldQT
    (\a b c d → a ++ b ++ c ++ d)
    (\x → [x])
```

La profundidad de un *quadtree* puede definirse como:

```
profundidad :: QT a → Int
profundidad = foldQT
   (\a b c d → 1 + maximum [a,b,c,d])
   (\_ → 1)
```

La función *foldQT* pertenece al conjunto de funciones que recorren y transforman una estructura recursiva en un valor. Estas funciones se denominan también *catamorfismos* y son estudiadas en el campo de la programación genérica [2].

8. No determinismo: Generación de *quadtrees* en programación lógica

Una de las dificultades de la asignatura es la introducción de los paradigmas funcional y lógico en un breve espacio de tiempo. En la actualidad se emplean, además, dos lenguajes diferentes: *Haskell* y *Prolog*. Para facilitar el cambio entre paradigmas y lenguajes, se ha optado por solicitar prácticas de programación muy similares de forma que los alumnos puedan observar más claramente las diferencias entre ambos.

De esta forma, los primeros trabajos prácticos en la parte de programación lógica vuelven a examinar sobre el tema de los *quadtrees.*

Una característica sobresaliente de la programación lógica es la posibilidad de definir predicados no-deterministas que admiten varias soluciones obtenidas por *backtracking.*

Es posible definir el predicado `col(Xs,Q1,Q2)` que se cumple cuando `Q2` es el *quadtree* formado por rellenar el *quadtree* `Q1` con los colores de la lista `Xs`.

```
col(Xs,b(_),b(X)):-elem(X,Xs).
col(Xs,d(A,B,C,D),d(E,F,G,H)):-
   col(Xs,A,E),col(Xs,B,F),
   col(Xs,C,G),col(Xs,D,H).
```

donde `elem(X,Xs)` es un predicado predefinido que se cumple si X es un elemento de la lista Xs. Obsérvese que al rellenar un quadtree con dos colores se obtienen varias respuestas.

```
?-col([0,1],d(b(_),b(_),b(_),b(_)),V).
V = d(b(0),b(0),b(0),b(0)) ;
V = d(b(0),b(0),b(0),b(1)) ;
V = d(b(0),b(0),b(1),b(0)) ;
V = d(b(0),b(0),b(1),b(1)) ;
. . .
```

Un problema clásico en el campo algorítmico es el de colorear un mapa de regiones con una serie de colores de forma que ninguna región adyacente tenga el mismo color. El problema puede plantearse para colorear *quadtrees.* En la figura 7 se presenta una posible solución al colorear un *quadtree* que representa un rombo.

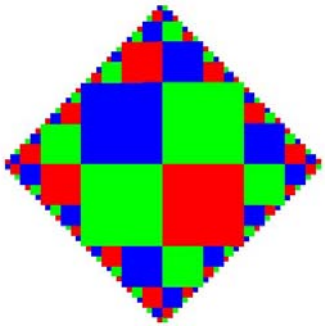


Figura 7. Solución del problema de coloreado

Una solución directa utilizando programación lógica consiste simplemente en generar todos los posibles quadtrees y comprobar la condición de no adyacencia mediante el predicado `noColor`:

```
noColor(b(_)).
noColor(d(A,B,C,D)):-
  noColor(A), noColor(B),
  noColor(C), noColor(D),
  right(A,Ar),left(B,Bl),diff(Ar,Bl),
  right(C,Cr),left(D,Dl),diff(Cr,Dl),
  down(A,Ad),up(C,Cu),diff(Ad,Cu),
  down(B,Bd),up(D,Du),diff(Bd,Du).

up(b(X),l(X)).
up(d(A,B,_,_),f(X,Y)):-up(A,X),
                       up(B,Y).

down(b(X),l(X)).
down(d(_,_,C,D),f(X,Y)):-down(C,X),
                         down(D,Y).

left(b(X),l(X)).
left(d(A,_,C,_),f(X,Y)):-left(A,X),
                         left(C,Y).

right(b(X),l(X)).
right(d(_,B,_,D),f(X,Y)):-right(B,X),
                          right(D,Y).

diff(l(X),l(Y)):-X\=Y.
diff(l(X),f(A,B)):notElem(X,A),
                  notElem(X,B).
diff(f(A,B),l(X)):-notElem(X,A),
                   notElem(X,B).
diff(f(A,B),f(C,D)):-diff(A,C),
                     diff(B,D).

notElem(X,l(Y)):-X\=Y.
notElem(X,f(A,B)):-notElem(X,A),
                   notElem(X,B).
```

9. Programación con restricciones

Los lenguajes declarativos ofrecen un marco ideal para la programación basada en restricciones

(*constraint programming*). Aunque una presentación en profundidad se saldría del ámbito de la asignatura, hemos considerado interesante presentar a los estudiantes una breve introducción a esta disciplina ya que para muchos de ellos, será el único acercamiento en toda su carrera universitaria. Continuando con el tema de los *quadtrees,* se presenta una solución del problema de colorear un *quadtree* utilizando las extensiones de programación lógica con dominios finitos soportadas por algunos sistemas Prolog. En concreto, a continuación se presenta un predicado `colC` que rellena un *quadtree* con los valores tomados de un dominio [M,N]. Se utiliza la sintaxis de CLP(FD) implementado en GNU Prolog [5].

```
colC(M,N,b(_),b(X)):-fd_domain(X,M,N).
colC(M,N,d(A,B,C,D),d(E,F,G,H)):-
   col(M,N,A,E),col(M,N,B,F),
   col(M,N,C,G),col(M,N,D,H).
```

La nueva versión de noColor utilizando restricciones sería:

```
noColorC(b(_)).
noColorC(d(A,B,C,D)):-
  noColorC(A), noColorC(B),
  noColorC(C), noColorC(D),
  right(A,Ar),left(B,Bl),diffC(Ar,Bl),
  right(C,Cr),left(D,Dl),diffC(Cr,Dl),
  down(A,Ad),up(C,Cu),diffC(Ad,Cu),
  down(B,Bd),up(D,Du),diffC(Bd,Du).

diffC(l(X),l(Y)):-X #\= Y.
diffC(l(X),f(A,B)):-notElemC(X,A),
                    notElemC(X,B).
diffC(f(A,B),l(X)):-notElemC(X,A),
                    notElemC(X,B).
diffC(f(A,B),f(C,D)):-diffC(A,C),
                      diffC(B,D).

notElemC(X,l(Y)):-X #\= Y.
notElemC(X,f(A,B)):-notElemC(X,A),
                    notElemC(X,B).
```

Obsérvese que la implementación es prácticamente igual a la presentada en la sección anterior ya que solamente se ha sustituido el predicado '`\=`' por el predicado '`#\=`'. Sin embargo, el algoritmo basado en restricciones obtiene una solución para N=3 en 0.01sg mientras que el algoritmo de la sección anterior tardaba 74.5sg.

10. Trabajo relacionado

La exigua utilización de los lenguajes declarativos [20] ha llevado a varios miembros de esta comunidad a la búsqueda de aplicaciones atractivas que resalten las capacidades de este tipo de lenguajes. Cabe destacar el libro de texto escrito por P. Hudak [10] en el que se presenta una introducción a la programación funcional incluyendo ejemplo relacionados con sistemas multimedia. En el libro se utilizan varias librerías específicas que permiten generar y visualizar los diversos ejercicios propuestos. La estrategia seguida en este artículo es similar en el objetivo, pero difiere en la forma de visualizar las construcciones gráficas. Se ha optado por utilizar vocabularios XML que se están convirtiendo en estándares en sus respectivos campos. Esta técnica supone varias ventajas: existencia de mayor número de herramientas de visualización, portabilidad entre plataformas e independencia de librerías específicas. Además, los estudiantes emplean tecnologías XML estándar que pueden ser beneficiosas en otros campos de su trayectoria profesional.

Las representaciones declarativas de *quadtrees* fueron estudiadas en [4,8,19,21]. Recientemente, C. Okasaki [16] toma como punto de partida la representación en Haskell de un *quadtree* para definir una implementación eficiente de matrices cuadradas mediante tipos anidados. Su implementación mantiene la consistencia de la representación gracias al sistema de tipos de Haskell.

En el campo imperativo existen varios trabajos [7,11,13] que resaltan la utilización de *quadtrees* como buenos ejemplos de prácticas de programación, centrándose fundamentalmente en su aplicación para comprimir imágenes. En [6] se describen diversos algoritmos de coloreado de *quadtrees* y su aplicación práctica en la planificación de computaciones paralelas.

11. Conclusión y Líneas de trabajo

El presente artículo propone un esquema de trabajos prácticos que pretende potenciar la visualización gráfica de los resultados a la vez que se exploran las diversas características de los lenguajes declarativos. Aunque no se ha realizado un estudio sistemático de la reacción de los estudiantes ante este esquema, puede afirmarse que las primeras impresiones son altamente positivas, con un porcentaje de abandono de la asignatura notablemente inferior al de cursos anteriores. Sin embargo, este tipo de afirmaciones debería contrastarse de una forma rigurosa comprobando, por ejemplo, que los estudiantes expuestos a este tipo de enseñanza realmente resuelven mejor otros problemas de programación.

La generación de mundos virtuales ha supuesto un aliciente en la investigación del grupo IDEFIX [14,15]. Entre las futuras líneas de investigación se encuentra el estudio de comunidades virtuales tipo *ActiveWorlds* [1] donde los estudiantes puedan visitar la comunidad, crear sus propios mundos y conversar con otros estudiantes mediante *chat.*

Referencias

[1] ActiveWorlds, página Web: http://www.activeworlds.com

[2] R. Backhouse, P. Jansson, J. Jeuring, L. Meertens, *Generic Programming – An Introduction,* Advanced Functional Programming, Lecture Notes in Computer Science, vol 1608, S. Swierstra, P. Henriques, J. N. Oliveira (Eds), 1999

[3] T. Bray, J. Paoli, C. M. Sperberg-McQueen, Extensible markup language (1.0), http://www.w3.org/TR/REC-xml, Oct 2000

[4] F. W. Burton, J. G. Kollias. *Functional programming with quadtrees.* IEEE Software, 6(1):90-97, Enero, 1989

[5] P. Codognet and D. Diaz. Compiling Constraints in CLP(FD). *Journal of Logic Programming*, Vol. 27, No. 3, June 1996

[6] D. Eppstein, M. W. Bern, B. Hutchings, *Algorithms for coloring quadtrees,* Algorithmica, 32(1), Ene. *2002*

[7] J. B. Fenwick Jr., C. Norris, J. Wilkes, *Scientific Experimentation via the Matching Game,*SIGCSE Bulletin 34(1), 33th SIGCSE Technical Symposium on Computer Science Education, Marzo, 2002

[8] J. D. Frens, D. S. Wise, Matrix inversion using quadtrees implemented in gofer. Technical Report 433, Computer Science Department, Indiana University, Mayo 1995

[9] J. Good, P. Brna, *Novice Difficulties with recursion: Do graphical Representations Hold the solution?*, European Conference on Artificial Intelligence in Education, Lisboa, Portugal, Oct.,1996

[10] P. Hudak, *The Haskell School of Expression: Learning Functional Programming through Multimedia,* Cambridge Univ. Press, 2000

[11] R. Jiménez-Peris, S. Khuri, M. Patiño-Martínez, *Adding breadth to CS1 and CS2 courses through visual and interactive programming projects,* ACM SIGCSE Bulletin 31(1), Marzo, 1999

[12] J. Kaasbøll, *Exploring didactic models for programming,* Norsk Informatikk-Konferanse, Høgskolen I Agder, 1998

[13] S. Khuri, H. Hsu, *Interactive Packages for learning image compression algorithms*, ACM SIGCSE Bulletin 32(3), Sept. 2000

[14] J.E. Labra, J.M. Morales, R. Turrado, *Plataforma de enseñanza de lenguajes de programación a través de Internet: Proyecto Idefix,* Jornadas de Enseñanza Universitaria de Informática, JENUI-2002, Cáceres, Jun. 2002

[15] J.E. Labra, J. M. Morales, A. M. Fernández, H. Sagastegui, *A Generic e-Learning Multiparadigm Programming Language System: IDEFIX Project*, ACM 34th SIGCSE Technical Symposium on Computer Science Education, Reno, Nevada, USA, Febrero 2003

[16] Chris Okasaki, *From Fast exponentiation to Square Matrices: An adventure in Types*, ACM SIGPLAN Notices 34(9), Intl. Conference on Functional Programming, pp. 28-35, 1999

[17] Hanan Samet *The quadtree and related hierarchical data structures.* ACM Computing Surveys, 16(2): 187-260, Junio 1984.

[18] J.Segal, *Empirical studies of functional programming learners evaluating recursive functions,* Instructional Science 22, 385-411, 1995

[19] S. Edelman and E. Shapiro, *Quadtrees in concurrent prolog*, Proc. Intl. Conference on Parallel Processing, 1985, pp. 544-551

[20] P. Wadler, *Why no one uses functional programming languages?,* ACM SIGPLAN Notices 33(8): 23-27, Agosto, 1998

[21] D. Wise, *Matrix algorithms using quadtrees.* Technical Report 357, Computer Science Department, Indiana University, Jun., 1992

Notación formalizada para la representación de árboles de seguimiento de algoritmos en Prolog

Nieves Pavón Pulido Omar Sánchez Pérez
Departamento de Ingeniería Electrónica Sistemas Informáticos y Automática
Palos de la Frontera, Huelva.
Universidad de Huelva
email: npavon@diesia.uhu.es
omar@diesia.uhu.es

Resumen

En el siguiente artículo se presenta un método de diseño y representación formal de árboles de ejecución de programas lógicos escritos en Prolog. Este método se ha aplicado con éxito en el desarrollo de ejercicios en una asignatura de programación declarativa que se imparte en tercer curso de Ingeniería Técnica en Informática de Gestión en la Universidad de Huelva. Además, se ha desarrollado una herramienta que permite generar automáticamente estos árboles de ejecución con la notación formalizada para la comprobación sencilla de ejercicios que consistan en un seguimiento del funcionamiento de código fuente escrito en Prolog.

1. Introducción

En la Universidad de Huelva, los alumnos de Ingeniería Técnica en Informática de Gestión se enfrentan al paradigma declarativo lógico en tercer curso [1].

Acostumbrados a las técnicas de programación imperativa, la comprensión de las técnicas declarativas lógicas de programación resultan verdaderamente complicadas.

Cuando se enseña una metodología de programación pueden seguirse los siguientes pasos:

- Descripción de aspectos teóricos.
- Descripción de cómo realizar el diseño del código fuente.
- Descripción de cómo realizar el seguimiento y depuración del código fuente.

El primer paso es, obviamente, el más sencillo. En el segundo paso se adquiere destreza cuando se han realizado muchos ejemplos y ejercicios. Por tanto, se puede asegurar que la realización de un número elevado de ejercicios es fundamental para comprender una técnica de programación. Sin embargo, el segundo paso no está completo si para cada programa diseñado no aplicamos el conjunto de operaciones que se exponen en el tercer paso, es decir, el seguimiento detallado del conjunto de instrucciones implementado.

En un lenguaje imperativo, el seguimiento de un programa consiste en la escritura del resultado de un conjunto de instrucciones secuenciales o encapsuladas en estructuras de control de sobra conocidas como *"repetir"*, *"si"*, *"caso de"*, etc., para las cuales ya existen técnicas de descripción y representación formal, sin embargo, para los lenguajes declarativos lógicos el proceso de seguimiento no es trivial, pues consiste en la generación de un árbol de ejecución cuyos nodos son originados por los procesos de recursión y backtracking intrínsecos a la implementación de este tipo de programas.

En los puntos siguientes del artículo se muestra un método para formalizar la representación de estos seguimientos, de modo que a los alumnos de la asignatura de Programación Declarativa les resulte sencillo realizarlos y comparar los resultados obtenidos con los expuestos por el profesor.

2. Descripción de elementos de un programa Prolog.

Un programa en Prolog consta de varios predicados que se implementan mediante un conjunto de cláusulas de tipo "*hecho*" o "*regla de inferencia*".

Los hechos se pueden definir como estructuras de la forma:

nombreclausula(arg1, arg2, ... , argN).

Donde "*nombreclausula*" describe la relación que existe entre los argumentos *arg1*, *arg2*, ... , *argN*.

Las reglas de inferencia se pueden definir como estructuras de la forma:

nombreclausula(arg1, arg2, ... , argN):-
c1(args), c2(args), ... ,cN(args).

Donde "*nombreclausula*" describe la relación que existe entre los argumentos *arg1*, *arg2*, ... , *argN*, y las cláusulas de la parte derecha del predicado deben ser ejecutadas (demostradas), para obtener el resultado de la estructura "*nombreclausula*".

La ejecución de un programa Prolog se basa en la realización de dos procesos diferenciados pero fuertemente relacionados entre sí [3][4]:

- *Unificación*: Se intenta demostrar que un objetivo planteado es exactamente igual a un hecho o a la parte izquierda de una regla de inferencia, en todos sus componentes. En el caso de una regla de inferencia, además de que el objetivo planteado se unifique con la parte izquierda de la regla, es necesario que todas las llamadas a las cláusulas situadas a la derecha de la regla puedan ser unificadas.
- *Backtracking*: Si no es posible unificar un objetivo con una cláusula de tipo hecho o parte izquierda de una regla de inferencia, el sistema, automáticamente, intenta buscar la solución con la cláusula siguiente del conjunto que implementa un determinado predicado. Si la llamada a una cláusula de la parte derecha de una regla de inferencia no se puede demostrar, se intenta ejecutar de nuevo la llamada a la cláusula inmediatamente anterior a la misma, de modo que la nueva solución obtenida permita satisfacer la llamada a la cláusula que no pudo ser demostrada.

El modo de ejecución de un programa Prolog genera, no un conjunto secuencial de resultados de operaciones, sino un árbol de intentos de operaciones de unificación [5]. Cuando un objetivo se puede demostrar con varias cláusulas, se fija lo que se denomina punto de backtracking, es decir, un lugar al que podemos regresar para buscar una solución alternativa en el caso de que la cláusula que se ejecuta posteriormente no pueda ser demostrada. Se observa, además, que en Prolog no se dispone de estructuras de control, tales como *"repetir"* o *"si"*, entre otras. El control lo lleva el propio intérprete del código. El único modo de *controlar* dicho control es mediante dos predicados: !(*predicado corte*) y *fail*.

El primero no produce ninguna operación visible para el usuario salvo que elimina la posibilidad de hacer backtracking con alguna cláusula anterior (borrado del punto de backtracking más próximo). El segundo produce un resultado falso (no demostrado), es decir, que falla siempre.

Para comprender todo lo expuesto anteriormente, veamos el siguiente bloque de código que implementa dos relaciones de parentesco: "*padre*" y "*abuelo paterno*":

```
padre("juan", "pedro").
padre("pedro", "pepe").
abuelo(X,Y):-
padre(X,Z),padre(Z,Y).
```

El programa implementa el siguiente conocimiento:

- *Juan es padre de Pedro y éste, a su vez, es padre de Pepe.*
- Por otro lado, la relación abuelo se define mediante la demostración de la sentencia siguiente: *si X es padre de Z y Z es padre de Y entonces X es abuelo de Y.*

Si se especifica como objetivo la cláusula:

```
abuelo("juan", Y).
```

Se intenta que el programa Prolog obtenga como resultado en la variable *Y* al nieto de "*juan*", es decir, *Y="pepe"*.

Si se especifica como objetivo la cláusula:

```
abuelo("juan", "alvaro").
```

El programa Prolog da como resultado *no*, ya que no es posible encontrar ningún nieto de *Juan* que se llame *Álvaro*.

Si se utilizase este ejemplo como ejercicio en la asignatura de Programación Declarativa no tendría valor pedagógico ya que es posible obtener la solución razonadamente sin llevar a cabo el seguimiento del código. Por tanto, es necesario

que el alumno, no sólo exponga la solución final del programa, sino que también escriba el árbol completo de ejecución del mismo. Sin embargo, si estamos realizando un ejercicio tipo test [2], la única pregunta que parece razonable es, simplemente, *¿cuál es el valor final en la variable Y?*

Puede suceder que si se pide el desarrollo del árbol en un ejercicio no de tipo test el alumno realice su propia versión, que aunque correcta, puede diferir mucho de la del profesor, con lo que al profesor puede resultarle difícil comprender la notación establecida por el alumno y viceversa.

De ahí la necesidad de establecer una notación formal que solucione la problemática expuesta.

3. Notación formal para construir árboles de seguimiento

En el seguimiento de un programa en Prolog nos interesa conocer la solución final del algoritmo y comprender como se ha conseguido dicha solución.

Por tanto, en un árbol que represente el seguimiento de un programa Prolog deben aparecer los siguientes elementos:

1. Caja de texto para la especificación de la cláusula que se está ejecutando. (En el caso de que se trate de una regla de inferencia provocará el procesamiento de varios subobjetivos que quedarán representados mediante cajas).
2. Líneas de seguimiento hacia adelante.
3. Líneas de seguimiento del retroceso cuando se hace backtracking.
4. Líneas de valores devueltos por los procesos de backtracking o recursión.
5. Puntos de backtracking activos.
6. Puntos de backtracking borrados.

La figura 1 muestra el elemento descrito en el punto 1.

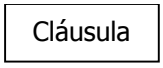


Figura 1. Representación de la ejecución de una cláusula

La figura 2 muestra los tres tipos de líneas que podemos encontrar, ya descritas en los puntos 2, 3 y 4.

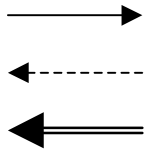


Figura 2. Representación de líneas de flujo de la información

La figura 3 muestra un punto de backtracking activo y un punto de backtracking borrado [5].

Figura 3. Representación de puntos de backtracking

Si se aplica el diseño formalizado al ejemplo descrito en el apartado anterior para el caso de la meta:

```
abuelo("juan", Y).
```

Queda el árbol de seguimiento que se muestra en la figura 4.

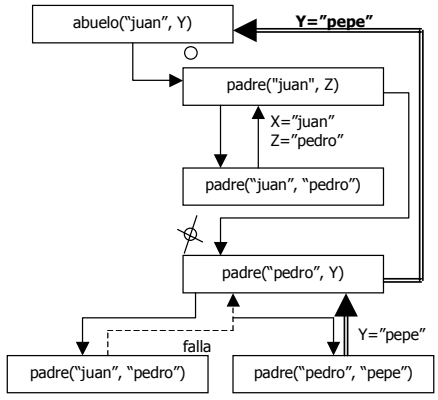


Figura 4. Árbol formalizado de seguimiento de un programa Prolog

Puede observarse que cuando se ejecuta la cláusula *abuelo("juan", Y)*, se generan dos subobjetivos que se han de satisfacer secuencialmente:

padre(X,Z)
padre(Z,Y)

Como *padre(X,Z)* puede resolverse mediante dos cláusulas, se ha de establecer un punto de backtracking. La solución es hallada con la utilización de la primera cláusula, luego el punto de backtracking permanece activo por si se necesita más adelante.

Tras la ejecución del primer subobjetivo planteado, es decir, *padre(X,Z)*, es necesario satisfacer el segundo subobjetivo *padre(Z,Y)*. Como en el caso anterior, se fija un punto de backtracking ya que el subobjetivo puede ser solucionado mediante dos cláusulas posibles. La primera de ellas falla, luego se utiliza el punto de backtracking y se intenta con la segunda y última cláusula. En este caso no quedan más cláusulas para nuevos intentos, por lo que el punto de backtracking se elimina. El subobjetivo queda satisfecho mediante el uso de esta segunda cláusula, por lo que el resultado final, almacenado en Y, se eleva hacia la raíz del árbol.

A través de este ejemplo podemos observar que debido a que en el gráfico se usa una notación formal que comparten tanto los alumnos como el profesor es fácil para ambos comparar resultados, realizar preguntas y especificar respuestas acerca del mismo.

A continuación se muestra un conjunto de preguntas tipo test que se puede elaborar teniendo en cuenta la representación formal del árbol de ejecución:

- ¿Cuántos puntos de backtracking se han generado en total durante el proceso de ejecución? Respuesta: 2 puntos de backtracking.
- ¿Cuántos puntos de backtracking se borran en total durante el proceso de ejecución? Respuesta: 1 punto de backtracking.
- ¿Cuántas veces se evalúa alguna de las cláusulas que componen el predicado padre? Respuesta: 3 veces.

La utilización de este método en la evaluación de esta clase de ejercicios en un examen tipo test revela los conocimientos del alumno de forma más exhaustiva que la simple elección del valor final devuelto por el programa que, sin duda, resulta evidente con un sencillo razonamiento, sin necesidad de llevar a cabo la ejecución del algoritmo.

Si el ejercicio se utiliza en un examen no de tipo test, donde se pide al alumno especificar manualmente el árbol, la notación formalizada proporciona al profesor un alto grado de comodidad en el momento de la corrección.

Además, en el proceso de construcción del árbol, el alumno nota como el intérprete Prolog lleva el control del algoritmo y puede considerar de una forma más inteligente el modo de optimizar el código mediante el uso del predicado corte o, simplemente, evitar errores de ejecución por el uso de puntos de backtracking que sobran.

La figura 5 muestra un árbol de backtracking que permite realizar el seguimiento del algoritmo definido a continuación.

```
fact(0,1).
fact(N,R):-
N1=N-1, fact(N1,R1), R=N*R1.
prueba(N,R):-fact(N,R), R=5.
```

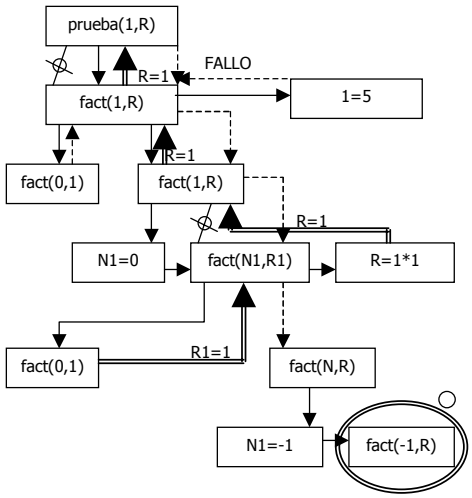


Figura 5. Árbol de ejecución para la meta prueba(1,R)

Como puede observarse, cuando *R=5* falla, regresamos al punto de backtracking que queda activo para buscar una nueva solución del factorial. Sin embargo, esta estrategia de control es errónea ya que el factorial es un problema determinista que genera una solución única para cada objetivo. Si se intenta buscar una nueva solución para el factorial, el algoritmo entra en un proceso de recursión infinito divergente del caso trivial. El alumno puede darse cuenta, a través del gráfico, de que es necesario eliminar el punto de backtracking que queda activo tras la demostración de la cláusula *fact(0,1)*. La solución, por tanto, consiste en modificar la cláusula:

```
fact(0,1).
```

sustituyéndola por:

```
fact(0,1):-!.
```

La figura 6 muestra el árbol de ejecución cuando se elimina el punto de backtracking conflictivo mediante la llamada al predicado corte.

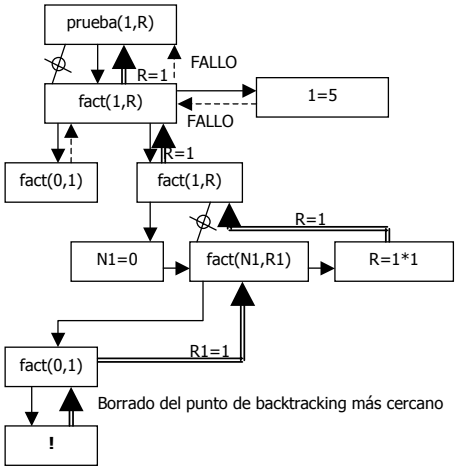


Figura 6. Árbol de ejecución para mostrar el resultado de aplicar el corte

La tarea de escribir correctamente los cortes es difícil ya que, en muchas ocasiones, no resulta obvia. Puede observarse que el gráfico es de gran ayuda en esta circunstancia.

4. Simulador de árboles de seguimiento

El método de representación formal de árboles de ejecución expuesto en los puntos anteriores se ha utilizado con éxito durante los dos últimos cursos impartidos.

Sin embargo, dicho método no resulta completo aún, puesto que el alumno no dispone de una herramienta de generación automática de árboles que le permita comparar su solución con la solución correcta para un determinado ejercicio, cuando el profesor no está presente para llevar a cabo esta corrección de forma personal.

Para solucionar este problema se ha implementado una herramienta software que genera el árbol de ejecución de un algoritmo escrito en Prolog.

El simulador ha sido construido por un alumno de Proyecto Fin de Carrera que, previamente, estudió la asignatura en tercer curso de carrera.

4.1. Características generales del simulador

El simulador es un programa que funciona en las siguientes plataformas: Windows 9X, Millenium y XP.

La aplicación consta de:

- Un editor de texto que permite la escritura de un algoritmo sencillo en Prolog.
- Un corrector de la sintaxis del algoritmo escrito.
- Una ventana para definir la meta u objetivo que se desea satisfacer.
- Una ventana para mostrar el árbol generado, con posibilidad de realizar operaciones para guardarlo o imprimirlo.

La figura 7 muestra el aspecto de la ventana principal de la aplicación. En el editor de texto está escrito el algoritmo expuesto en el apartado 3.

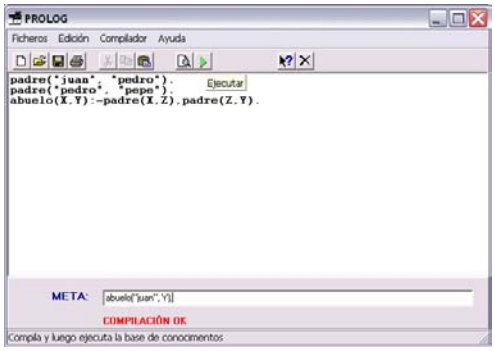


Figura 7. Pantalla principal del simulador

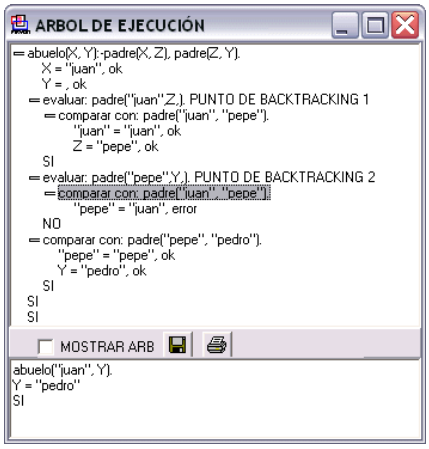


Figura 8. Diálogo que muestra un árbol de ejecución

La figura 8 muestra el diálogo que permite la visualización y las operaciones de "guardar" e "imprimir" el árbol de ejecución obtenido para el algoritmo descrito en el apartado 3.

Puede observarse, además, que el simulador permite operaciones de expansión y contracción de los nodos del árbol para facilitar su estudio.

4.2. Fase de implantación del simulador en la asignatura

Dado que el programa de generación de árboles de ejecución ha sido fruto del desarrollo que un alumno ha realizado durante su proyecto fin de carrera aún no ha sido posible implantar el sistema en la asignatura de Programación Declarativa.

Se están planeando dos formas de introducir el simulador en la asignatura:

- En las horas de prácticas, como una herramienta más a utilizar en el diseño e implementación de las soluciones incorporadas a las hojas de problemas prácticos a resolver.
- En las horas de teoría, como apoyo en la corrección de ejercicios. Para ello, sería conveniente que el profesor utilizase un ordenador y un cañón de vídeo para impartir sus clases, de modo que, para ejercicios propuestos por él mismo, o para dudas planteadas por los alumnos, pueda utilizar el simulador durante la clase teórica proyectando los resultados en tiempo real a modo de transparencia.

5. Conclusiones

La principal conclusión que se obtiene de todo lo expuesto anteriormente es que el uso de una estrategia formal de representación de árboles de ejecución de algoritmos en Prolog supone una mejora en la calidad de la docencia de la asignatura de Programación Declarativa.

Esta mejora se obtiene a partir de las ventajas que conlleva el uso de la notación formalizada:

- Permite una mayor versatilidad a la hora de proponer ejercicios para exámenes tipo test.
- Unifica el método de representación de árboles de seguimiento, de modo que profesor y alumnos utilizan unas normas comunes en la escritura de estos árboles, facilitando la compresión de las soluciones aportadas para los ejercicios propuestos.
- Facilita la comprensión de predicados como el *corte* y el *fail*.
- Permite al alumno conocer de forma exhaustiva el método de ejecución de un algoritmo escrito en Prolog.

Se ha comprobado experimentalmente que, desde que se está usando este método, los alumnos comprenden mejor el funcionamiento del intérprete Prolog.

6. Líneas Futuras

Las líneas futuras están encaminadas hacia la mejora de la aplicación de generación automática de árboles de seguimiento.

Se pretende que, en el curso que viene, la herramienta se encuentre disponible para su uso, principalmente, en las sesiones teóricas de la asignatura.

Agradecimientos

Al alumno Antonio Villarán Carrellán, autor de la aplicación de generación automática de árboles de ejecución de algoritmos en Prolog, por el eficiente trabajo realizado para su proyecto fin de carrera.

Referencias

[1] Pavón, N. *Actas de las VII Jornadas de Enseñanza Universitaria de Informática*, JENUI 2001, pp. 329-334, Palma de Mallorca, 2001.

[2] Pavón, N. et all. *Actas de las VIII Jornadas de Enseñanza Universitaria de Informática*, JENUI 2001, pp. 231-236, Cáceres, 2002.

[3] Sterling & Shapiro. *The Art of Prolog*. MIT Press, 1994.

[4] Adarraga & Zaccagnini. *Psicología e Inteligencia Artificial*. Editorial Trotta, 1994.

[5] Prolog Development Center A/S. *Language Tutorial. Visual Prolog version 5.X*. Copenhagen, Denmark.

Entorno web de desarrollo para el aprendizaje de paradigmas de programación

Juan Ramón Pérez Pérez, Mª del Puerto Paule Ruiz, Martín González Rodríguez, Ramón González Suárez
HCI-RG. Departamento de Informática,
Universidad de Oviedo
C/ Calvo Sotelo S/N 33007 – Oviedo - España
e-mail: {jrpp, paule}@pinon.ccu.uniovi.es, martín@lsi.uniovi.es, i9440259@petra.euitio.uniovi.es

Resumen

En este artículo, se propone un entorno orientado a la enseñanza de lenguajes de programación pensado para entornos de enseñanza virtuales. Se considera que la enseñanza de lenguajes de programación ha de tener una parte en la que el alumno realice prácticas de programación, esto plantea dificultades extra en entornos de enseñanza virtuales convencionales. Para superar estas dificultades, planteamos un entorno que recibe la denominación de DWE (Development Web Environment), en el que se integra el compilador del lenguaje en un entorno Web. Junto con este compilador se incluyen distintos módulos de apoyo para ayudar al alumno mientras realiza los programas, utilizando para ello la experiencia de todos los usuarios del grupo, facilitando la comunicación alumno – profesor y alumno – alumno. Por último, se propone un módulo que permite al profesor realizar el seguimiento de los alumnos, conocer sus puntos fuertes y débiles y orientar la enseñanza para superar estos puntos débiles.

1. Introducción

El aprendizaje de un paradigma de programación y su aplicación en un lenguaje concreto no es una tarea trivial. Son muchos los autores que abordan las dificultades cognitivas que plantea el aprendizaje de un nuevo paradigma o el cambio entre paradigmas; incluso cuando se supone que el paradigma conlleva procesos mentales más "intuitivos". Este es el caso del cambio desde la programación estructurada a un paradigma orientado a objetos [10].

La asimilación y puesta en práctica de los conceptos que conlleva un paradigma de programación requiere un importante esfuerzo para lograr su asimilación por parte de alumno; por otra parte, para una correcta comprensión es imprescindible combinar el trabajo de conceptos teóricos con la puesta en práctica de estos utilizando un lenguaje de programación concreto. Esto ayuda al alumno a consolidar todo aquello aprendido en teoría y a elevar su nivel de conocimiento. Si nos fijamos en la taxonomía del conocimiento de Bloom, es en la práctica donde el alumno maneja más objetivos de aplicación y análisis y donde puede descubrir errores en el ámbito de comprensión.

Este proceso, para que sea efectivo, requiere un seguimiento y un apoyo continuo por parte del profesor. Esta tarea de tutorización es casi imprescindible en las fases iniciales de aprendizaje. Sin esta tutorización, un alumno tendría dificultades para aplicar los conocimientos obtenidos en teoría. Estos problemas van desde la configuración de determinadas características del compilador, hasta la interpretación de los errores de compilación y que pasos hay que llevar a cabo para corregir un determinado tipo de error. Además de esto, el profesor suele proporcionar patrones de programación que permiten evitar errores y relacionar conceptos de teoría que el alumno no ha comprendido correctamente y se reflejan en errores en los programas.

Todo esto conlleva, una constante interacción entre alumno – profesor que después de un tiempo le proporcionará un conocimiento adecuado del lenguaje de programación y de la herramienta de compilación, lo que permitirá al alumno aplicar determinadas técnicas para evitar que se

produzcan errores y para solucionarlos si se producen. En entornos de enseñanza presencial, esta interacción se produce de forma más o menos fluida. Incluso habría que tener en cuenta el factor de la comunicación alumno – alumno.

Este modelo de enseñanza de la programación es difícilmente transferible a los entornos de enseñanza virtual convencionales. Por definición, en estos entornos el alumno sólo puede establecer cauces de comunicación alumno – profesor y alumno – alumno a través del propio entorno. Normalmente estos entornos disponen de herramientas más o menos específicas; pero que se basan en el correo electrónico, los grupos de discusión y chats; con estos medios el alumno tiene dificultades para transmitir los problemas de programación con los que se encuentra y, por tanto, también será más difícil para el profesor ayudarle a resolveros.

2. Herramientas y problemas para la enseñanza virtual

Actualmente existe una gran proliferación de entornos de enseñanza virtual que utilizan Internet como medio para transmitir los contenidos de una determinada asignatura, complementando la exposición teórica con ejercicios que suelen ser del tipo de preguntas de respuesta cerrada.

Existen en la actualidad varias herramientas que permiten la construcción y la impartición de este tipo de cursos [4]. En concreto, nuestra universidad esta utilizando WebCT [3] para el desarrollo de este tipo de cursos dentro de un proyecto denominado AulaNet [2] dentro un campus virtual integrado por las universidades del grupo G7.

Estos entornos permiten al alumno navegar, siguiendo ciertas pautas, a través de los temas teóricos, hacer evaluaciones del alumno y ofrecen herramientas que permiten la comunicación entre alumno y profesor o entre varios alumnos basadas en el correo electrónico, los chats o los grupos de discusión.

Estos entornos no son suficientes cuando se están realizando prácticas de lenguajes de programación. Al alumno le resulta difícil expresar las dificultades que tiene con un fragmento de código; muchas veces los alumnos se refieren al error; pero el profesor no tiene el contexto en el que se produce para poder dar una respuesta; con frecuencia los alumnos incluyen el propio fragmento de código en el mensaje, y el profesor, para poder responder, tiene que examinar el fragmento detenidamente o incluso compilarlo, lo cual no siempre es posible. Esto también provoca una pérdida de tiempo. Todo esto conlleva ambigüedades y falta de precisión que dificultan la solución de problemas.

En la actualidad hay herramientas que permiten la corrección automática de prácticas de programación [5,6] que permiten ahorrar tiempo al profesor a la hora de evaluar los conocimientos del alumno e incluso hay planteamientos [9] para que estas herramientas puedan dar información al alumno del tipo de errores que comete y que conceptos tiene que repasar para no cometerlos; pero no existen herramientas que permitan asesorar al alumno mientras está realizando las prácticas de programación y antes de que el alumno obtenga el resultado final de la evaluación.

Lo que planteamos en este artículo es un entorno de desarrollo basado en Internet para la enseñanza virtual del paradigma orientado a objetos utilizando el lenguaje Java, que denominamos DWE (Development Web Environment). Este entorno se realiza en relación con el proyecto IDEFIX (Integrated Development Environment based on Internet and eXtensible technologies) [1,7, 8].

3. Descripción del entorno

DWE utiliza una interfaz web, por tanto el alumno sólo necesita un navegador para acceder a la aplicación, y no deberá instalar nada en su equipo local. Esta característica proporciona dos importantes ventajas que son imprescindibles para integrarse en un entorno virtual de enseñanza en el que el mantenimiento y la configuración son siempre a distancia y, por tanto, costosa de realizar. La primera ventaja es que tendremos un mantenimiento centralizado de la aplicación, cuando se modifica cualquier componente de la aplicación sólo es necesario modificarlo en el servidor y no es necesario modificar nada en el ordenador del usuario final. Como segunda ventaja, el usuario final no tiene que realizar ningún proceso de instalación local del sistema, evitando posibles problemas de compatibilidad y / o configuración del ordenador del usuario final.

Para poder utilizar esta aplicación el alumno debe estar registrado previamente y debe introducir su nombre de usuario en el sistema y su contraseña. Esto permite a la aplicación identificar al usuario y monitorizar las operaciones que realiza sobre el sistema.

La interfaz de usuario, en el desarrollo actual de la aplicación, es una interfaz general para todos los usuarios pero podría personalizarse fácilmente para cada uno de los usuarios mediante un módulo de configuración.

DWE permite a los usuarios escribir, compilar y corregir errores de sus programas escritos en java. Para ello, como se puede ver en la figura 1, la interfaz de compilación del entorno dispone de una barra de botones desde donde se acceden a las principales funciones del gestor de ficheros con el que el cada usuario podrá introducir, desde su máquina local, nuevos ficheros fuente para compilar y, descargar otros de su directorio privado en el servidor, además de poder crear nuevos ficheros. El entorno también dispone de un área de edición de código.

Cuando el usuario compila un fichero, el sistema captura los errores que se producen, informa al usuario de los errores de su programa y guarda la información sobre los errores en una base de datos.

4. Implementación del entorno

DWE está basado en tecnologías Java del servidor: servlets y JSP. Como se dijo anteriormente, es un requisito básico el no tener que instalar nada en los ordenadores de los usuarios finales; además con tecnologías Java se puede utilizar tanto servidores con sistema operativo UNIX como servidores con sistema operativo Windows e incluso combinar los dos tipos, lo cual da una flexibilidad extra al sistema respecto a la infraestructura del sistema.

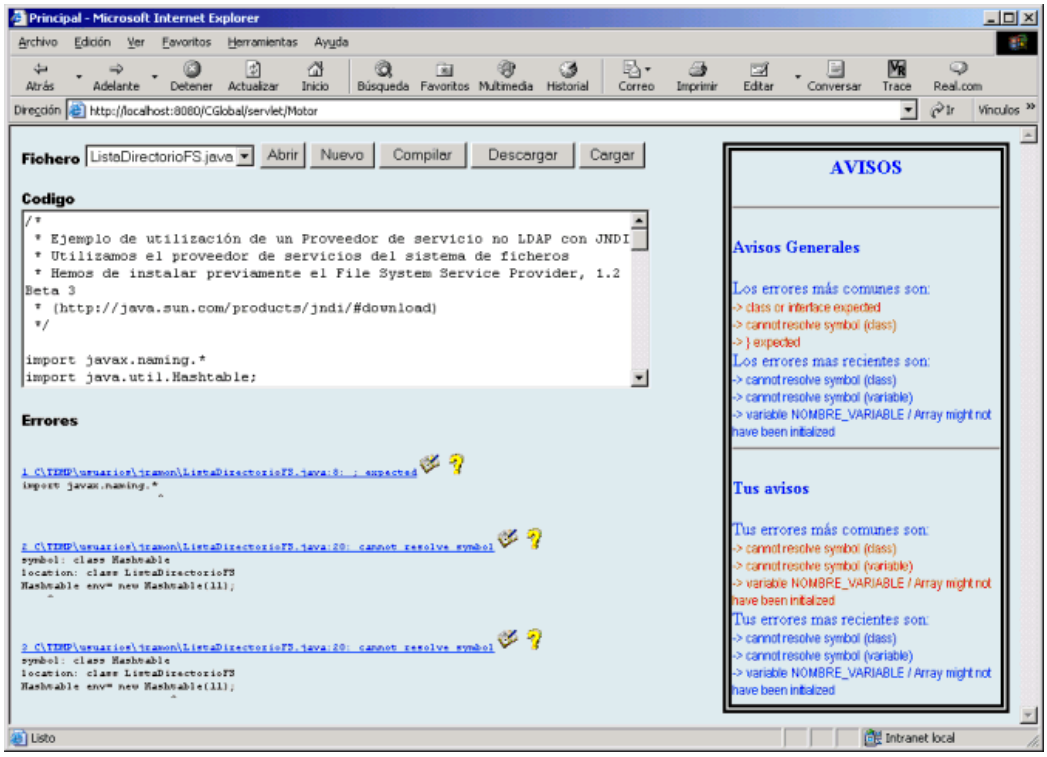


Figura 1. Interfaz del entorno DWE

Se trata de una combinación de páginas JSP, servlets y JavaBeans que separa la presentación de los demás módulos siguiendo el patrón *Modelo Vista Controlador*.

Un servlet se encarga de identificar y autenticar al usuario y crear una sesión que permite a la aplicación hacer un seguimiento de todas las acciones que lleva a cabo este usuario.

El servlet controlador se encarga de identificar el evento que genera el usuario al pulsar uno de los botones de opciones e invocar al JavaBean correspondiente que es que realiza la acción. Uno de estos JavaBeans se encargará de la gestión de la compilación y la captura de los errores. Además, existen clases asociadas a este JavaBean que almacenan estos errores en la base de datos utilizando la API JDBC de Java.

5. Arquitectura de DWE

DWE está formado por distintos módulos funcionales especializados (figura 2).

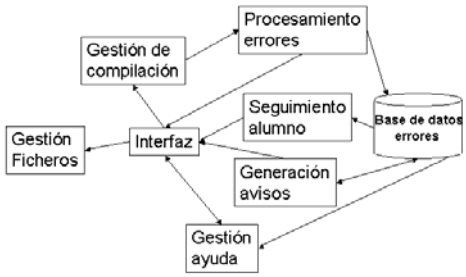


Figura 2. Módulos que constituyen DWE

Mód. Gestión de ficheros, se encarga de gestionar ficheros fuente y compilados, su organización jerárquica y posibilita el intercambio entre el servidor y el ordenador local del usuario.

Mód. Gestión de compilación, gestiona la llamada al compilador con las opciones adecuadas pasando los ficheros fuente para realizar la compilación.

Mód. Procesamiento de errores, recoge la salida del compilador, comprueba si hay errores y si los hay los almacena en la base de datos, los clasifica y actualiza los contadores correspondientes.

Gestión de ayuda, es el módulo encargado de proporcionar ayuda al usuario ante los errores. Esto módulo también se encarga de construir la ayuda con las aportaciones de profesores y alumnos.

Generación de avisos, este módulo se encarga de generar avisos para el usuario, a partir de la información almacenada en la base de datos, que le indica cuales son sus errores más frecuentes y los últimos que ha cometido.

Seguimiento del alumno, permite a los profesores de la asignatura hacer un seguimiento de los alumnos y de grupos de alumnos, que acciones han realizado los usuarios, cuantas compilaciones han hecho, los errores más frecuentes de un alumno o un grupo de alumnos, etc.

Módulo Interfaz, valida e identifica a los usuarios y les permite interactuar con los distintos módulos.

5.1. Análisis de los tipos de errores

En primer lugar, el módulo de gestión de errores debe clasificar los errores en su tipo correspondiente, cada tipo tiene una información asociada que puede ser distinta para distintos tipos. Supongamos que al compilar un fichero llamado HolaMundo.java javac genere el siguiente error:

```
HolaMundo.java:7:cannot    resolve
symbol
symbol  : class Usuario
location: class HolaMundo
Usuario user = new Usuario ();
^
```

Y ahora supongamos que otro usuario compila un fichero llamado Helloword.java y se genera el siguiente error:

```
Helloword.java:8:cannot    resolve
symbol
symbol  : class Admin
location: class Helloword
Admin ad = new    Admin ();
^
```

En este ejemplo se ve claramente que se trata del mismo tipo de error, tiene el mismo tipo de información con distintos valores. Para clasificar cada error concreto en un tipo, el módulo de procesamiento de errores se apoya en el conjunto

de palabras comunes a todos los errores de ese tipo, esto es lo que llamamos *patrón del error*. En este caso, por ejemplo, el patrón del error es, en la primera línea: "cannot resolve symbol" y en la segunda línea: "class". En la base de datos de la aplicación, se han definido patrones para distintos tipos de errores, de manera que ante cada error de compilación se realiza una búsqueda para descubrir que patrón cumple, cuando se encuentra una coincidencia se puede clasificar el error dentro de ese tipo y almacenarlo para su posterior uso en la generación de estadísticas y avisos que sirvan de ayuda al usuario.

Por otra parte, diferentes tipos de errores no sólo tienen un patrón diferente, sino que la información asociada que proporcionan sigue diferentes formatos. Se ha realizado una recopilación de los distintos tipos de errores de compilación java que se pueden producir con el fin de conocer y almacenar el formato de los mismos, para que el módulo de gestión de errores pueda extraer de forma automática los datos que se quieren almacenar en la base de datos.

5.2. Entorno colaborativo de ayuda a los errores

El módulo de gestión de ayuda, además de mostrar al usuario los errores de compilación que se produzcan, clasifica los errores, como se ha visto en el apartado anterior y almacena una serie de datos relacionados con los errores generados en la compilación de los ficheros. Se pretende que el sistema pueda proporcionar la información que tendría un usuario avanzado para solucionar el error y que en un entorno presencial podría proporcionar el profesor.

Así, cada tipo de error tendrá asociada información de posibles causas y acciones que debe tomar el alumno para corregirlo. Esta información puede ser creada inicialmente por el profesor; pero además el alumno puede añadir sus propios comentarios sobre el error: causas que producen el error, cómo puede solucionarse,... esta información completa y complementa a la creada inicialmente por el profesor. Cuando aparezca un error de un tipo determinado, el alumno podrá ver la información proporcionada por el profesor y la que él mismo a anotado anteriormente.

Queremos seguir desarrollando este módulo para que todos los usuarios pudieran ver las anotaciones de cualquier otro usuario y que hubiera un sistema para seleccionar los comentarios más "valiosos", los que a otros usuarios les han parecido más eficaces para comprender y solucionar el error. De esta forma estaríamos aprovechando la experiencia de determinados usuarios para los demás usuarios con lo que el aprendizaje debería ser más rápido.

5.3. Gestión de avisos y seguimiento

Se pretende que el sistema sea preventivo, es decir, que no sólo ayude a la corrección de errores en un programa ya realizado, sino que permita al usuario, mientras esté escribiendo el código, darse cuenta de cuales son los errores más frecuentes para no volver a cometerlos.

Para realizar esto, el sistema almacena en la base de datos contadores de los distintos tipos de errores y el momento en el que se generó el último, de esta forma puede obtener los errores que el usuario comete con mayor frecuencia y los últimos errores que ha cometido. Con esta información, el sistema muestra una serie de avisos personalizados mientras el usuario está escribiendo el código (figura 3).

Además, el sistema aprovecha la trayectoria de todo el grupo de usuarios que está utilizando el sistema, analiza el conjunto de todos errores del sistema y proporciona al usuario avisos sobre los errores más recientes y más frecuentes globalmente para todos los usuarios.

5.4. Base de datos de errores

La aplicación almacena toda la información sobre los errores en una base de datos relacional. Una de las tablas almacena los datos de cada uno de los errores de compilación que se producen. Los datos que se introducen en los campos de esta tabla se obtienen de la salida producida por el javac al compilar un fichero. En primer lugar, se clasifica el error en uno de los tipos mediante la búsqueda del patrón correspondiente y se separa en tokens siguiendo el formato correspondiente al tipo de error. Además, una vez clasificado el error, existen tablas en las que se contabilizan los errores de cada tipo por cada usuario y a nivel de todo el grupo y se guardan los errores más recientes también para estas dos categorías.

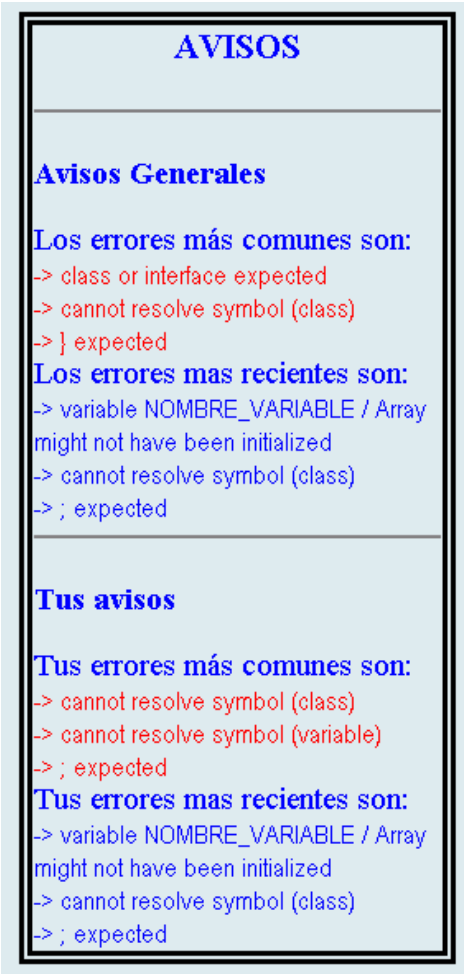


Figura 3. Avisos sobre los errores

Por otra parte para realizar el proceso de clasificación tenemos una tabla que contiene el patrón y el formato de cada tipo de error.

Por último, la base de datos tiene una tabla en la que recoge la información de ayuda que aportan los profesores y los alumnos sobre cada uno de los tipos de errores.

6. Conclusiones

Se ha diseñado e implementado un entorno para la enseñanza de lenguajes de programación que aporta varias novedades sobre otros entornos ya existentes:

- Compilador integrado en un entorno web, lo que evita la instalación y configuración por parte del usuario del entorno y permite la centralización del mantenimiento.
- Aportación de ayuda al usuario mientras está escribiendo código, no sólo cuando surgen errores. Mediante el módulo de gestión de avisos que obtiene métricas de frecuencia y últimos errores, tanto para el propio usuario como para el grupo en global. Esto ayuda a prevenir la aparición de nuevos errores.
- El usuario puede aprovechar la experiencia de otros en la solución de errores mediante el módulo de gestión de ayuda, que utiliza métodos de trabajo colaborativo para recoger información de todos los usuarios.
- El profesor puede hacer un seguimiento de los alumnos individualmente consultando los errores que comete el alumno. También puede obtener información de los errores que comete todo el grupo con mayor frecuencia y reforzar la explicación de los temas relacionados con los conceptos que están fallando, para evitar esos errores.

Actualmente se ha realizado una primera implementación del entorno y se han hecho pruebas de uso con un reducido grupo de alumnos. Se está planificando la implantación de este entorno para la realización de prácticas en asignaturas presenciales y con la experiencia adquirida implantarlo en un entorno de enseñanza virtual.

Referencias

[1] Integrated Development Environment based on Internet and eXtensible technologies: IDEFIX Proyect. http://www.di.uniovi.es/aplt/idefix , 2002.

[2] Página de inicio del proyecto Aulanet http://aulanet.uniovi.es/, 2002.

[3] Página de inicio de la herramienta WebCT. http://www.webct.com, 2002.

[4] Clear, T., Haataja, A., Meyer, J., Suhonen, J., and Varden, S. A. Dimensions of distance learning for computer science education. SIGCSE Bulletin 33, 2 (2001). ITiCSE 2000 Working Group Reports.

[5] Dawson-Howe, K.M. Automatic submission of programming assignments. SIGCSE Bulletin 27, 4 (1995) 51-53.

[6] Huizinga, Dorota M. Identifying Topics for Intructional Improvement Through On-line Tracking of Programming Assignments. SIGCSE Bulletin 33, 3 (Sep. 2001) 129-132. 6th Annual SIGCSE/SIGCUE Conference on Innovation and Technology in Computer Science Education.

[7] Labra Gayo, J.E., Morales Gil, J. M., Turrrado C., R. Plataforma de enseñanza de lenguajes de programación a través de Internet: Proyecto IDEFIX. Actas de JENUI 2002, 13-20.

[8] Labra Gayo, J.E. , Morales Gil, Jose M., Fernández Álvarez, A. M., and Sgastegui Chigne, H. A Generic e-Learning Multiparadigm Programming Language System: IDEFIX Project. ACM Technical Symposium on Computer Science Education (SIGCSE 2003), Reno, Nevada, USA, Feb. 2003.

[9] Marco Galindo, Ma Jesús; Prieto Blázquez, Joseph Necesidades específicas para la docencias de programación en un entorno virtual. Actas JENUI 2002, 5-12.

[10] Whitelaw, M. and Weckert, J. The Humannness of Object-Oriented Programming. I Cognitive Technology Conference. Hong Kong (1995).

Un enfoque para la enseñanza de la depuración de errores en las asignaturas de programación

Sergio Luján-Mora
Departamento de Lenguajes y Sistemas Informáticos
Universidad de Alicante
Carretera de San Vicente del Raspeig s/n
E-03080 San Vicente del Raspeig, Alicante
e-mail: sergio.lujan@ua.es

Resumen

Uno de los aspectos más difíciles de la programación es la depuración de errores. Como dice el aforismo, "errar es de humanos": los errores de programación son muy comunes incluso en los productos comerciales que se "supone" que se ven sometidos a controles de calidad exhaustivos. Por ello, no es de extrañar que cuando se comience a programar por primera vez o se aprenda un nuevo lenguaje de programación se cometan muchos errores, lo que suele ser origen de muchas frustraciones. Sin embargo, aunque se trata de un aspecto importante de la programación, no se le presta mucha atención tanto en la bibliografía sobre programación (existen pocos libros cuyo tema central sea la depuración de errores) como en la docencia. En este artículo mostramos cómo afrontamos este problema en el contexto de la asignatura "Programación y estructuras de datos" en la Universidad de Alicante: comentamos las carencias y malos hábitos que presentan los alumnos, presentamos las soluciones que hemos planteado para que el alumno desarrolle su capacidad de depurar programas y valoramos los resultados obtenidos durante los primeros meses de aplicación.

1. Introducción

La depuración de los programas es un tema que se suele dejar de lado cuando se aprende/enseña a programar debido a diferentes factores: falta de tiempo, complejidad del proceso, etc. Sin embargo, parece algo inherente a la programación (y se asume) que cualquier código contenga errores, lo que puede producir en ocasiones graves problemas. Así, por ejemplo, en la industria del software se suele considerar que lo normal es que haya 15 errores de programación por cada 1000 líneas de programa. Puede parecer una pequeña cantidad, pero en un programa real pueden ser muchos fallos: un comunicado interno de Microsoft afirmaba que Windows 2000 (con unos 40 millones de líneas de código) se pondría a la venta con 63.000 fallos [12], lo que es 10 veces menos de lo que cabría esperar si se aplica el valor de 15 errores por cada 1000 líneas.

Por otro lado, la baja calidad del software supone un grave problema hoy en día: un estudio del Instituto Nacional de Estándares y Tecnología de los Estados Unidos [14] estima que en dicho país el empleo de software con errores supone un coste anual cercano a los 60 mil millones de dólares (daños ocasionados, acciones legales, recursos destinados a la depuración de errores, etc.). Otro dato que confirma está desastrosa situación es que el "60% de los desarrolladores de software en Estados Unidos están involucrados en resolver errores que se podrían haber evitado" [7]. El mismo autor también estima que "sólo 47 días del calendario laboral se destinan a desarrollar el software, mientras que 150 días se dedican a verificar y depurar el software".

Aunque los datos anteriores se refieren a la situación en Estados Unidos, no creemos que la situación en España sea muy distinta. Ello nos lleva a creer que se debería de prestar más atención a la depuración de los programas e integrarla en el proceso educativo. Además, se debería de incidir en lo importante que es obtener software de calidad.

En este artículo mostramos cómo afrontamos este problema en el contexto de la asignatura

"Programación y estructuras de datos" (PED) de la Universidad de Alicante (UA). La depuración de errores es uno de los principales problemas a los que se enfrenta un alumno cuando estudia un nuevo lenguaje de programación. En este artículo mostramos los principales consejos y materiales que ofrecemos a los alumnos para que aprendan a depurar sus programas.

El resto del artículo se ha estructurado de la siguiente forma: en la sección 2 se comenta el tratamiento que recibe la depuración de errores en la bibliografía más extendida sobre los lenguajes de programación C y C++; en la sección 3 se describe la asignatura PED en la que se ha realizado la experiencia que se presenta en este artículo; en la sección 4 se comentan algunas de las carencias y malos hábitos que suelen presentar los alumnos de cara a la depuración de errores; en la sección 5 se presenta cómo abordamos el aprendizaje de la depuración de errores en PED; finalmente, la sección 6 cierra el artículo con las conclusiones.

2. Cómo se enseña a depurar errores en los libros

Muchos libros que tienen como objetivo enseñar a programar en un lenguaje de programación dedican poco o nada de su contenido a la depuración de errores. Por ejemplo, si nos centramos en los lenguajes C y C++ y consultamos algún libro de la bibliografía más extendida:

- En el libro de referencia básico de C [8] no se trata el tema de la depuración de errores en sus casi 300 páginas.
- En [4] sólo se dedican 4 páginas de casi 600 a la depuración de errores.
- En [2] únicamente se dedican 3 páginas de casi 300 a tratar los errores más comunes relacionados con la gestión de memoria.
- En [3], en el capítulo dedicado al preprocesador se explica la aplicación de la compilación condicional a la depuración y de forma dispersa a lo largo de las casi 1000 páginas del libro se comentan los errores sintácticos, los errores de enlazado y los errores lógicos, pero sin profundizar.
- Finalmente, en el libro de referencia de C++ [15] existe un capítulo dedicado al tratamiento de errores y excepciones en tiempo de ejecución, pero no se trata la depuración de errores.

El único libro que conocemos que trata el tema de la depuración de errores es [13], aunque sólo el primer tercio del libro se centra en la depuración de programas en general, ya que el resto del libro trata la depuración de errores en entornos concretos como son los programas para Microsoft Windows, los controles ActiveX o los componentes COM.

En definitiva, la depuración de errores es un tema que se suele dejar de lado. En la mayoría de los libros, se remite al lector a que consulte las secciones sobre compilación y depuración de errores en los manuales de referencia de su compilador. Pero en estos últimos únicamente se explican las herramientas que posee el compilador y cómo emplearlas, pero no se explican los errores de programación que se pueden cometer.

3. Características de la asignatura

La asignatura PED es una asignatura troncal de segundo curso en el plan de estudios de Ingeniería en Informática, Ingeniería Técnica en Informática de Gestión e Ingeniería Técnica en Informática de Sistemas en la UA. Consta de 9 créditos anuales: 4,5 créditos teóricos y otros 4,5 créditos prácticos. Posee como prerrequisitos las asignaturas "Fundamentos de Programación I" y "Fundamentos de Programación II", ambas de primer curso. El descriptor de la asignatura que aparece en el plan de estudios de las tres titulaciones en la UA es el siguiente:

- Estructuras de datos y algoritmos de manipulación.
- Tipos abstractos de datos.
- Diseño recursivo.

En el primer curso, los alumnos estudian los principios básicos de la algoritmia, las estructuras usuales de programación y la representación de los datos en la asignatura "Fundamentos de Programación I" durante el primer cuatrimestre, mientras que en el segundo cuatrimestre estudian un lenguaje de programación concreto (C++ sin la orientación a objetos [10]) en la asignatura "Fundamentos de Programación II".

En la asignatura PED, los alumnos estudian las estructuras de datos básicas (pila, lista, árbol binario, etc.) y algunas más avanzadas (árbol

AVL, árbol 2-3-4, trie, etc.). En las prácticas se emplea el lenguaje C++ [15] para implementar algunas de las estructuras de datos estudiadas. En las últimas prácticas, la implementación de una estructura de datos puede ser muy compleja, con varios miles de líneas de código.

4. Carencias de los alumnos

Muchos alumnos de la asignatura PED presentan una serie de carencias o falta de habilidades:

- No comprenden realmente qué funciones realizan las herramientas de desarrollo (compilador, enlazador, etc.).
- No entienden o no los tienen en cuenta los mensajes de error que generan las herramientas de desarrollo.
- No tienen muy claras las diferencias entre los distintos tipos de errores que existen (de diseño, lógicos, sintácticos, etc.).
- No saben cómo emplear un depurador para corregir los errores de sus programas.
- No son sistemáticos ni tienen paciencia a la hora de corregir los errores.

Estas carencias además se agravan por el comportamiento que suelen presentar los alumnos respecto a los errores en el código:

- No le dan importancia a los errores: "No es importante, si sólo falta declarar la variable y poner un punto y coma".
- No leen los mensajes de error: simplemente se fijan en que línea se ha producido un error.
- Piensan que la tarea del profesor es actuar como depurador: en cuanto se presenta un problema, solicitan la ayuda del profesor para que resuelva el error.
- "Echan balones fuera": a veces el alumno no puede localizar el origen de un error y llega a la convicción de que el código fuente no contiene errores. En esas ocasiones, muchos alumnos piensan que el problema se encuentra en el hardware o en el software (y en especial en el compilador).

5. Aprender a depurar

Debido a la importancia que creemos que posee la depuración de errores y a los problemas que presentan los alumnos, en la asignatura PED hemos incorporado el aprendizaje de técnicas y habilidades relacionadas con la depuración de errores. En esta sección comentamos los principales aspectos que enseñamos a los alumnos.

5.1. No todos los errores son iguales

Antes de intentar solucionar un error hay que saber de qué tipo de error se trata. A los alumnos les presentamos una clasificación de los errores de programación en base al momento en que se presenta u origina el error:

- *Errores de diseño*. Este tipo de errores aparece en el momento de plantear el algoritmo que resuelve el problema al que el alumno se enfrenta. Estos errores son independientes del lenguaje de programación empleado.
- *Errores lógicos*. Estos errores se producen por una incorrecta codificación del algoritmo planteado. Ejemplos de este tipo de errores son: condiciones lógicas mal planteadas, incorrecta elección de los tipos de datos de las variables, etc.
- *Errores sintácticos* (también llamados *errores gramaticales*, *de compilación* o *en tiempo de compilación*). Este tipo de errores impide que el código fuente se compile con éxito (si existen errores no se podrá generar el código objeto). Ejemplos de estos errores son: palabras clave mal escritas, expresiones incompletas, referenciar una variable no declarada, puntuación incorrecta (ausencia de paréntesis, punto y coma), etc.
- *Avisos de compilación*. Además de los mensajes de error, el compilador también puede generar avisos (*warnings*), errores que no son suficientemente graves como para impedir la generación del código objeto. Ejemplos de estos avisos son: empleo de variables no inicializadas, conversiones automáticas del tipo de las variables con una posible pérdida de información, etc.
- *Errores de enlazado*. Estos errores impiden que se genere el ejecutable final a partir de los ficheros con el código objeto. Ejemplos de estos errores son: ausencia de algún fichero objeto, empleo de funciones no definidas, empleo incorrecto de funciones de las librerías del lenguaje, etc.
- *Errores de ejecución* (también llamados *errores en tiempo de ejecución*). Este tipo de

errores impide que el programa se ejecute con éxito. Ejemplos típicos de este tipo de errores son: desbordamiento en una operación aritmética, división por cero, calcular la raíz cuadrada de un número negativo, etc.

Algunos autores defienden otros tipos de clasificaciones. Así, por ejemplo, en [3] se distinguen los errores sintácticos, los errores de enlazado y los errores lógicos, que pueden ser fatales ("[…] hará que un programa falle y que termine de forma prematura") o no fatales ("[…] permitirá que continúe el programa, pero produciendo resultados incorrectos").

Los errores de sintaxis y de enlazado son los más fáciles de detectar y corregir, ya que se genera un mensaje de error que nos avisa del problema. Los errores de ejecución se pueden resolver fácilmente si se genera algún mensaje de error, ya que muchas veces el programa simplemente termina su ejecución bruscamente. Sin embargo, los errores de diseño y lógicos son más difíciles de localizar, ya que pueden pasar desapercibidos. Esta cuestión se la remarcamos repetidamente a los alumnos, ya que muchas veces los alumnos piensan que si han obtenido un programa ejecutable significa que su código no contiene errores.

Por otro lado, en la mayoría de los casos los mensajes de error que producen los compiladores son breves y muy crípticos, lo que no ayuda a corregir los errores. A veces, la interpretación correcta de los mensajes de error suele implicar un conocimiento profundo del lenguaje por parte del programador, lo cual no posee un estudiante que se inicia en el uso de un nuevo lenguaje.

5.2. Mentalízate

Como se puede decir que es "inevitable" que se cometa algún error al programar, aconsejamos a los alumnos que cuenten por anticipado con la necesidad de tener que depurar su código en algún momento.

Además, un objetivo que nos marcamos es que comprendan lo importante que es obtener código sin errores. A los alumnos les describimos numerosos ejemplos [5] sobre las consecuencias que pueden acarrear los errores en el software, pero les resaltamos tres casos por la notoriedad o impacto que tuvieron:

- Entre 1985 y 1987 se produjeron en Canadá y Estados Unidos seis accidentes, con muertes y heridos graves por sobredosis de radiación, debido a un error de software en el ordenador que controlaba una máquina de radioterapia Therac-25 [9]. Bajo ciertas condiciones se producían condiciones de carrera, lo que originaba un funcionamiento incorrecto de la máquina.
- El 4 de junio de 1996 y durante su vuelo inaugural, el nuevo cohete Ariane 5 de la Agencia Espacial Europea se destruyó 40 segundos después de despegar. El cohete (resultado de un proyecto con un coste de 7 mil millones de dólares y más de 10 años de duración) y su carga (no asegurada por ser el primer vuelo) estaban valorados en 500 millones de dólares. La causa del fallo fue un error de software: una conversión errónea de un número en coma flotante de 64 bits a un número entero de 16 bits produjo un desbordamiento en el sistema de guiado y su posterior apagado automático.
- El 23 de septiembre de 1999, la sonda espacial Mars Climate Orbiter de la NASA, valorada en 125 millones de dólares, oficialmente se dio por perdida. La causa de la perdida fue un error en las unidades de medida empleadas en el software: mientras que en un módulo se empleó el sistema de medidas anglosajón (millas), en otro módulo se empleó el sistema métrico decimal (kilómetros). Ambos módulos tenían que comunicarse entre sí, pero no se realizaba ninguna conversión de los valores de un sistema de medida al otro.

5.3. Inténtalo tú mismo

A los alumnos les aconsejamos que depuren su código sin solicitar la ayuda de un compañero o del profesor. Si siempre que tienen un problema piden ayuda a alguien, nunca serán capaces de desarrollar buen código. Programar requiere ciertas habilidades que sólo se pueden lograr con la práctica.

5.4. Estado mental

La depuración de errores es una actividad mental que requiere una gran concentración. Por ello, recomendamos a los alumnos que si no encuentran a veces un error después de varios intentos, a

veces es mejor dejar la búsqueda para un momento posterior. Muchas veces, la fatiga y la frustración impiden localizar un error obvio.

5.5. Localiza el error

Lo más importante en el proceso de depuración es localizar el punto exacto del error. Para ello, una de las técnicas más comunes consiste en modificar el código fuente para que escriba ciertos mensajes para saber por donde discurre el hilo de ejecución o mostrar resultados intermedios para comprobar que los cálculos son correctos y volver a ejecutar el programa. En nuestra asignatura les enseñamos esta técnica, pero les indicamos que aprovechen las ventajas que ofrece la compilación condicional para no tener que modificar el código fuente cada vez que se quiera depurar el código o se desee obtener el programa final. De este modo, a partir del mismo código fuente se puede obtener una versión de depuración y una versión final. Por ejemplo, en la Tabla 1 se muestra un ejemplo de código que contiene instrucciones que sólo se compilan si el identificador `__DEPURACION__` se encuentra definido.

```
#include <iostream>
#define __DEPURACION__

int unaFuncion(void) {
#ifdef __DEPURACION__
cout << "Entra en: int unaFuncion(void)" << endl;
#endif

  int algo;
  /* Instrucciones de la función */

#ifdef __DEPURACION__
cout << "Sale de: int unaFuncion(void)" << endl;
cout << "Devuelve: " << algo << endl;
#endif

  return algo;
}

void main(void) {
#ifdef __DEPURACION__
cout << "Inicio del programa" << endl;
#endif

  /* Instrucciones */
  unaFuncion();

#ifdef __DEPURACION__
cout << "Fin del programa" << endl;
#endif
}
```

Tabla 1. Ejemplo de compilación condicional

5.6. Prueba y error

Una de las técnicas que más emplean los alumnos para depurar su código es la de “prueba y error”, que consiste en el ciclo modificar el código fuente, compilar y probar (ejecutar), que se repite sucesivamente hasta que “parece” que se ha resuelto el problema. A los alumnos les explicamos que esta técnica es “peligrosa” si no se emplea con cuidado y de forma sistemática, ya que en el proceso de resolver un error puede ser que se hayan introducido nuevos errores.

5.7. Cuidado con los mensajes de error

Los mensajes de error que genera un compilador suelen ser crípticos y poco explicativos, lo que ayuda poco a la depuración. Pero si además no se tienen en cuenta las dos siguiente reglas, el proceso de depuración puede ser un "infierno":

- Cuando el compilador genera muchos mensajes de error, puede ser que haya tantos que no quepan en una pantalla y se produzca un desplazamiento. En esa situación, muchos alumnos tienen la mala costumbre de comenzar a resolver los últimos errores, ya que son los que pueden ver en la pantalla. Sin embargo, los errores hay que resolverlos desde el primero hacia el último, ya que al resolver un error pueden desaparecer mensajes de error posteriores que son falsos: "El primer mensaje de error mostrado siempre es un error real; sin embargo, los últimos errores puede ser que no sean reales" [1]. Por ello, aconsejamos a los alumnos que redirijan la salida de error del compilador a un fichero para poder consultarlos desde el primero.
- En muchas ocasiones, los alumnos se centran en la línea del código fuente que el mensaje de error indica que posee un error y se olvidan del resto del código. Sin embargo, un error se puede manifestar en una línea aunque su origen se encuentre en una línea distinta: "El origen de un mensaje de error se puede encontrar en cualquier línea por encima de la línea señalada en el mensaje de error; sin embargo, el origen no puede estar en una línea posterior" [1]. A los alumnos les aconsejamos que cuando depuren tengan una visión global del código fuente y que no se centren exclusivamente en una línea.

5.8. Tu amigo el depurador

Los alumnos suelen huir del depurador, ya que lo consideran "engorroso" de emplear y poco útil. En [3] se deja claro este problema: "[…] Sin embargo, con frecuencia los depuradores son difíciles de utilizar y de comprender, por lo que son rara vez utilizados por los estudiantes de un primer curso de programación".

El origen de esta situación suele ser el desconocimiento: normalmente a los alumnos nadie les ha enseñado con ejemplos cómo emplear un depurador, lo único que han recibido son explicaciones sobre su manejo. Así, en cuanto descubren las ventajas del empleo de los puntos de ruptura (*breakpoints*), la ejecución paso a paso o la visualización de los valores de las variables, no dudan en emplearlo a partir de entonces. Por ello, a los alumnos de PED les enseñamos con ejemplos reales de código cómo se emplea un depurador.

5.9. Conjunto de ficheros de prueba

Aun cuando con un programa se obtengan los resultados correctos (esperados), no se puede asegurar que el programa no contenga errores, ya que algunos errores sólo se producen en situaciones muy concretas. A los alumnos les indicamos que se deben crear sus propios ficheros de prueba que les permitan verificar automáticamente y de forma exhaustiva el correcto funcionamiento de sus programas. Les aconsejamos que empleen estos ficheros conforme finalicen módulos de sus programas y que cuando hayan finalizado e integrado todos los módulos, vuelvan a verificar todos los ficheros para comprobar que no existe algún error producido por la integración de los módulos.

5.10. Contempla todos los posibles casos

Una de las principales causas de error es no contemplar todos los posibles casos porque se cree que es "imposible" que se produzcan. A los alumnos les indicamos que siempre tienen que contemplar todos los posibles casos.

Por ejemplo, en las sentencias condicionales simples (`if`) o múltiples (`switch`) es conveniente proporcionar siempre un caso por defecto (`else` o `default`, respectivamente) para atrapar los posibles errores que se puedan producir, aun cuando se esté completamente seguro de que el programa no contiene errores y se crea que sólo se pueden dar los casos contemplados.

5.11. Los errores son evitables

En la industria del software existe la creencia de que los errores son inevitables en los programas. Debido a ello existe la política de verificar un programa cuando está terminado. Pero este

procedimiento se ha comprobado desde hace años que es muy costoso y poco eficiente [6]. Los principales problemas que presenta esta forma de trabajar son:

- El proceso de depuración consume mucho tiempo.
- El proceso de depuración no asegura que un programa esté libre de errores.

Por ello, a los alumnos les explicamos que con un buen análisis y diseño de su código se pueden evitar muchos errores desde el principio. Los errores no se tienen que dejar para el final: se tiene que abordar su resolución desde el principio.

Finalmente, a los alumnos les proporcionamos una guía [11] que explica los principales errores lógicos, sintácticos y de enlazado que suelen cometer los estudiantes cuando comienzan a trabajar con el lenguaje C++. En esta guía los errores se han clasificado en siete categorías:

- Sobre el fichero *makefile* y la compilación.
- Sobre las directivas de inclusión.
- Sobre las clases.
- Sobre la sobrecarga de los operadores.
- Sobre la memoria.
- Sobre las cadenas.
- Varios.

5.12. Errores más comunes

Al enlazar, no incluir un fichero necesario. Ejemplo:

```
g++ -c unaclase.cc
g++ -c prueba.cc
g++ -o prueba prueba.o
```

Mensaje de error:

```
prueba.o: In function 'main':
prueba.o(.text+0xe): undefined reference to 'UnaClase::UnaClase(void)'
prueba.o(.text+0x2f): undefined reference to 'UnaClase::~UnaClase(void)'
prueba.o(.text+0x4a): undefined reference to 'UnaClase::~UnaClase(void)'
```

Solución: Verificar que en el proceso de enlazado se tienen en cuenta todos los ficheros necesarios.

```
g++ -c unaclase.cc
g++ -c prueba.cc
g++ -o prueba prueba.o unaclase.o
```

Tabla 2. Ejemplo de error de compilación

Confundir la declaración de una función amiga (friend) con los modificadores de visibilidad. Ejemplo:

```
class UnaClase {
  friend:
    int funcionAmiga(void);
  public:
    UnaClase();
    ~UnaClase();
    ...
};
```

Mensaje de error:

```
In file included from unaclase.cc:1:
unaclase.h:9: parse error before '('
```

Solución: El modificador friend se tiene que poner a cada función que sea amiga.

```
class UnaClase {
  friend int funcionAmiga(void);
  public:
    UnaClase();
    ~UnaClase();
    ...
};
```

Tabla 3. Ejemplo de error sobre las clases

El objetivo de la guía es mostrar cómo reconocer y corregir errores comunes y otros poco conocidos de forma rápida y fácil. Para cada error se incluye un enunciado del error, un ejemplo que contiene el error, el mensaje de error que se produce (al compilar, al ejecutar, etc.) y una solución al problema. Por ejemplo, en la Tabla 2 se muestra un ejemplo de error perteneciente a la categoría de errores de compilación y en la Tabla 3 un ejemplo de error perteneciente a la categoría de errores sobre las clases.

6. Resultados obtenidos

Es difícil valorar si la incorporación del estudio de la depuración de errores influye en el rendimiento académico, ya que este es el primer año en que hemos incorporado este tema y no disponemos de datos cuantitativos. Sin embargo, sí que se aprecia un mayor interés en los alumnos por lograr programas sin errores y una disminución en el número de consultas que realizan referentes a su código. Por otro lado, la guía de errores más comunes en C++ [11] ha recibido una buena valoración por parte de los alumnos e incluso algunos alumnos nos han hecho llegar ejemplos que no se habían incluido en la guía.

7. Conclusiones

En este artículo hemos explicado cómo nos planteamos la enseñanza de la depuración de errores dentro de la asignatura "Programación y estructuras de datos". Creemos que la depuración de errores es un aspecto de la programación que no suele recibir la atención que debería. Por ello, en nuestra asignatura abordamos su estudio. Con este artículo pretendemos dar a conocer nuestro planteamiento e intercambiar ideas con la comunidad educativa universitaria.

Como trabajo futuro está pendiente la realización de un estudio para averiguar si la enseñanza de la depuración de errores influye en el rendimiento académico del alumno y el desarrollo de una metodología de depuración.

Referencias

[1] Carter, Paul. *How To Debug Programs*. 2001. Disponible en Internet: http://www.drpaulcarter.com/cs/debug.php.

[2] De Pereda, Miguel y Matero, Javier. *Programación Orientada a Objetos con C++*. Anaya Multimedia, 1994.

[3] Deitel, H. M. y Deitel, P. J. *Cómo programar en C/C++*. Prentice Hall Hispanoamericana, 2ª edición, 1995.

[4] Gottfried, Byron S. *Programación en C*. McGraw-Hill, 1991.

[5] Huckle, Thomas. *Collection of Software Bugs*. Disponible en Internet: http://wwwzenger.informatik.tu-muenchen.de/persons/huckle/bugse.html.

[6] Humphrey, Watts S. *Comments on Software Quality*. En National Conference of Commissioners on Uniform State Laws for their Annual Meeting, 1997. Disponible en Internet: http://www.2bguide.com/docs/whsq.html.

[7] Jones, Capers. *The Impact of Poor Quality and Canceled Projects on the Software Labor Shortage*. Informe técnico, Software Productivity Research, Inc., 1998.

[8] Kernighan, Brian W. y Ritchie, Dennis M. *El lenguaje de programación C*. Prentice Hall Hispanoamericana, 1986.

[9] Leveson, Nancy y Turner, Clark S. *An Investigation of the Therac-25 Accidents*. IEEE Computer, Vol. 26, No. 7, julio 1993, páginas 18-41.

[10] Llopis Pascual, Fernando y Pérez López, Ernesto. *C++-OO como lenguaje introductorio a la programación*. En VII Jornadas de Enseñanza Universitaria de la Informática, páginas 299-304, 2001.

[11] Luján Mora, Sergio. *Errores más comunes en C++*. Disponible en Internet: http://www.dlsi.ua.es/~slujan/files/errores.pdf.

[12] Minasi, Mark. *¿Sirven para algo las versiones betas?* Windows 2000 Magazine 47, 2000. Disponible en Internet: http://www.windowstimag.com/atrasados/2000/47_nov00/articulos/engarde.htm.

[13] Pappas, Chris H. y Murray, William H. *C++. Sin errores*. McGraw-Hill, 2001.

[14] RTI. *The Economic Impacts of Inadequate Infrastructure for Software Testing*. Informe técnico, National Institute of Standards & Technology (NIST), PR 02-3, 2002.

[15] Stroustrup, Bjarne. *El Lenguaje de Programación C++*. Addison Wesley, 2002.

Mejora de la comprensión de las estructuras de datos

Raquel Lacuesta, Karmelo Urzelai
Departamento de Informática e Ingeniaría de Sistemas
Escuela Universitaria Politécnica de Teruel
Universidad de Zaragoza
lacuesta@unizar.es, karmelo@unizar.es

Resumen

El artículo presenta el desarrollo de animaciones interactivas para dar soporte a la docencia de la asignatura Estructuras de Datos, en una ingeniaría técnica en informática.

La asignatura consta de un amplio temario que hace que al alumno no le resulte fácil asimilar todos los modelos; el objetivo propuesto es facilitar su aprendizaje, así como animarle al autoestudio, poniendo en internet animaciones de soporte para su consulta.

Las animaciones desarrolladas se han clasificado en 8 grupos, cada uno de ellos asociado a un tema de la asignatura: costes computacionales, pilas, colas, listas, tablas, árboles, grafos y skip-lists. Para cada uno de estos grupos se describen las animaciones realizadas, mostrándose imágenes de algunas de ellas.

Se presenta un enfoque didáctico centrado en la motivación y refuerzo mediante animaciones interactivas que ayuden a la compresión de la asignatura.

1. Introducción

La asignatura de Estructuras de Datos se imparte en Teruel en el primer curso de la Ingeniería Técnica en Informática de Gestión. Los alumnos llegan a la asignatura únicamente con los conocimientos adquiridos en la asignatura de Programación I impartida en el primer cuatrimestre.

Los resultados de los primeros años de implantación de la titulación han sido poco satisfactorios debido al gran número de abandonos y a al escasa número de aprobados finales.

Analizando el problema, y dejando de lado motivos ajenos a la propia asignatura y su docencia en los que no se puede actuar, o en los que la resolución incumbe a otros entornos, las causas resultantes se han concentrado en dos aspectos: la amplitud del temario, y el todavía escaso nivel de programación y consecuente falta de capacidad de abstracción.

Analizando ambos aspectos, el temario, a pesar de ser amplio, aparte de ser coherente con la misma asignatura dada en otros centros, se considera adecuado a su número de créditos. De todas formas se ha realizado una adecuación en el tiempo dedicado a cada tema, quitando peso a algunos temas menos importantes, y dando más otros.

Por otro lado el nivel ofrecido en la asignatura previa de programación se considera correcto, y al menos de momento no se ha tomado en consideración desplazar la asignatura en el plan de estudios para dar una asignatura adicional intermedia.

2. Descripción de la asignatura.

La asignatura se divide en nueve temas, de los cuales cinco se dedican a la presentación de diferentes estructuras de datos, y donde para cada una de ellas se muestran varias implementaciones.

Los Tipos Abstractos de Datos (TAD) mostrados son: estructuras de datos lineales (pilas, colas, listas), tabla (*hashing*), árboles, grafos y *skip-lists*. [2][8]. Todos los conceptos teóricos se refuerzan en prácticas de laboratorio, que se desarrollan en el lenguaje de programación Ada 95 [1], donde deben implementarse variantes de los diferentes TADs explicados en las clases de teoría.

Lo que se pretende a lo largo del temario impartido es facilitar un esquema lógico para manipular los datos en función del problema que se deba tratar y el algoritmo que se utilice. Es importante escoger la estructura de datos y su

implementación más adecuadas, por lo que es fundamental conocerlas y comprenderlas.

Cabe destacar que se tratan tanto estructuras de datos estáticas (tamaño en memoria fijo) como dinámicas (tamaño en memoria variable), y que el planteamiento a la hora de explicarlas se debe adaptar a sus peculiaridades.

3. Motivación

El objetivo principal para la mejora de resultados, dentro del ámbito docente de la asignatura, ha sido intentar mejorar el entendimiento por parte de los alumnos de los diversos temas, intentado facilitarles la visualización de las diferentes estructuras de datos, y hacerles más ameno y llevadero el temario.

La explicación de algunas estructuras de datos implica mostrar al alumno cómo se mueven los datos, cómo cambian ciertos valores, como evolucionan por ejemplo los punteros, y todo ello es difícil hacer con medios estáticos como es la pizarra o las transparencias, que cómo mucho pueden mostrar imágenes inanimadas.

Se han desarrollado ya diversas aplicaciones para mejorar la compresión de algunas estructuras [7], queriendo aquí ampliar las posibilidades de una forma muy visual.

Por tanto tras considerarse insuficientes los medios utilizados hasta ahora, se decidió incorporar las nuevas tecnologías para mostrar la evolución de las estructuras de datos de forma animada. A su vez se consideró que una vez que se realizaba el esfuerzo sería interesante que estas animaciones no se vieran de forma pasiva sino que sería conveniente la implicación del alumno, haciendo que éstas fueran interactivas exigiendo al menos un mínimo de actuación por su parte, y en algunos casos añadiendo ventanas explicativas de los pasos seguidos.

La interactividad de las animaciones facilita un mejor entendimiento de los conceptos, frente a aquellas animaciones que sólo proporcionan una visualización del dinamismo de la estructura, ya que entre otras cosas permiten al alumno seguir la animación paso a paso de acuerdo a sus propias necesidades, y a su velocidad de asimilación.

Las animaciones han sido desarrolladas siguiendo el proceso de aprendizaje de un alumno que desconocía la asignatura, para plasmar desde una perspectiva más realista las dificultades encontradas y expresar de forma más cercana al alumno las ideas [6]. Lógicamente en todo momento el profesor supervisaba la corrección de las operaciones mostradas en las animaciones y su aspecto pedagógico. Así mismo es el profesor quien con su experiencia ha escogido las animaciones más representativas y las estructuras de datos e implementaciones que más dificultad presentan habitualmente para su aprendizaje.

Por otro lado se ha ajustado el conjunto de animaciones a las estudiadas dentro de la asignatura, abarcando todo el temario. Y no sólo se han desarrollado animaciones pensando en el comportamiento del T.A.D., sino que en algunos cosas la animación se ha centrado más en una posible implementación, que es la que más dificultad de comprensión tenía. Y todo ello se ha realizado buscando siempre la perspectiva más didáctica posible.

El empleo de animaciones se ha utilizado también en otras materias, por ejemplo para la animación y simulación de algoritmos paralelos de exploración de grafos[3]. También se han hecho animaciones para representar estructuras de datos por ejemplo en [7], no adecuándose a la asignatura Estructuras de Datos sino a la de Programación, por lo que resultan insuficientes para impartir un temario completo de la asignatura Estructuras de Datos.

4. Objetivos

Con esta iniciativa nos proponemos:

- Ofrecer al alumno un material de apoyo que facilite el aprendizaje. Todas las animaciones están puestas a disposición de los alumnos para repasar cuantas veces necesite el temario impartido. Cada animación irá adjunta al tema correspondiente.
- Incentivar al estudiante mediante apoyos visuales a preparar la asignatura. La unión de los temas teóricos a las animaciones gráficas facilita y ameniza el aprendizaje.
- Promover el uso de nuevas tecnologías por parte del alumno. Tanto los temas de la asignatura como las animaciones están en la página *web* de la asignatura con lo que el alumno puede acceder fácilmente a todo el material, tanto desde casa como desde la universidad
- Mejorar la compresión de conceptos durante la clase. El profesor puede utilizar las diversas animaciones durante las clases para hacer

comprender a los alumnos los temas explicados.

Como conclusión, la base principal para la mejora docente está en el apoyo que las animaciones pueden dar tanto durante las clases de teoría y prácticas, como durante la fase de autoaprendizaje que posteriormente el alumno lleve a cabo para asentar los conceptos explicados previamente.

Y por otro lado las animaciones son un gran apoyo para todas aquellas personas que no pudiendo asistir a las clases presenciales desean seguir la asignatura, ya que muchas veces los apuntes de otros alumnos resultan totalmente insuficientes para seguir una explicación oral que mostraba el flujo de datos dentro una estructura.

5. Animaciones

El entorno utilizado para el desarrollo de las animaciones ha sido Flash [4][5] debido a su facilidad de uso en la creación de películas dinámicas e interactivas, consiguiendo, además, una optimización en el tamaño de cada animación mediante la utilización de gráficos vectoriales.

Para la visualización de las animaciones desde las páginas *web* de la asignatura será necesario el programa *Flash Player*, gratuito, de fácil acceso y disponible en casi todos los navegadores.

Cada estructura de datos tiene su correspondiente animación, y algunas de ellas disponen de varias animaciones ya sea por su dificultad, o por la variedad de implementaciones. Dentro de cada animación se podrán realizar las operaciones pertinentes a cada estructura, consiguiendo así la interactividad del alumno con la animación y por lo tanto un refuerzo personalizado.

Para cada una de ellas se estudiaron las diferentes posibilidades de representación, escogiéndose la más intuitiva. El objetivo perseguido fue el de conseguir un efecto de autoaprendizaje y una fácil asimilación de los contenidos durante su visualización.

Dentro de las animaciones hay algunas cuyo objetivo es mostrar de forma detallada el comportamiento de la estructura y otras cuya única función es dar al estudiante la idea intuitiva de la estructura. Por ejemplo en la explicación de las colas una primera animación representa el comportamiento de una cola de coches en una parada, pudiendo añadir coches y quitar coches, lo que ya de por sí puede dar la idea del comportamiento de una cola mejor que ciertas explicaciones formales. Una segunda animación representa ya una secuencia de datos en los que podemos realizar las operaciones de *encolar* y *desencolar*.

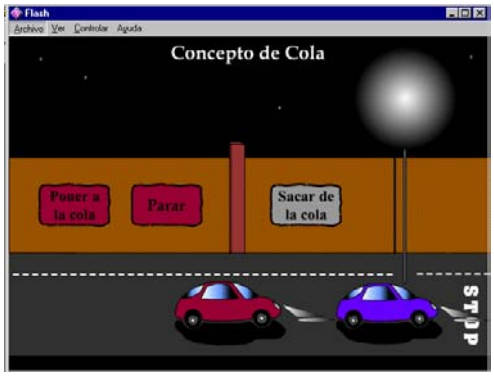


Figura 1. Concepto de cola

Todas las animaciones disponen de una serie de botones que permiten, dentro de ciertas restricciones, decidir la evolución de los datos, ver la animación paso a paso, reiniciar, o elegir entre diversas opciones. Algunas de ellas además disponen de cuadros de texto en los que se va explicando paso a paso las acciones que se están realizando y la fase en la que se encuentra.

En las animaciones que trabajan directamente con datos se ha mantenido una interfaz uniforme para darle homogeneidad, facilitar su seguimiento y no distraer al alumno.

Las animaciones realizadas [6] han sido:

1. Costes computacionales: La problemática de este tema es por un lado que su formalismo hace que el alumno sea más reacio a su estudio y por otro considerar que la eficiencia es un tema menor a causa de la creciente potencia de los ordenadores. La animación desarrollada por tanto incide en ambos aspectos de modo que por un lado presenta gráficamente los diferentes tipos de costes, y a continuación de una forma visual muestra las grandes diferencias entre unas y otras, para que el alumno sea consciente de la importancia de elegir un algoritmo de un coste razonable.
2. Pilas: Las animaciones de estructuras comienzan con ésta primera, muy sencilla, y

que introduce al alumno en la utilización de las animaciones que encontrará más tarde, que serán más complejas y elaboradas. En este caso una primera animación ofrece la idea intuitiva de pila por medio del apilamiento y desapilamiento de unos cubos, y otra animación posterior ya muestra el comportamiento más formal de una estructura de datos *pila*, con sus operaciones de *apilar*, *desapilar*, *cima* y *vacía*. Esta segunda animación ya introduce al alumno en la estética general de las animaciones.

3. Colas: A pesar de ser también una estructura muy sencilla se han definido dos animaciones, al igual que en las pilas una primera para aportar la idea intuitiva de *cola* (Figura 1), y una segunda para la definición más formal. Se intenta así con estas primeras lecciones atraer al alumno a "jugar" con las animaciones, de modo que posteriormente con animaciones más complejas le resulte más natural.
4. Listas con punto de interés: Para este caso, siguiendo la línea de las dos anteriores, se ha creado una primera animación para entender el concepto por medio de un pequeño editor de texto simulado. Y por otro se han construido una serie de animaciones orientadas a mostrar el comportamiento de una *lista* implementada de forma encadenada. Pese a explicarse en el temario también otro tipo de implementaciones se ha considerado innecesario realizar animaciones para ellas debido a su simplicidad.

   De las animaciones realizadas se ha incidido en los aspectos más críticos como la creación de la lista, para que se entiendan los conceptos de apuntador y de pila de posiciones libres. Otras de las animaciones muestran las operaciones más habituales de las listas y que pueden resultar difíciles de seguir sólo con una explicación oral, como el avance del *Punto de Interés* (Figura 2), la inserción de un nuevo elemento o el borrado. Las animaciones avanzan paso a paso, bajo el control del alumno, para que vea los cambios producidos tanto en el encadenamiento de la lista como en la gestión del espacio libre.

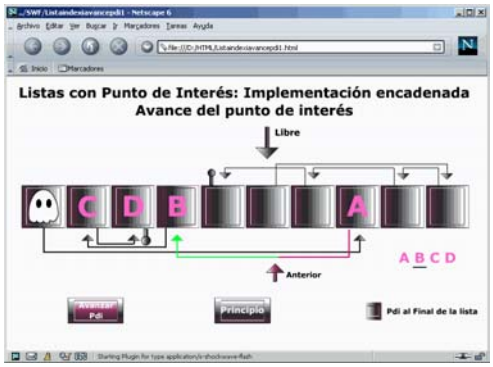


Figura 2. Listas con Punto de Interés

5. Tablas *Hash*: Parte de la dificultad de esta estructura de datos es mostrar el comportamiento de los tres tipos básicos de implementaciones que pueden darse a las tablas de dispersión: encadenamiento indirecto, encadenamiento directo y no encadenadas. Por ello se han realizado animaciones que muestran cómo se pueden implementar cada una de ellas y su comportamiento, mostrándose por ejemplo para las primeras, en tres animaciones, la inserción de un elemento en la tabla, el borrado de elementos y la consulta de si un elemento pertenece a la tabla (Figura 3).

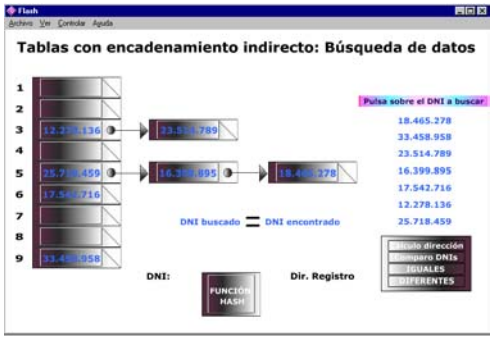


Figura 3. Búsqueda en tabla *hash*

6. Árboles: En el tema de árboles a modo de ejemplo se ha realizado una animación, que llama a otras tres animaciones, que muestran los tres tipos de recorrido en profundidad (Figura 4), y otra animación para mostrar el recorrido en anchura. A causa de que la implementación del recorrido en anchura de los árboles es un poco más complejo por la necesidad de una cola

que permita visitar los nodos en el orden correcto, en la animación a medida que se recorre el árbol se muestra también paralelamente paso a paso la evolución de la cola que nos permite realizar dicho recorrido

Figura 4. Recorrido en árboles

7. Grafos: Como ejemplo de utilización de grafos se ha realizado una animación que muestra la ejecución del algoritmo de Dijkstra. En dicha animación se avanza poco a poco seleccionando el siguiente nodo del camino de coste mínimo, y calculando los costes de los caminos a los demás nodos. Se muestra de forma explícita la tabla de costes y los caminos que los generaron, al mismo tiempo que se representa el grafo y se van marcando dinámicamente los diferentes caminos que se están calculando.
8. Skip List: Para mostrar esta estructura de datos, se ha dibujado una *skip-list* sobre la que se ha realizado el proceso de inserción de un elemento (Figura 5). Para la inserción en primer lugar se hace la búsqueda del punto de inserción, con lo que se está mostrando también cómo funciona ésta operación, pero para la posterior inserción además se van guardando los enlaces para poder redireccionarlos después.

6. Ejemplo de Animación

A modo de ejemplo se muestra paso a paso una de las animaciones asociadas a la estructura más simple de las realizadas: la pila.

Como ya se ha dicho, hay una primera animación que muestra la idea intuitiva de pila por medio de unos cubos etiquetados con letras, y la que aquí se explica es la segunda, que muestra, con la estética general de las animaciones, el comportamiento del TAD pila y sus operaciones.

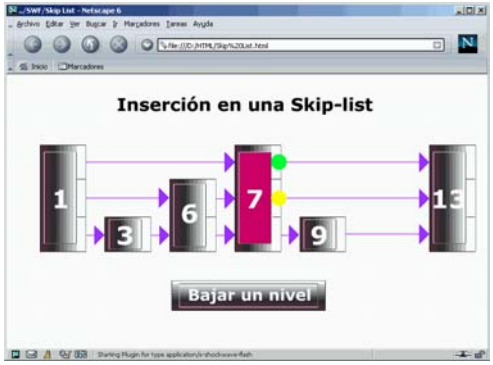


Figura 5. Skip-List

La animación comienza con una pila vacía, y el indicador *Pila Vacía* parpadeando para mostrar que está activado. El marcador de la cima de la pila está vacío al no haber ningún elemento en ella (Figura 6).

Figura 6. Pila vacía

Si se pulsa el botón de *Apilar*, el primer dato aparece por la izquierda y se desplaza hasta entrar en la pila y colocarse como cima. El indicador de *Pila Vacía* se desactiva y el dato introducido pasa a mostrarse en la ventana *Cima de Pila* (Figura 7).

Figura 7. Apilado de un dato

Si se pulsa *Apilar* más veces el proceso se repite llenándose poco a poco la pila, y modificándose el marcador de la cima de la pila (Figura 8).

Figura 8. Apilado de tres datos

En cualquier momento se puede pulsar también el botón de *Desapilar*, en cuyo caso el elemento de la cima, sale de la pila y desaparece por la izquierda (Figura 9).

Figura 9. Desapilando un dato

En el caso de desapilarse todos los elementos vuelve a encenderse el indicador de *Pila Vacía*.

El resto de animaciones pueden verse en http://eupt.unizar.es/eda/temario/Temario.html [1]

7. Conclusiones

Ante la problemática planteada en la asignatura Estructuras de Datos en la que los alumnos abandonaban fácilmente a causa de una temática amplia y la introducción de algunos conceptos nuevos, se ha decido apoyar tanto las clases como el autoestudio con animaciones.

Las animaciones se han realizado en Flash por su versatilidad, facilidad de uso y por ser prácticamente un estándar de *Internet*, ya que uno de los destinos principales de dichas animaciones es colocarlas en las páginas web de la asignatura, de modo que los alumnos puedan consultarlas en cualquier momento. También podrán ser utilizadas durante las clases para las explicaciones.

Las animaciones han sido realizadas por una persona que desconocía previamente la asignatura de modo que pudiera plasmar sus propias dudas y problemas durante el aprendizaje en las mismas animaciones. Lógicamente dichas animaciones en todo momento han estado guiadas por el profesor de la asignatura que además de aportar la experiencia sobre las dudas más habituales de los alumnos y controlar la corrección de todo lo realizado, volcaba su experiencia docente en la búsqueda de la representación más pedagógica posible

A lo largo del segundo cuatrimestre del curso 2002-2003 se van a utilizar las animaciones desarrolladas como herramienta docente, esperando una buena recepción del esfuerzo desarrollado, siendo positiva la impresión de los alumnos que ya las han visto. De todas formas con la realimentación de la reacción y la opinión de los alumnos se espera mejorar, adecuar y ampliar este primer paso.

Por último, consideramos que con este desarrollo no se completa ningún trabajo, sino al contrario la experiencia nos ha mostrado que se abren las puertas para la utilización de animaciones en muchas otras asignaturas, en áreas que pueden ser completamente diferentes.

Agradecimientos

Los autores de este artículo desean agradecer a Celia Marzo su dedicación y desarrollo de las animaciones. También desean agradecer a Elvira Mayordomo, del mismo Dpto., la iniciativa y desarrollo del proyecto, llevando ella posteriormente una línea totalmente independiente a pesar de compartir objetivos.

Referencias

[1] Barnes J. "Programmimng in Ada 95". Addison Wesley. 1999

[2] Franch X. "Estructuras de datos. Especificación, diseño e implementación". Edición UPC.

[3] José Fco. Cachairo González, Manuel Díaz Rodríguez, Antonio Vallecillo Moreno . "Animación y Simulación de Algoritmos Paralelos de Exploración de Grafos".

[4] Macromedia. "Manual Macromedia Flash 5. Guía de Consulta de Actionscript". Macromedia Inc.

[5] Macromedia . "Manual Macromedia Flash 5. Utilización de Flash". Macromedia Inc.

[6] Mazo C. "Animaciones Flash para la asignatura Estructura de Datos". Trabajo Fin de Carrera. Escuela Universitaria Politécnica de Teruel.

[7] Salamó M., Camps J., Vallespí C., Vernet D., Llorá X., Bernadó E., Garrell J.M., González X.. "Iniciativas para motivar a los alumnos de Programación". Universitat Ramon Llull.

[8] Weiss M.A. "Estructuras de datos y algoritmos". Addison-Wesley-Iberoamericana.

Análisis híbrido: una propuesta práctica

Francisco Palomo Lozano, Inmaculada Medina Bulo
Dpto. de Lenguajes y Sistemas Informáticos. Universidad de Cádiz.
Escuela Superior de Ingeniería de Cádiz. C/ Chile, s/n. 11003 Cádiz.
{francisco.palomo, inmaculada.medina}@uca.es

Resumen

Este artículo defiende la necesidad de introducir el *análisis híbrido* en las prácticas de las asignaturas en las que se imparte análisis y diseño de algoritmos secuenciales, y presenta un entorno de trabajo adecuado para su realización en el laboratorio.

Esta técnica de análisis es una mezcla de las técnicas tradicionales de *análisis teórico y empírico*: en el análisis híbrido se emplea la primera de ellas para establecer un modelo y la segunda para obtener datos experimentales que, por último, permiten ajustar el modelo mediante regresión.

1. Introducción

En las recomendaciones ACM/IEEE-CS del año 1991 [1] se sugiere explícitamente que dentro del área AL de *algoritmos y estructuras de datos* y, en concreto, en las unidades AL4 (análisis de la complejidad), AL5 (clases de complejidad) y AL6 (ordenación y búsqueda), se realice la

> *...medida de los tiempos de algoritmos seleccionados de distintas clases de complejidad...*

y también la

> *...corroboración de la complejidad teórica mediante el empleo de métodos experimentales...*

como parte de las prácticas de laboratorio a desarrollar siguiendo un modelo de *laboratorio cerrado*.

Actualmente, tras la última revisión realizada en el 2001 [2], el área AL ha pasado a ser *algoritmos y complejidad* y en AL1 (análisis básico de algoritmos) aparece explícitamente recomendada la

> *...medida empírica del rendimiento de los algoritmos.*

Como expondremos a continuación, limitar los métodos experimentales a un *análisis empírico* presenta claros inconvenientes.

Por otro lado, pensamos que un enfoque más integrado en el que se emplee un método de *análisis híbrido* ayuda a que el alumno aplique en el laboratorio una parte mayor de los conocimientos sobre *análisis teórico* adquiridos en las clases de teoría.

Nos centraremos en el análisis del tiempo de ejecución, aunque el método descrito es aplicable al análisis de otros recursos computacionales (como el espacio) siempre que exista una manera sencilla de medir su consumo en la práctica.

2. Técnicas de análisis

Siguiendo a [4], un texto de análisis y diseño de algoritmos de amplio uso, las dos técnicas principales de análisis de algoritmos son el *análisis teórico* y el *análisis empírico*. De la mezcla de ambas surge el *análisis híbrido*.

2.1. Análisis teórico

En el análisis teórico se fija una determinada operación patrón y se calcula una función matemática que mide el número de instrucciones de dicho tipo que ejecuta el algoritmo frente a una determinada medida del tamaño de su entrada.

Decimos que una operación es *crítica* cuando su función asociada tiene al menos el mismo orden que la correspondiente a cualquier otra operación del algoritmo y *elemental* si en su implementación su tiempo de ejecución tiene un orden constante.

Cuando el algoritmo se analiza respecto de una operación crítica, el *principio de invariancia* garantiza que el orden de la función que mide el tiempo de ejecución de cualquier implementación del algoritmo (en la que dicha operación sea elemental) coincide con el del algoritmo.

Es decir, el resultado es independiente de la máquina, del lenguaje de programación y del programador. En tanto en cuanto se mantengan las condiciones descritas, los resultados podrán extrapolarse a la implementación: el orden del tiempo abstracto del algoritmo corresponderá con el del tiempo físico de ejecución del programa.

La principal desventaja radica en que un análisis teórico preciso puede ser muy complejo. Por ejemplo, el análisis de la familia de algoritmos de ordenación de Shell no se ha completado, pese a ser un método clásico sobre el que se ha investigado exhaustivamente.

Afortunadamente, ocurre en muchos casos que, aunque el análisis preciso de la función puede ser muy complicado, la obtención de su orden puede realizarse con un esfuerzo bastante menor.

2.2. Análisis empírico

En el análisis empírico se mide el tiempo de ejecución de una implementación para un número suficiente de datos de entrada. La gráfica resultante de representar los tiempos frente al tamaño de la entrada puede emplearse para comprobar la tendencia del algoritmo y compararlo con otros.

Su principal ventaja es que proporciona datos reales de ejecución para una determinada implementación. Las desventajas de este método son evidentes. Desde un punto de vista teórico, y a falta de otra información, no podemos predecir nada acerca del comportamiento del algoritmo fuera de los valores observados.

Desde un punto de vista práctico, se puede perder una gran cantidad de tiempo en implementar un algoritmo muy ineficiente que no nos interesa en absoluto. En muchas ocasiones este tiempo no compensará al que se hubiera empleado en realizar un análisis teórico con el fin de rechazar el algoritmo como impracticable.

Incluso si el algoritmo es ineficiente pero interesante, como en el caso de los que resuelven problemas inherentemente complejos (por ejemplo, problemas NP-completos), este enfoque presenta sus inconvenientes, ya que un análisis empírico requerirá ingentes cantidades de tiempo para recabar los datos experimentales.

2.3. Análisis híbrido

En el análisis híbrido se realiza primero un análisis teórico encaminado a determinar la forma aproximada de la función de tiempo y posteriormente un análisis empírico para recabar datos. Finalmente se completa nuestro conocimiento de la función con los datos extraídos experimentalmente.

Se observa que podemos descomponer el proceso de análisis en tres etapas: modelado, experimentación y regresión.

Modelado

Es necesario establecer un modelo aproximado del comportamiento de la función de tiempo.

Una forma inicial de establecer dicho modelo es emplear un criterio asintótico. La ventaja de este enfoque es que únicamente necesitamos conocer el orden del algoritmo.

Supongamos que el tiempo del algoritmo viene dado por $t(n)$, siendo n el tamaño de la entrada, y que hemos determinado que $t(n) \in \Theta(f(n))$, pese a desconocer la expresión exacta de $t(n)$.

En la gran mayoría de los casos que se nos presentan en la práctica $t(n) \in \Theta(f(n))$ porque

$$\lim \frac{t(n)}{f(n)} = a \in \mathbb{R} \ .$$

En tales casos, podemos tomar $\hat{t}(n) = af(n)$ como aproximación de $t(n)$ para valores de n lo suficientemente grandes, es decir, $\hat{t}(n)$ será un modelo de la función desconocida $t(n)$.

La desventaja es que, al ser un criterio asintótico, la estimación sólo será realmente precisa para valores de n lo suficientemente grandes, es decir, a partir de un cierto umbral. Con frecuencia, este umbral es pequeño y se obtienen buenos resultados incluso para valores pequeños de n.

Si el algoritmo es lo suficientemente sencillo y no sólo se conoce el orden de $t(n)$ sino que también se conoce su expresión exacta, pueden emplearse modelos más detallados, con más parámetros.

Experimentación

El algoritmo debe implementarse con sumo cuidado, de manera que el programa obtenido refleje fielmente la idea subyacente sin introducir costes ocultos adicionales que no hayan sido tenidos en cuenta durante el modelado.

Pero esto no es suficiente. Cualquier hipótesis sobre los datos de entrada que haya sido considerada durante el modelado (por ejemplo, al calcular el orden) debe ser reflejada al seleccionar los datos de entrada para el experimento.

Por ejemplo, al analizar en el caso promedio un algoritmo de ordenación por comparación para vectores, es muy común que se suponga que todos los elementos del vector de entrada son distintos y sus permutaciones equiprobables. A menos que los vectores de entrada para el experimento se seleccionen de acuerdo con dichos criterios, el resultado no será significativo.

La medida de los tiempos también ha de realizarse con cuidado. Es posible que los tiempos que estemos midiendo sean menores que la resolución de nuestro aparato de medida. Por ejemplo, hemos comprobado que, en nuestro sistema, la resolución de la función estándar `clock()` es muy baja: aproximadamente $0{,}01$ s. Si no disponemos de un instrumento más preciso, podemos realizar una medida indirecta, midiendo el tiempo total de un número suficiente de repeticiones del experimento y promediando el resultado. Es muy importante que para cada repetición de un mismo experimento se empleen exactamente los mismos datos de entrada.

Siguiendo con el ejemplo, basta medir el tiempo total de 10 repeticiones de un mismo experimento y dividirlo entre 10 para obtener su tiempo con una precisión de $0{,}001$ s.

Regresión

Una vez que se dispone de los datos experimentales, hay que estimar los parámetros de regresión que aparecen en el modelo. Nótese que, salvo en casos muy sencillos, el modelo no es lineal. Por lo tanto, una regresión lineal simple no es aplicable.

Los métodos de tipo LLS (*linear least squares*) permiten, a partir de una función $\hat{y}(x) = ax + b$ y un conjunto de observaciones (x_i, y_i), obtener los parámetros a y b que proporcionan el mejor ajuste del modelo a las observaciones en el sentido de minimizar el error cuadrático medio [6, 9].

El método se denomina lineal no porque lo sea el modelo respecto de x, sino porque lo es respecto de a y b. De hecho, puede generalizarse para tratar modelos del siguiente tipo:

$$\hat{y}(x) = \sum_{k=1}^{n} a_k f_k(x) \ ,$$

donde las $f_k(x)$ son funciones arbitrarias.

Los métodos de tipo NLLS (*non-linear least squares*) son más generales y permiten trabajar prácticamente con cualquier tipo de modelo. El más utilizado por los diversos paquetes especializados es el de Levenberg-Marquardt [7, 8, 9].

3. El entorno de trabajo

Nuestro entorno de trabajo en un laboratorio equipado para realizar prácticas de las características descritas está constituido íntegramente por herramientas de software libre. Todas pueden encontrarse en cualquiera de las distribuciones habituales de LINUX.

Sobre todo empleamos GNU MAKE, GNU C++ y GNU PLOT, que pertenecen todas al proyecto GNU. Son herramientas potentes, estables, muy probadas y, sobre todo, gratuitas, algo muy importante si queremos que el alumno pueda trabajar en casa sin tener que realizar un costoso desembolso adicional.

MAKE nos ayuda a organizar el trabajo, no sólo de compilación, sino de experimentación. Nuestra experiencia demuestra que emplear un poco de tiempo en explicar esta herramienta, y suministrar a los alumnos unos apuntes no muy extensos sobre ella, ayuda a mejorar el rendimiento de éstos en el laboratorio.

GNU PLOT nos permite representar gráficamente los resultados experimentales, contenidos en ficheros de texto, frente a los modelos, suministrados como funciones, y comparar «a ojo» la bondad del ajuste (orden `plot`). Además, implementa internamente el método de Levenberg-Marquardt con lo que nos permite ajustar los parámetros de regresión de modelos muy complejos (orden `fit`) sin tener que emplear herramientas externas.

Otra ventaja que presenta es que es programable mediante *guiones* que pueden ser ejecutados desde la línea de órdenes o, automáticamente, desde MAKE.

3.1. ¿Por qué C++?

C++ [5, 10] es un lenguaje multiparadigma que permite programar en una variedad de estilos. Podemos diferenciar tres paradigmas básicos dentro del lenguaje que pueden mezclarse entre sí para lograr aún más flexibilidad. Esto hace que el lenguaje se adapte bien a una gran variedad de itinerarios curriculares.

Por un lado, puede emplearse como un lenguaje estructurado. En este sentido, puede considerarse que C++ es un C mejorado con un sistema de tipos más estricto que el de C, espacios de nombres separados, un mecanismo de control de excepciones y sobrecarga de funciones y operadores.

Por otro lado, implementa el paradigma de la programación genérica a través del concepto de *plantilla*. Las plantillas son definiciones paramétricas que permiten un alto grado de abstracción.

Por último, permite programar utilizando orientación a objetos. A diferencia de los lenguajes orientados a objetos puros, C++ posee herencia múltiple y el enlace es estático por omisión.

La posibilidad de sobrecargar el operador de llamada a función dentro de una clase, permite además la creación de *objetos función* que pueden pasarse como parámetro dotando a C++ de características propias de los *lenguajes de orden superior*.

No es necesario para el alumno dominar el lenguaje para sacar partido de él. Mostrarle un subconjunto adecuadamente escogido en relación a sus conocimientos previos puede ser suficiente. En una asignatura donde se enseña análisis y diseño de algoritmos, los alumnos aprecian la eficiencia del lenguaje como un valor añadido. C++ fue diseñado expresamente para tal fin.

Otra razón de peso para elegir C++ es la existencia de una *biblioteca estándar de plantillas* asociada al lenguaje que permite trabajar cómodamente con contenedores y algoritmos genéricos. Se trata de la STL [3, 5]: la única biblioteca que conocemos que forma parte de un estándar internacional y que incluye como parte de su especificación (normativa para todas las implementaciones) requisitos de complejidad. Esto la hace especialmente apropiada para su empleo en un laboratorio de análisis y diseño de algoritmos.

La STL se incorporó al lenguaje tras aprobarse una propuesta de A. A. Stepanov y M. Lee en una reunión del comité ANSI/ISO para la estandarización de C++ celebrada en julio de 1994. El estándar ISO/IEC [5] se aprobó finalmente en 1998.

3.2. ¿Por qué la STL?

La biblioteca estándar de plantillas es una biblioteca de clases contenedoras, iteradores y algoritmos. Proporciona implementaciones muy eficientes de estructuras de datos y algoritmos que se necesitan habitualmente en el desarrollo de programas más complejos y es el resultado de años de investigación desarrollada por sus autores sobre *programación genérica*.

El que la STL sea una biblioteca especialmente diseñada para la programación genérica se refleja en el hecho de que prácticamente todos sus componentes son paramétricos.

Existen otras bibliotecas escritas en C++, algunas muy completas, como LEDA, que podrían utilizarse para nuestros propósitos. Frente a ellas, STL presenta la ventaja de formar parte de cualquier implementación de C++ que cumpla con el estándar. La mayoría de los fabricantes de compiladores tienden a que sus productos se encuentren en dicho estado, con lo que incluyen implementaciones de calidad de la STL integradas en la distribución de sus compiladores. Con esto se facilita la creación de código transportable entre distintas plataformas.

Pero, sin duda, una de las características más sorprendentes e innovadoras de la STL es que como parte de su especificación incluye la complejidad asintótica temporal. Esto significa que cualquier fabricante de compiladores o bibliotecas que desee cumplir el estándar de C++ está obligado a proporcionar implementaciones de sus algoritmos que cumplan ciertos requisitos de eficiencia.

4. Conceptos clave de la STL

A continuación se exponen de manera general, los aspectos más relevantes que presenta la STL y que facilitan nuestra tarea a la hora de analizar y diseñar algoritmos escritos en C++.

4.1. Especificaciones de complejidad

Todas las operaciones de la STL poseen una especificación de complejidad. Ésta puede venir dada por un orden asintótico o, en los casos más simples, por una función concreta. La mayoría de las veces se especifica la complejidad del peor caso y, en ocasiones, la del caso promedio.

Otras veces, la complejidad temporal se especifica mediante un *análisis amortizado*. Esto es útil cuando el tiempo de una operación puede sufrir grandes variaciones a lo largo de una secuencia de operaciones. En este caso un análisis en el peor caso podría ser excesivamente pesimista y un análisis en el promedio, poco significativo, por la gran desviación de los tiempos.

4.2. Contenedores

Los contenedores son objetos que se utilizan para almacenar otros objetos (incluso otro contenedor) proporcionando operaciones para su manipulación.

La STL posee diversas clases contenedoras agrupadas en dos categorías: secuencias (como los vectores) y contenedores asociativos ordenados (como los conjuntos). También posee adaptadores de secuencia (como las pilas) que se comportan como contenedores especializados. Los nombres de las clases y su descripción aparecen en el siguiente cuadro:

`vector`	Vector
`deque`	Cola doble
`list`	Lista
`set`, `multiset`	Conjunto y multiconjunto
`map`, `multimap`	Asociación mono/multivalor
`stack`	Pila
`queue`	Cola simple
`priority_queue`	Cola simple de prioridades

4.3. Iteradores

Los iteradores son una generalización del concepto de puntero y se emplean principalmente para recorrer un contenedor. Muchos algoritmos manejan rangos de iteradores.

El rango $[i, j)$ representa a todos los elementos comprendidos entre los iteradores i y j sin incluir al último. Si `c` es un contenedor, todos sus elementos pueden representarse mediante el rango $[$`c.begin()`, `c.end()`$)$.

Existen distintos tipos de iteradores y cada contenedor proporciona los apropiados para que puedan emplearse algoritmos eficientes sobre ellos.

Los vectores y las colas dobles proporcionan iteradores de acceso directo. Esto significa que se puede acceder a cualquier elemento en tiempo constante, es decir, que el acceso se puede considerar como una operación elemental. Sin embargo, los iteradores de las listas y los contenedores asociativos son únicamente bidireccionales, ya que no es posible realizar acceso directo a un elemento de estos contenedores en tiempo constante.

Así, un algoritmo genérico diseñado eficientemente para emplear acceso directo, como es el caso del algoritmo de ordenación `sort()` de la STL, sería ineficiente si se aplicara a una lista que proporcionara iteradores de «acceso directo» de complejidad $O(n)$. Por lo tanto, las listas no proporcionan tales iteradores y para compensar poseen su propia función `sort()`.

4.4. Algoritmos

Hemos visto que la STL define contenedores que incluyen algoritmos específicos, pero también define algoritmos que son independientes del contenedor, en el sentido de que pueden emplearse sobre cualquiera que cumpla unos determinados requisitos.

La genericidad de los algoritmos independientes del contenedor puede descomponerse en tres factores ortogonales de diseño: son paramétricos, reciben iteradores (en lugar de contenedores) y pueden recibir objetos función.

5. Algunos ejemplos concretos

Vamos a aplicar las técnicas previamente descritas al análisis en el promedio de tres algoritmos de ordenación tal y como debería hacerlo un alumno en el laboratorio.

Las funciones en C++ que se presentan a continuación implementan distintos algoritmos genéricos que ordenan un rango $[i, j)$ de iteradores de acceso directo. En adelante, $n = j - i$ representará el número de elementos del rango.

La función `merge_sort()` implementa el algoritmo de ordenación por fusión. La STL define `inplace_merge()` que funde dos rangos ordenados $[i, k)$ y $[k, j)$ en no más de $n - 1$ comparaciones si hay memoria suficiente. Supondremos que la hay, en caso contrario, `inplace_merge()` puede emplear hasta $O(n \log n)$ comparaciones.

```
template <typename I>
void merge_sort(I i, I j)
{
  if (j - i > 1) {
    I k = i + (j - i) / 2;
    merge_sort(i, k);
    merge_sort(k, j);
    inplace_merge(i, k, j);
  }
}
```

La función `median_quick_sort()` implementa una variante del algoritmo de ordenación rápida, de Hoare. En esta variante se emplea la mediana como pivote. La STL define `nth_element()` que emplea un tiempo promedio lineal en reorganizar los rangos $[i, k)$ y $[k, j)$ de forma que los elementos del primer rango no sean mayores a los del segundo y que en k quede el elemento que ocuparía dicha posición si $[i, j)$ estuviera ordenado.

```
template <typename I>
void median_quick_sort(I i, I j)
{
  if (j - i > 1) {
    I k = i + (j - i) / 2;
    nth_element(i, k, j);
    median_quick_sort(i, k);
    median_quick_sort(k, j);
  }
}
```

La función `heap_sort()` implementa el algoritmo de ordenación por montículo, de Williams. La STL define `make_heap()`, que construye un montículo en no más de $3n$ comparaciones, y también `sort_heap()`, que lo ordena en $n \log_2 n$ comparaciones.

```
template <typename I>
void heap_sort(I i, I j)
{
  make_heap(i, j);
  sort_heap(i, j);
}
```

Primero se elige el modelo. Estamos interesados en un análisis en el promedio. No es difícil deducir, aun sin conocer la expresión exacta de $t(n)$, que los tres algoritmos realizan $\Theta(n \log n)$ comparaciones en tal caso. Así, según el criterio asintótico, podemos tomar $\hat{t}(n) = an \ln n$ como modelo.

En segundo lugar, hemos de realizar los experimentos. Hay que cronometrar el tiempo entre dos puntos de un programa, para lo que creamos una clase que emplea la función `clock()` de la biblioteca para medir el tiempo por diferencia. Aquí `CPS` es `double(CLOCKS_PER_SEC)`, el número de unidades internas de tiempo por segundo.

```
class Cronometro {
  clock_t t0;
public:
  Cronometro() { t0 = clock(); }
  double tiempo()
  { return (clock() - t0) / CPS; }
};
```

Con el siguiente programa obtenemos los datos para un análisis en el promedio. Por cada n se crea un vector de n elementos que se rellena con los valores $1, \ldots, n$. Entonces, se permuta aleatoriamente obteniendo una de las $n!$ permutaciones posibles. Así es como se selecciona la entrada para cada experimento.

```
int main()
{
  for (int n = N0; n <= N; n += I) {
    vector<double> v(n);
    generate(v.begin(), v.end(), G());
    random_shuffle(v.begin(), v.end());
    vector<double> t = v;
    long int r = 0;
    Cronometro c;
    do {
      ORDENAR(v.begin(), v.end());
      v = t;
      ++r;
    } while (c.tiempo() < 1.0);
    cout << n << '\t'
         << c.tiempo() / r << endl;
  }
}
```

La medida del tiempo es adaptativa. Cada experimento se repite durante, al menos, 1 s. Esto asegura que si el experimento dura menos de $0{,}01$ s se repita al menos 100 veces, permitiendo obtener una precisión de $0{,}1$ ms.

En el programa, `ORDENAR` representa a cualquier algoritmo que ordene un rango. Por otro lado, `N0` es el tamaño inicial, `N` el final e `I` el incremento. Para el experimento escogimos los valores `500`, `50000` y `1000`, respectivamente.

Como se observa, `generate()` requiere un objeto función para poder generar la secuencia $1, \ldots, n$. Para ello creamos la clase `G`, en la que se sobrecarga el operador de llamada a función.

```
class G {
  double i;
public:
  G(): i(1.0) {}
  double operator ()() { return i++; }
};
```

Por último, un sencillo guión para GNU PLOT permite realizar la regresión y obtener una gráfica del resultado frente a las observaciones. Por ejemplo, para `heap_sort()` hacemos:

```
set dummy n
t(n) = a * n * log(n)
fit t(n) "heap_sort.dat" via a
plot "heap_sort.dat", t(n)
```

En este caso `fit` calcula que $a = 83{,}0 \cdot 10^{-9}$, con lo que podemos estimar que el programa tarda un tiempo de $83{,}0\, n \ln n$ ns $\approx 57{,}5\, n \log_2 n$ ns. Análogamente, se obtiene $a = 133{,}1 \cdot 10^{-9}$ para `median_quick_sort()` y $a = 192{,}3 \cdot 10^{-9}$ en el caso de `merge_sort()`.

En los tres casos el sistema nos informa de que la desviación típica de los errores es inferior a $0{,}002$ y de que el error asintótico para el parámetro estimado es inferior al $1\,\%$. Esto da una idea de la bondad del modelo. En la figura 1 se presenta una gráfica comparativa que muestra los datos experimentales y su regresión.

6. Evaluación de la propuesta

Actualmente utilizamos exclusivamente análisis empírico en nuestras prácticas. Parece que ésta es la situación general en la Universidad española, donde no tenemos conocimiento de experiencias docentes de implantación del método aquí propuesto.

La principal desventaja de este enfoque es que no resulta fácil para los alumnos contrastar los datos experimentales obtenidos en el laboratorio con los resultados teóricos presentados en clase. Se produce así un salto entre ambos niveles que influye negativamente en la formación del alumno.

Si bien suele ser fácil para ellos comprobar cualitativamente que dos algoritmos tienen complejidades diferentes, no lo es comparar dos algoritmos de similar complejidad ni comprobar el impacto de ciertas mejoras como la elección del umbral óptimo en algoritmos de «divide y vencerás».

No obstante, realizamos una experiencia piloto con la ocasión de un curso de verano y las encuestas establecieron un grado elevado de satisfacción por parte de los alumnos. Entre las ventajas apreciadas al utilizar análisis híbrido cabe destacar que:

- Permite plantear prácticas más realistas a los alumnos.
- Facilita la comparación de algoritmos de similar complejidad.
- Permite estimar la constante oculta en la notación asintótica.
- Permite realizar predicciones cuantitativas sobre la evolución temporal.

En concreto, hemos observado que el alumno queda muy satisfecho cuando se le suministra un algoritmo desconocido y es capaz por sus propios medios de establecer un modelo cuantitativo de su complejidad temporal en el laboratorio.

7. Conclusiones

Hemos diseñado un entorno de trabajo adecuado a la realización de prácticas de análisis híbrido en el laboratorio. Este entorno está construido a partir de herramientas potentes y estables de software libre, lo que reduce los costes de implantación y facilita su uso al alumno.

Se ha ilustrado paso a paso el desarrollo con dichas herramientas de una práctica de laboratorio de estas características en la que se comparan diversos algoritmos de ordenación y se han expuesto las ventajas que aporta la técnica de análisis híbrido frente a un análisis meramente empírico.

Pretendemos implantar esta propuesta en el próximo curso, en las asignaturas *análisis y diseño de algoritmos I y II* de los nuevos planes de estudio de Informática en nuestra universidad. Estas asignaturas se impartirán en segundo curso, durante el primer y segundo cuatrimestres, respectivamente.

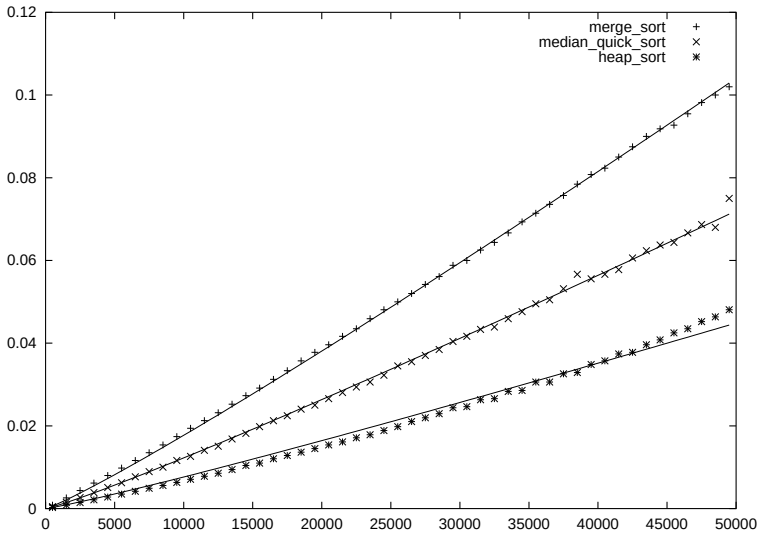


Figura 1: Resultados del análisis híbrido de los tres algoritmos

Referencias

[1] Joint Task Force on Computing Curricula. IEEE/ACM. *Computing Curricula 1991*. ACM Press and IEEE Computer Society Press (1991)

[2] Joint Task Force on Computing Curricula. IEEE/ACM. *Computing Curricula 2001*. ACM Press and IEEE Computer Society Press (2001)

[3] Austern, Matthew H. *Generic Programming and the STL: Using and Extending the C++ Standard Template Library.* Addison-Wesley (1998)

[4] Brassard, Gilles & Bratley, Paul. *Fundamentos de Algoritmia.* Prentice-Hall (1997)

[5] ISO/IEC 14882:1998. *Programming Languages – C++*. (1998)

[6] Jain, Raj. *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation and Modeling.* Wiley (1991)

[7] Levenberg, Kenneth. *A Method for the Solution of Certain Non-linear Problems in Least-Squares.* Quarterly of Applied Mathematics **2**(2) (1944)

[8] Marquardt, Donald W. *An Algorithm for the Least-Squares Estimation of Nonlinear Parameters.* SIAM Journal of Applied Mathematics **11**(2) (1963)

[9] Press, William H.; Teukolsky, Saul A.; Vetterling, William T. & Flannery, Brian P. *Numerical Recipes in C++: The Art of Scientific Computing.* Cambridge University Press. 2ª ed. (2002)

[10] Stroustrup, Bjarne. *The C++ Programming Language. Special Edition.* Addison-Wesley (2000)

Robótica e informática industrial

Docencia de la Planificación y el Desarrollo de un Proyecto de Informática Industrial

Lorenzo Moreno, Evelio González, Jonay Toledo, Leopoldo Acosta, Alberto Hamilton, J. Albino Méndez, Sergio Hernández, Marta Sigut, Nicolás Marichal, Santiago Torres
Grupo de Computadoras y Control
Departamento de Física Fundamental y Experimental, Electrónica y Sistemas
Universidad de La Laguna
28307 La Laguna, S/C Tenerife
e-mail: {lorenzo, evelio, leo, marta, jonay, nico, alberto, jamp, sergio, santiago}@cyc.dfis.ull.es

Jesús F. Montañez
Alumno de la Universidad de La Laguna y Becario de TOTALBAR, S.L.
email: jesusfm@cyc.dfis.ull.es

Resumen

Este trabajo elabora un plan de docencia de la planificación y desarrollo de un proyecto de Informática Industrial. Se basa en un proyecto real de colaboración de la ULL con una empresa del entorno. Dicho plan consiste en guíar al alumno a través de la mayor cantidad posible de fases: planificación, obtención de la placa de circuito impreso,... Aunque a fecha de redacción de esta ponencia, los alumnos no han pasado por todos los módulos propuestos, los resultados en cuanto a interés mostrado y conocimientos adquiridos es positivo. Animamos a emplear otros proyectos similares con idénticos fines docentes.

1. Introducción

El presente trabajo constituye un proyecto de docencia que pretendemos realizar en la asignatura de Informática Industrial y que pretende mostrar a los alumnos todos los aspectos que conlleva la realización de un proyecto comercial en esta disciplina: electrónica de potencia, electrónica digital, fabricación de circuitos impresos, verificación y validación del primer prototipo y optimización del producto final. No obstante, se planea extender estos contenidos a asignaturas de Diseño de Sistemas Digitales, Proyecto o Informática Industrial en los estudios de Ingeniería Eléctrica, Ingeniería Técnica Industrial e Ingeniería en Informática Especialidad de Sistemas, pudiendo entonces elaborar una distribución en horas por módulo según la cantidad de créditos a cubrir.

Para ello, nos basamos en un proyecto real de un desarrollo industrial realizado por nuestro departamento con una empresa del entorno. Hemos elegido este caso por:

- lo completo del proyecto que abarca todos los aspectos señalados anteriormente
- la clara diferenciación de las fases del proyecto, lo cual lo dota de un alto valor pedagógico.
- los abundantes recursos disponibles (documentación, prototipos iniciales y mejorados, producto final) que se derivan de un proyecto de estas características.
- nuestra experiencia de que los alumnos suelen mostrar un mayor interés cuando analizan situaciones y dispositivos de la "vida real"

A pesar de lo aparentemente concreto de esta aplicación, pensamos que es fácilmente extensible a otros proyectos reales que pueden ser aprovechados para ser mostrados a los alumnos, incluso en la docencia de otras áreas de la Informática.

No pretendemos por tanto, describir de forma exhaustiva este proyecto particular, sino guíar a los alumnos a través de la mayor cantidad posible de aspectos que surgen en un desarrollo comercial. Tampoco perseguimos que el alumno reproduzca el producto final (lo cual sería además imposible con la limitación temporal de la asignatura) sino que trabaje y asimile la mayor parte posible de las fases de un proyecto.

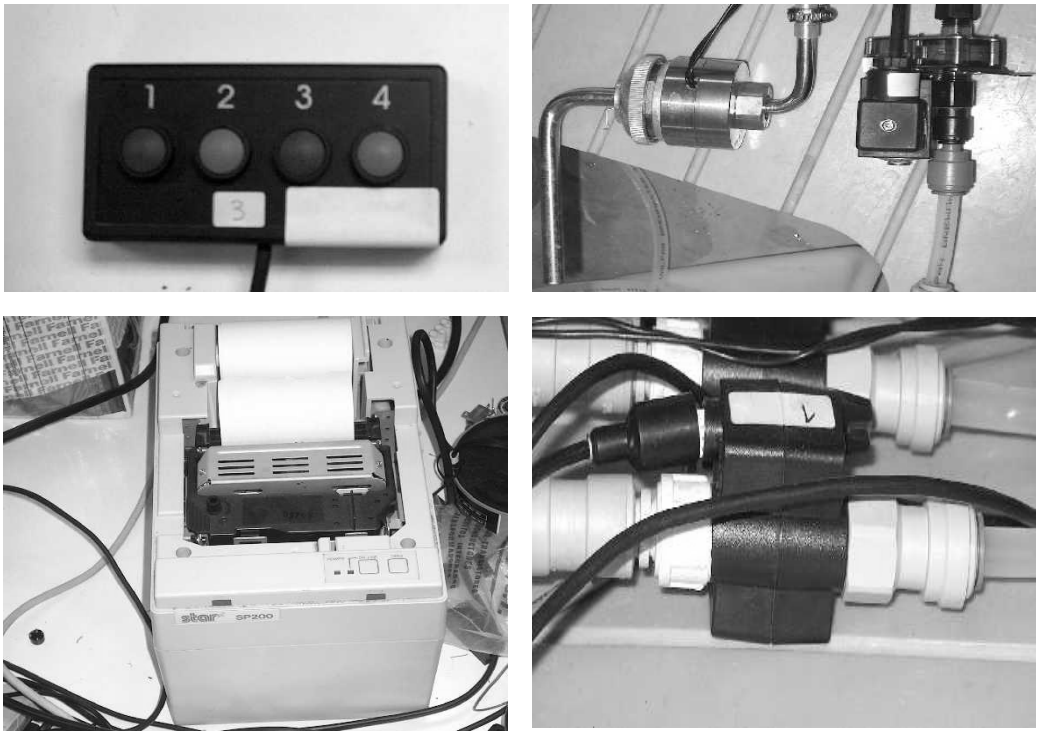


Figura 1. Elementos del proyecto presentado: botonera, electroválvulas, impresora y caudalímetro

El resto de la ponencia se estructura del siguiente modo. En primer lugar describiremos las especificaciones del proyecto, para posteriormente detallar las fases del mismo y cómo se reflejan en su docencia. Concluiremos esta ponencia evaluando el impacto observado en los alumnos de nuestra asignatura.

2. Descripción del proyecto

El proyecto que queremos desarrollar con los alumnos consiste en lo siguiente:

Se desea gobernar automáticamente el proceso de llenado de líquido de recipientes de diferente capacidad seleccionados mediante una botonera (figura 1 arriba izq.), activando una electroválvula (figura 1 arriba der.). El sistema debe asímismo poder envíar a una impresora (figura 1 abajo izq.) información del número de recipientes llenados durante el tiempo de funcionamiento.

Las cantidades de líquido a servir deben ser fijadas previamente en un proceso de calibración, ya sea por tiempo de apertura de la electroválvula o por caudal (método más exacto). La información de este caudal es sumistrado por un caudalímetro (figura 1 abajo der.) que suministra un tren de pulsos proporcional a la cantidad de líquido que circula por unas mangueras.

Finalmente, la información tanto de la calibración de las cantidades a servir como de la cantidad de recipientes llenados deben conservarse para la próxima vez que se encienda el dispositivo.

Podemos por tanto, listar las especificaciones de este proyecto del siguiente modo:

- Características Hardware
  - Modelo de electroválvula impuesto por la empresa
  - Modelo de caudalímetro impuesto por la empresa

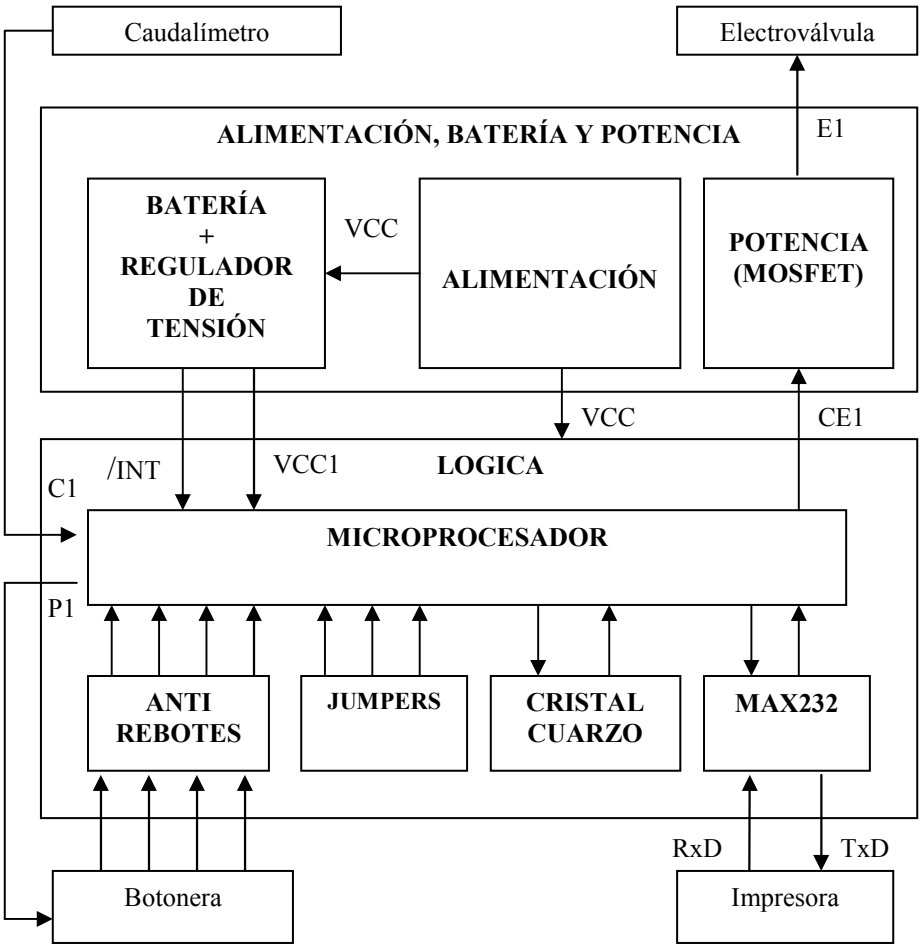


Figura 2. Esquema de bloques hardware del montaje, integrando los componentes suministrados por la empresa con dos placas que separan la parte de lógica de las de alimentación y potencia para activar la elevtroválvula. Se suministra al alumno para su estudio

- Microcontrolador a emplear
- Sistema de alimentación: batería para situaciones de power-fail
- Modelo de botonera impuesto por la empresa
- Limitación del tamaño del producto final

- Características Software
  - Modos servir por tiempo/servir por caudal
  - Modos calibración/operación/informe
  - Grabación de la información
- Tiempo de realización del proyecto
- Coste del producto

Como se puede observar en estas especificaciones, el sistema electrónico a diseñar contempla todos los aspectos de la electrónica que el alumno ha estudiado en anteriores asignaturas: electrónica digital, electrónica de potencia, alimentación y finalmente programación en lenguaje ensamblador para microcontroladores.

3. Planificación y Diseño del Primer Prototipo

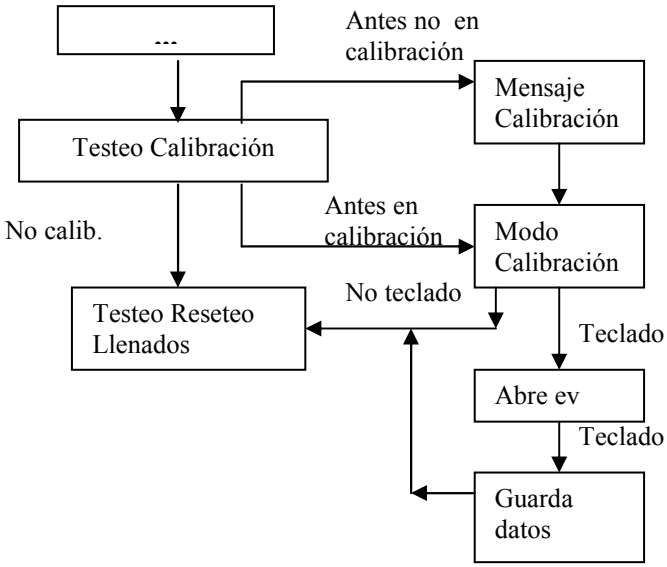


Figura 3. Detalle del diagrama de flujo del software del dispositivo, referido a la calibración del sistema. En el bucle principal se testea si el sistema se encuentra en modo de calibración. Si lo está y anteriormente no se encontraba en ese modo, lanza un mensaje a la impresora indicando que se acaba de entrar en ese modo, del que sale si no se detecta pulsación en las botoneras, pasando a la siguiente comprobación de modo (el de reseteo) . Al detectar una pulsación, el sistema abre la electroválvula y espera a que se pulse otra vez el botón, lo que indica que se ha llenado el recipiente con la cantidad de líquido deseada, almacenando el dato. Con este tipo de diagramas, los alumnos deben programar rutinas clave como la gestión de los pulsos o el almacenamiento de los datos.

Tras la descripción de las especificaciones del proyecto a realizar, los alumnos pasan al estudio de los apectos relacionados con la planificación del proyecto.

En primer lugar se les remarca la importancia de un cuidadoso proceso de planificación lo más realista posible, incluyendo todos los aspectos involucrados en el diseño del prototipo. Los alumnos deben saber extraer de este proceso de planificación las tareas a desarrollar y en qué plazos de tiempo, secuenciándolas a través de un diagrama de Gantt [2], poniendo de manifiesto las tareas que resultan ser críticas en la realización del proyecto. Existen numerosas aplicaciones para la elaboración de estos diagramas siendo la más empleada el Microsoft Project. No obstante ésta no es la única. Una simple búsqueda por Internet nos permite acceder a otras herramientas que se ajusten más a nuestra forma de enseñar estos aspectos. Cuando los alumnos, divididos en grupos de 2 ó 3, han realizado sus correspondientes diagramas, se hace una puesta en común, que permite enriquecer esta fase. Así los alumnos se dan cuenta de aspectos que habían pasado por alto o de suposiciones realizadas poco realistas.

Una vez realizado este proceso de planificación, empezaremos con el diseño del prototipo inicial. Dividiremos este proceso en las siguientes fases:

3.1. Elección de dispositivos

En esta fase se procede a la elección del procesador que gobernará el sistema (microcontrolador, PLC o procesador convencional) y la del resto de componentes: transformador, elementos de potencia, fuentes

conmutadas, módulos de transmisión serie, batería, rectificadores, relés,...

Respecto a la primera elección, se presentan a los alumnos las ventajas y desventajas (precio, funcionalidades, facilidad de programación) de cada una de los posibles procesadores. Tras ello se justifica la elección de un microprocesador de la familia ATMEL.

Para la segunda, cada grupo se encargará de la elección de los componentes de una parte del sistema. Estas partes son escogidas de modo cuidadoso para que no resulten excesivamente complicadas para el alumno y no tengan relación directa con la elección de los componentes de otras partes. Cada grupo dispondrá para esta elección de abundante documentación (catálogos y hojas técnicas de diversos componentes) que faciliten su labor. Una vez finalizado este estudio, el profesor de la asignatura lo evaluará.

3.2. Obtención del diagrama de bloques

La figura 2 presenta un diagrama de bloques del sistema. Dicho diagrama incluye cada uno de los elementos que debe ser diseñado en el prototipo y es suministrado a los alumnos para su estudio, lo cual permitirá visualizar el proyecto y ayudará en fases posteriores. En este caso, para el prototipo inicial se ha optado por dos placas que separen la parte de lógica de la de potencia y alimentación, para evitar en lo posible todo tipo de interferencias.

3.3. Diseño del circuito impreso

Se emplea un paquete de software como ORCAD u otro similar para el diseño del plano esquemático con el SDT (Schematic Design Tool). A partir del fichero esquemático, se genera la placa de circuito impreso o PCB (Printed Circuit Board) con el correspondiente módulo del programa (PCB layout en el caso del ORCAD), para posteriormente definir los ficheros GERBER [3] (formato ampliamente empleado y que contiene información sobre la anchura de las pistas y los puntos de inicio y final de cada una).

A nivel docente, en esta fase se pretenden que los alumnos trabajen con estos programas e investiguen todas las posibilidades que ofrecen. Como la placa global tiene demasiados componentes, se divide en módulos interconectados más pequeños, cuyo esquemático

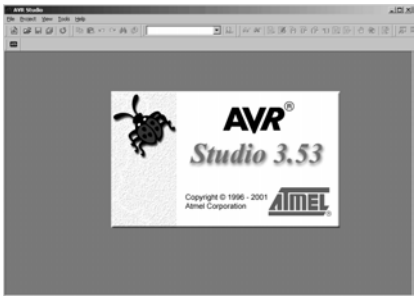


Figura 4. Entorno de programación AVR Studio

se suministra a cada grupo. Los alumnos obtienen a continuación el PCB y GERBER de cada módulo.

3.4. Diseño Software

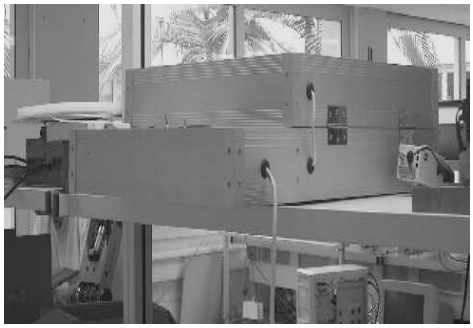
La fase final del diseño del apartado anterior se puede simultanear con el diseño software: diagrama de bloques, código y una primera comprobación del mismo mediante un simulador, si existe, del microcontrolador o autómata elegido. En nuestro caso, ATMEL [1] proporciona un completo entorno de programación para una amplia gama de microprocesadores: el AVR Studio (Figura 4). Este entorno es gratuito y se complementa con una abundante documentación sobre el mismo y sobre los microprocesadores.

A nuestros alumnos se les suministra el código fuente del dispositivo en ensamblador y el diagrama de flujo del mismo (detalle en Figura 3), a falta de unas rutinas principales (por ejemplo la gestión de los pulsos envíados por el caudalímetro o la detección de la caída del suministro eléctrico). La cantidad de trabajo a realizar dependerá del tiempo disponible en la asignatura.

3.5. Obtención de la placa de circuito impreso

Una vez que se dispone de la placa de circuito impreso, la cual ha sido realizada en nuestro laboratorio, se procede al testeo de la misma, detectando y corrigiendo todos los errores encontrados. En un prototipo como el nuestro, que dispone tanto de elementos de potencia

(electrónica analógica) como elementos digitales, los errores más comunes detectados son:



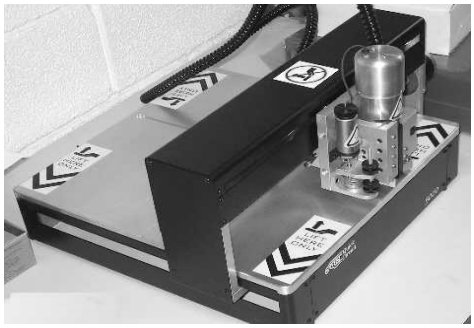


Figura 5. Diversos métodos de obtención de placas de circuito impreso: insoladora (izq.) y fresadora (der.). En el laboratorio se muestra a los alumnos brevemente ambos métodos

- Errores producidos por un diseño térmico deficiente. Se puede disipar demasiado calor en determinados componentes, lo que reduce su vida media
- Errores intermitentes: falsos contactos debidos a una mala soldadura, problemas de ruido producidos en el entorno que afectan al sistema, problemas de tierra, etc...
- Falta de optimización en el prototipo inicial: exceso de cableado, exceso de componentes, elementos redundantes, etc...

En relación con los alumnos, inicialmente se les comenta brevemente los procesos de elaboración del circuito impreso: el artesanal mediante insoladora y el de fresado. Esto se realiza en el laboratorio, mostrándoles el instrumental correspondiente que puede verse en la Figura 5.

Figura 6. Programador ALL-11 para el microcontrolador.

Al contar con los ficheros originales del proyecto, podemos replicar placas del prototipo inicial que son testeadas por los alumnos (una pequeña parte por cada grupo), incidiendo en los diferentes clases de errores posibles.

3.6. Finalización del primer prototipo

Disponiendo ya de la placa depurada, se integra en ella la primera versión de software depurada con simulador. Se inicia entonces otro proceso de testeo y depuración de errores, tanto hardware como software, hasta finalizar un primer prototipo.

Al final de esta fase, el alumno se habrá familiarizado con la programación y depuración de código ensamblador para microcontroladores. Cuentan para ello con un programador ALL-11 (Figura 6), que permite programar un amplio conjunto de microprocesadores.

3.7. Presentación del prototipo y proceso de mejora

Este primer prototipo cumple con las especificaciones iniciales del proyecto, pero no supone el fin del mismo. No es extraño, más bien todo lo contrario, que una vez que se presente el sistema a la empresa, ésta introduzca nuevas especificaciones. Éstas pueden derivarse de razones tan variadas como por ejemplo de un nuevo estudio de mercado, nuevas funcionalidades que se quiere asignar al dispositivo o de componentes complicados de

conseguir en el lugar de comercialización (recordamos que se desea comercializar el producto en otros países).

En paralelo, pueden localizarse posibles mejoras en el diseño (por ejemplo un diseño térmico más eficiente) y que a la vez faciliten el replicado del producto. En este sentido cabe citar:

- el prototipo inicial normalmente presenta conexiones adicionales (derivadas del testeo y verificación) que complican el replicado. Por tanto se realizan nuevas placas con dichas modificaciones.
- mejorar el conexionado, sustituyendo en lo posible las soldaduras por conectores modulares
- englobar el sistema en una única placa, ahorrándose de este modo las conexiones entre placas

En cuanto a nivel docente se plantean dos actividades:

- Se plantean algunas de las especificaciones reales sugeridas por la empresa en esta fase y se solicita a los alumnos que determinen qué modificaciones implican en el dispositivo.
- Se presentan en el laboratorio el prototipo inicial y diversos prototipos evolucionados, donde los alumnos puedan comprobar fácilmente las mejoras introducidas. En la Figura 7, pueden verse tres prototipos. En el inicial (arriba), un estudio térmico inicial no optimizado conduce a un disipador de grandes dimensiones. Esto fue reparado en el segundo prototipo (medio), pero las excesivas conexiones con cables dificultan el replicado. Finalmente el prototipo final (abajo) ha unificado el sistema con una única placa con conectores modulares.

3.8. Generación de la Documentación

Tras diversas realimentaciones desde el empresario en el punto anterior, el producto se encuentra listo para su comercialización. Pero antes debe ser complementado con una documentación imprescindible: el manual de usuario y el manual de instalador.

El plan de docencia termina proporcionándole a los alumnos copia de ambos manuales, incidiendo en algunos aspectos de redacción (detalles de las figuras, orden, completitud...)

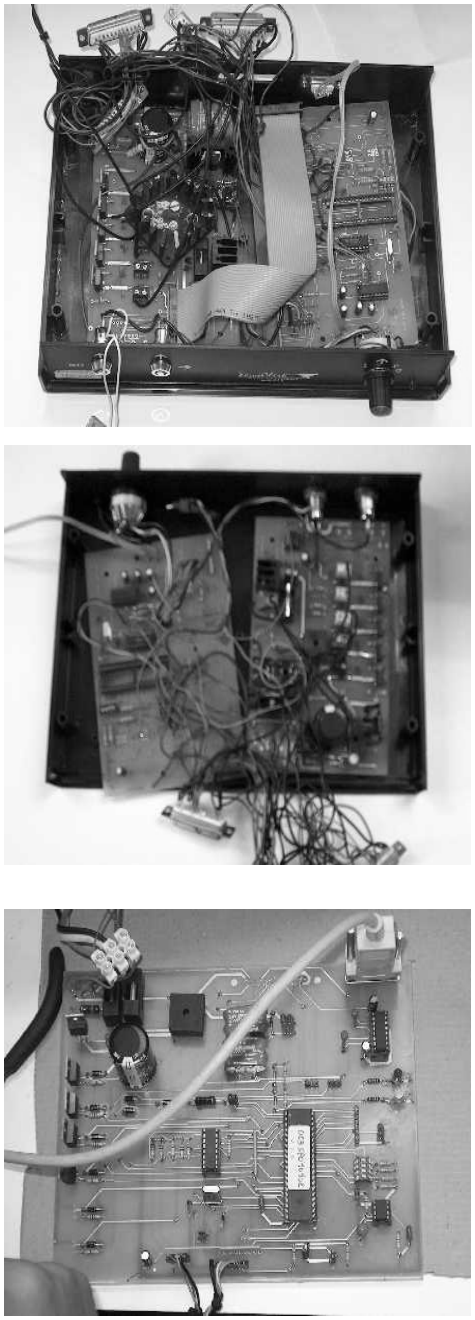


Figura 7. Evolución de los prototipos

4. Evaluación del impacto de la docencia

A fecha de redacción de esta ponencia no se ha podido valorar de un modo global este proyecto de docencia. Por razones de tiempo, los alumnos solamente han realizado algunos de los módulos descritos en esta ponencia, como la programación con simulador y la depuración código de placa, tras suministrarle todos los diagramas descritos aquí. No obstante, se planea extender estos contenidos a asignaturas de Diseño de Sistemas Digitales, Proyecto o Informática Industrial en los estudios de Ingeniería Eléctrica, Ingeniería Técnica Industrial e Ingeniería en Informática Especialidad de Sistemas, pudiendo entonces elaborar una distribución en horas por módulo según la cantidad de créditos a cubrir.

Sin embargo, los alumnos han mostrado un notable interés por este proyecto y por las prácticas que han realizado. Agradecen trabajar siguiendo los pasos de diseño y fabricación de dispositivos que se están comercializando con bastante éxito, e incluso llegan a pedir más documentación o analizar más la placa.

Desde los autores consideramos altamente instructivo esta propuesta pues los alumnos tienen una visión global de lo que supone un proyecto de colaboración con una empresa, englobando numerosas áreas como la electrónica de potencia, electrónica digital, fabricación de circuitos impresos, elaboración de documentación,...

5. Conclusiones

Se ha propuesto un proyecto de docencia en asignaturas de Informática Industrial. El objetivo de este plan es guíar a los alumnos por las diferentes fases de un proyecto de colaboración con empresa. Este proyecto está basado en un proyecto real de la ULL con una empresa del entorno.

Los alumnos van pasando por diversos módulos que reflejan las diferentes fases: planificación de tareas, diseño de prototipo, elección de los componentes, diseño del circuito impreso, desarrollo del software, obtención de la placa, mejoras en el prototipo,...

Aunque no ha sido puesto en marcha todos los módulos, la opinión entre los alumnos es positiva. Por otra parte, los autores consideran este proyecto altamente instructivo pues los alumnos tienen una visión global de lo que supone un proyecto de colaboración con una empresa englobando un gran número de disciplinas.

Desde aquí se anima al lector a aprovechar proyectos similares para fines docentes.

Agradecimientos

Los autores desean mostrar su agradecimiento a los componentes del servicio técnico del Departamento de Física Fundamental y Experimental, Electrónica y Sistemas de la ULL, D. Roberto Betancor Bonilla, D. Manuel Fernández Vera y D. Agustín Padrón Name por sus horas de trabajo en este proyecto. Asimismo queremos agradecer a D. José León González, representante de la empresa TOTALBAR S.L., originaria del proyecto, por su confianza en nosotros para sacar adelante este producto.

Referencias

[1] ATMEL. Página principal: http://www.atmel.com

[2] Cos Castillo, Manuel de *Ingeniería de Proyectos*. Madrid, 1993.

[3] Page J., Delgado A., Blanco C. *Banco de CAD-CAM del laboratorio de circuitos de alta frecuencia de la ETSIT-UPM.* Actas URSI'96. http://www.etc.upm.es/alcaf.htm

Propuesta para la Integración de Prácticas de Laboratorio en Intensificaciones de la titulación de ITIS[1]

Vicent Lorente, Sílvia Terrasa, Salvador Petit y Alfons Crespo
Departamento de Informática de sistemas y computadores
Universidad Politécnica de Valencia
{vlorente, sterrasa, spetit, alfons}@disca.upv.es

Resumen

Los cambios en los planes de estudios de las ingenierías técnicas hacen que el profesorado en general se plantee cambios en la forma de impartir la docencia. En el presente artículo se propone la utilización de una plataforma común para la realización de actividades prácticas para asignaturas pertenecientes a una misma intensificación. Esto permite, por una parte, aunar esfuerzos a la hora de realizar la puesta en marcha de las prácticas, y, por otra, ofrecer al alumno una visión global a un problema complejo.

1. Introducción

En la actualidad, la Universidad Politécnica de Valencia está efectuando, desde el pasado curso 2001-2002, cambios en los planes de estudio de algunas titulaciones técnicas. Dos de estas titulaciones son las de Ingeniero Técnico en Informática de Gestión (ITIG) y la de Ingeniero Técnico en Informática de Sistemas (ITIS). Uno de los cambios más interesantes, desde el punto de vista de la formación del alumnado, es la aparición del concepto de intensificación. Con éste término, lo que se intenta es ofrecer al alumno la posibilidad de realizar una especialización durante el último curso que pueda acreditarse de alguna forma. Para ello lo que se intenta es reunir asignaturas optativas con una temática común. La idea es que cuando un alumno decida elegir una intensificación, deberá cursar obligatoriamente las asignaturas centrales de la intensificación (normalmente tres asignaturas) y luego escoger dos entre un conjunto de entre 6 a 8 asignaturas de temática relacionada para cubrir un mínimo de créditos de la intensificación y que esta se pueda acreditar.

Esta nueva forma de concebir las materias optativas hace que muchos profesores intenten aunar esfuerzos para efectuar su docencia de la forma más adecuada. Hasta ahora, las asignaturas optativas eran en cierta forma asignaturas independientes, en las que sus profesores podían cambiar contenidos o reestructurarlos sin que ello tuviera mayores consecuencias. Sin embargo, a partir de ahora se deberá actuar con más cautela, y se tendrán que revisar los contenidos de las asignaturas optativas de una determinada intensificación para asegurar que ni se repiten conceptos, ni se queda ningún concepto por impartir.

A parte de lo que es meramente teoría, uno de los objetivos más interesantes es que las sesiones prácticas sean lo más completas posibles, intentando cubrir todos los objetivos prácticos de la intensificación. Para poder llevar a cabo este objetivo de forma adecuada es necesario tener en cuenta las limitaciones temporales que plantean estas asignaturas, ya que, en el mejor de los casos, sólo disponen de dos horas de laboratorio a la semana, y la mayoría sólo disponen de dos horas de laboratorio cada quince días. Estas limitaciones hace que, en muchas ocasiones las prácticas no cubran todos los aspectos necesarios. Por otra parte, también suelen existir limitaciones de

[1] Ingeniero Técnico en Informática de Sistemas

espacio y de dinero a la hora de adquirir un material de prácticas un poco más sofisticado.

El presente artículo propone un método de diseño de las sesiones prácticas para que las limitaciones antes expuestas se vean aliviadas. La idea fundamental es la de utilizar el mismo material de soporte de prácticas en varias de las asignaturas de la intensificación. De esta manera, se pueden unir los esfuerzos de los profesores implicados para, por una parte, conseguir un material de apoyo más sofisticado y robusto, y, por otra parte, montar prácticas relacionadas entre si que cubran todos los aspectos interesantes de la intensificación en cuestión.

La propuesta está orientada a una intensificación en concreto, la intensificación de Informática Industrial de la titulación de ITIS. Se proponen, a modo de ejemplo, unas prácticas combinadas para las asignaturas de Sistemas de tiempo real, Automática Industrial y Control y la de Sistemas Robotizados, todas ellas pertenecientes a esta intensificación.

El resto del capítulo se organiza como sigue: en la sección 2 se describe el marco de las intensificaciones de la titulación de ITIS, y en concreto de la intensificación de informática industrial. La sección 3 está dedicada a describir el material de soporte que se va a utilizar en las sesiones prácticas. La sección 4 se plantearan las sesiones prácticas de las distintas asignaturas de la intensificación. Finalmente en la sección 5 se proporcionaran algunas conclusiones.

2. Intensificaciones de la titulación de ITIS.

En los planes de estudio que se están poniendo en práctica en la Escuela Superior de Informática Aplicada de la UPV[2] , el número de créditos optativos que tiene que cursar un alumno para la obtención del título de Ingeniero Técnico en Informática de Sistemas es de 34,5. La totalidad de dichos créditos se encuadran en el tercer curso de acuerdo al plan de estudios. Ello permite ofrecer un marco de intensificaciones destinado a imprimir un perfil más profesional a los futuros titulados.

Las materias optativas están agrupadas por intensificaciones. Cada intensificación está constituida por un Núcleo de Intensificación (18 créditos) y un conjunto de materias optativas afines.

Para obtener el título de ITIS un alumno deberá cursar obligatoriamente al menos un Núcleo de Intensificación perteneciente a una de las siguientes intensificaciones:

- Administración de Sistemas y Redes
- Informática Industrial.
- Ingeniería de computadores.
- Multimedia.
- Tecnologías y Servicios Web

A continuación se pasa a describir más en profundidad la intensificación de Informática Industrial, objeto del presente artículo.

2.1. Intensificación de Informática Industrial.

La intensificación de Informática Industrial tiene como objetivo principal el formar a profesionales con sólidos conocimientos de las técnicas, dispositivos y herramientas propias del ámbito industrial que le capaciten para la especificación, diseño, montaje, depuración y mantenimiento de sistemas informáticos de control y su integración en el ámbito de las redes industriales de área local, así como el desarrollo de aplicaciones de tiempo real y de software en general para el control de procesos industriales a través de computador.

Como se ha comentado anteriormente, todo alumno que desee cursar la intensificación de Informática Industrial, ha de cursar obligatoriamente las asignaturas correspondientes al núcleo de la intensificación y al menos 2 asignaturas de las materias complementarias de la misma Las asignaturas correspondientes a esta intensificación son:

Núcleo de la intensificación:

- Automática Industrial y Control.
- Sistemas de Tiempo Real.

[2] Universidad Politécnica de Valencia

- Sistemas de Entrada / Salida.

Materias complementarias de la intensificación:

- Análisis aplicado.
- CAD / CAM
- Configuración, administración e interconexión de redes de área local.
- Control estadístico de calidad.
- Diseño de sistemas lógicos.
- Fundamentos físicos de la robótica.
- Gestión y mantenimiento de empresas industriales.
- Periféricos e interfaces.
- Sistemas robotizados.
- Técnicas de inspección y de mantenimiento.

Como se comentó en la introducción la propuesta que se realiza en este artículo está relacionada con tres de las asignaturas de esta intensificación, dos de ellas pertenecientes al núcleo (Sistemas de tiempo real y Automática industrial y control), y la tercera perteneciente a las materias complementarias (Sistemas robotizados). En las siguientes secciones se describe las prácticas propuestas.

3. Diseño y montaje de los prototipos hardware.

A la hora de abordar el reto de realizar prácticas conjuntas, lo primero que se debía tener claro era el material que se iba a utilizar. Lo que se intentaba era disponer de un material común sobre el cual realizar distintas actividades. Debido a la temática de las asignaturas se optó por el montaje de la figura 1. Como se puede observar, el montaje consta de dos brazos robots, de cuatro grados de libertad y de una cinta transportadora.

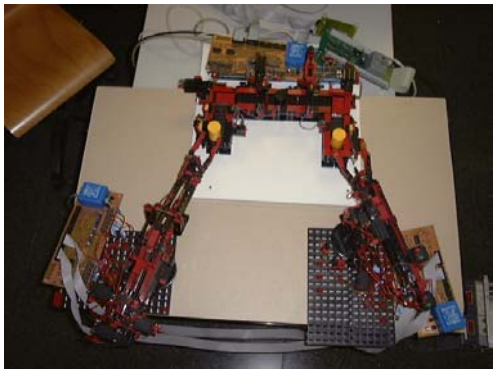


Figura 1. Montaje propuesto para las prácticas.

Como se ha comentado, los brazos robot disponen de cuatro articulaciones, cada una de ellas conectadas a un motor. También dispone de los finales de carrera para cada articulación (figura 2). Todos la entradas y las salidas del brazo robot son digitales.

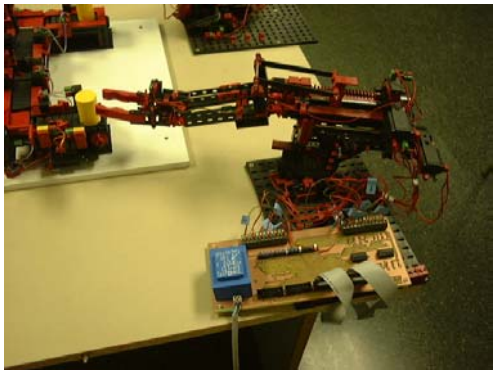


Figura 2. Brazo Robot.

La cinta transportadora (figura 3), por su parte dispone de sensores de posición así como de mecanismos para desplazar los objetos que se transporten. Al igual que en los robots todas las entradas y salidas son digitales.

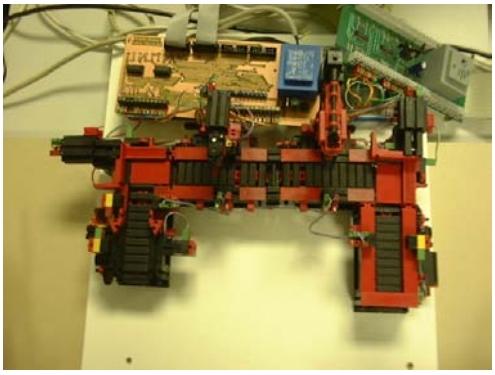


Figura 3. Cinta transportadora.

Para cada uno de los elementos (los dos brazos robots y la cinta transportadora) se han diseñado y realizado una serie de prototipos *hardware* para hacer posible la conexión a una tarjeta de adquisición de datos (PCLAB), de forma que sea posible su control desde un ordenador.

Este montaje nos permite disponer de un abanico bastante amplio de actividades prácticas. Por una parte, para la asignatura de sistemas de tiempo real, el simple hecho de disponer de un proceso físico a controlar ya proporciona el marco adecuado para la realización de sus prácticas, tal y como se veía en [1]. Además, con este montaje, disponemos de tres procesos distintos que pueden cooperar en un momento dado.

Respecto a la asignatura de Automática industrial y control, se pueden realizar muchas actividades relacionadas tanto con la fase de diseño de control, diagramas de estado, etc, hasta la implementación de algoritmos de control para controlar el proceso de producción.

Finalmente, en la asignatura de Sistemas Robotizados el simple hecho de disponer de brazos robots ya supone un marco ideal para realizar las actividades prácticas, ya que se puede desde realizar el modelado de los brazos robots hasta resolver el problema de la cinemática directa o inversa, y luego comprobar sobre el propio robot le funcionamiento de las ecuaciones calculadas.

4. Planteamiento de las sesiones prácticas según la asignatura.

Una vez descrito el marco común de las actividades prácticas, en este apartado se va a describir las actividades prácticas para cada una de las asignaturas.

4.1. Sistemas de tiempo real.

El descriptor de la asignatura de sistemas de tiempo real dice que los objetivos de la misma son:

- Saber las características de los sistemas de tiempo real (STR) y la de los sistemas operativos de tiempo real.
- Conocer los requisitos de los lenguajes para STR.
- Ser capaz de programar en Ada.
- Conocer tanto las metodologías de diseño de los STR como conceptos sobre su planificación.

Como se comentaba en [1], la evaluación de la asignatura se realiza mediante un trabajo de asignatura que consiste en el control de un proceso físico real. A este respecto, el montaje descrito en la sección anterior es un buen ejemplo de proceso a controlar.

Para la elaboración tanto del trabajo como de las actividades prácticas, se dispone de un entorno cruzado de desarrollo de aplicaciones empotradas, proporcionado por el sistema operativo de tiempo real MarteOS desarrollado por el departamento de Electrónica y Computadores de la Universidad de Cantabria [3]. Este entorno nos ofrece la posibilidad de desarrollar código en un entorno Linux y luego ejecutarlo en un sistema operativo de tiempo real (para más información sobre este punto consultar [2]).

Las actividades prácticas se deben organizar de forma que la dificultad vaya incrementándose de forma paulatina. De forma que se intente primero actuar de forma independiente sobre los distintos elementos del montaje. Una vez claro el comportamiento de los elementos por separado ya se plantea en una última práctica la el control coordinado de toda la plataforma.

En estas actividades se debe hacer especial hincapié en los aspectos propios de los sistemas de tiempo real, utilizándose para tal efecto un sistema operativo de tiempo real (MarteOS) y un lenguaje de programación también de tiempo real (Ada95).

4.2. Automática industrial y control.

En los descriptores de la asignatura de Automática industrial y control (AIC) aparece como objetivos principales los de :

- Conocer los controladores lógicos programables.
- Saber realizar el modelado y análisis de sistemas continuos, discretos y muestreados
- Entender el bucle de control por computador.
- Saber diseñar de reguladores discretos.
- Ser capaz de realizar una implementación práctica.

Si nos fijamos en el montaje de la figura 1, bajo el punto de vista de esta asignatura, se puede ver el conjunto como una planta industrial en la cual se ofrece la visión de un proceso industrial completo. En esta planta se pueden distinguir tres elementos:

1. Un brazo robot destinado a alimentar el proceso con piezas,
2. Una cinta transportadora dedicada a transportar y manipular las piezas.
3. Un brazo robot destinado a ir apilando el producto final.

Con esta visión del montaje, las actividades prácticas se pueden orientar de forma que los alumnos realicen primero un modelado del sistema discreto. A partir del modelado, sus conocimientos del control en bucle cerrado, sería interesante que fueran diseñando y aplicando distintos tipos de reguladores con el fin de realizar el control de la planta.

4.3. Sistemas robotizados.

Finalmente los descriptores de la asignatura de Sistemas Robotizados (SR) incluyen los siguientes objetivos:

- Conocer los componentes y la estructura de los sistemas robotizados.
- Ser capaz de programar robots.
- Saber como realizar el modelado y control de robots.
- Tener nociones sobre sensorización en robótica.
- Saber realizar aplicaciones.

Como se puede deducir de los objetivos de la asignatura, ésta intenta dar una visión de lo que es un sistema robotizado, desde los aspectos más técnicos (componentes y estructura) hasta los aspectos de más alto nivel (modelado y control). A este respecto, el montaje propuesto es ideal ya que es en sí mismo un sistema robotizado.

Las actividades prácticas propuestas para esta asignatura incluyen por orden de dificultad:

1. Realizar un modelado de los brazos robot y de la cinta transportadora.
2. Realizar un control de forma independiente de cada uno de los elementos.
3. Finalmente realizar un control coordinado.

La diferencia fundamental entre los algoritmos de control de esta asignatura y los vistos en la asignatura de AIC, son básicamente que los de esta asignatura están centrados en los SR, mientras que los de la asignatura de AIC son mucho más generales.

5. Conclusiones

Una de las ventajas que tienen los cambios de estudios en las titulaciones de informática es que obligan en cierta manera al profesorado a estar al día en cuanto a la temática de sus asignaturas optativas, a aparte de introducir cambios en la propia estructura del plan de estudios.

Con esta motivación, y gracias a los nuevos planes de estudio de la titulación de ITIS de la

Universidad Politécnica de Valencia, ha surgido un nuevo concepto: el concepto de intensificación.

El presente artículo ha intentado ofrecer una propuesta sobre cómo mejorar la docencia, al menos en cuanto a actividades prácticas se refiere, aprovechado este nuevo concepto de intensificación. Básicamente lo que se intenta es, por una parte, mejorar los recursos utilizados en las actividades prácticas ya que el hecho de compartir los montajes en distintas asignaturas puede hacer que éstos sean un poco más costoso y, además, hay más profesores implicados en la puesta de marcha de los mismos. Por otra parte, el alumno es capaz de relacionar mejor los conocimientos adquiridos en distintas asignaturas, pudiendo obtener una visión global del problema, ya que, aunque cada asignatura se centra en un aspecto concreto, el hecho de realizar las actividades prácticas sobre la misma plataforma le permite asociar conceptos más fácilmente.

Referencias

[1] S. Terrasa, P. Balbastre, A. Crespo. "La importancia del uso de procesos físicos reales en la enseñanza universitaria en la ingeniería". JENUI'2000, pp.304-310

[2] S. Terrasa, P. Balbastre, A. Crespo. "Experiencia docente en el desarrollo de aplicaciones empotradas con MarteOS", JENUI'2002, pp.304-310

[3] MarteOS : http://marte.unican.es

[4] "Titulación de Ingeniero Técnico en Informática de sistemas (plan 2001)" B.O.E. nº 259 de 17/10/2001

Sistemas distribuidos y paralelos

Domótica y Edificios Inteligentes en la Universidad de Alicante

Jorge Azorín, Andrés Fuster, Francisco Maciá, Francisco J. Ferrández
Dpto. Tecnología Informática y Computación
Universidad de Alicante
Apdo. Correos 99. E-03080. Alicante
e-mail: {jazorin,fuster,pmacia,fjferran@dtic.ua.es}

Resumen

En este artículo se presenta el enfoque de la asignatura Domótica y Edificios Inteligentes impartida en las Ingenierías Informáticas de la Universidad de Alicante. La propuesta subraya el papel del Ingeniero en Informática en el campo de la domótica, con un planteamiento motivado por la evolución de este campo de aplicación tecnológica hacia la integración de servicios de comunicaciones y computación. Inicialmente, se exponen las motivaciones de la incorporación de la asignatura a los planes de estudio y los objetivos orientados a la aplicación en este campo del cuerpo de conocimiento de los estudios de informática. A continuación se propone el contenido teórico y práctico con un marcado carácter aplicado, finalizando con una propuesta de evaluación continua del alumno.

1. Introducción

La incorporación de la asignatura Domótica y Edificios inteligentes se sitúa en la reforma de los planes de estudio de Informática que se produce en el curso 2001/2002 en la Universidad de Alicante [1][2][3]. En los nuevos planes de estudio de Informática, la optatividad representa un porcentaje importante en el total de créditos que han de cursar los alumnos, concretamente un 20% para la Ingeniería Informática, 18,66% para la Ingeniería Técnica en Informática de Sistemas y 16% para la Ingeniería Técnica en Informática de Gestión. Dentro de este plan de estudios y concretamente para el área de conocimiento Arquitectura y Tecnología de Computadores, se contemplan asignaturas optativas concebidas con carácter complementario al cuerpo de conocimiento impartido en materias obligatorias y troncales (Sistemas Operativos, Arquitectura de Computadores, redes, etc) y otras con un marcado carácter aplicado. Domótica y Edificios Inteligentes es una asignatura optativa de primer y segundo ciclo para las tres carreras de informática y forma parte del enfoque aplicado de las materias ofertadas a los estudiantes de informática. Se trata de una asignatura cuatrimestral y consta de tres créditos de teoría y tres de prácticas.

La domótica o el concepto más amplio de edificios inteligentes es un campo de aplicación tecnológica que está irrumpiendo con fuerza desde hace relativamente muy poco tiempo y donde el Ingeniero en Informática tiene un campo de trabajo poco explotado en la actualidad. Se trata de un campo multidisciplinar, si bien, en el enfoque clásico se ha tratado como un problema de automatización. Estos hechos están presentes en el contexto de la universidad española (ver tabla 1), donde solamente dos universidades, la Universidad de Oviedo [4] y la Universidad Politécnica de Madrid [5], ofertan una asignatura específica para este campo, aunque con un planteamiento distinto del que nosotros queremos ofrecer.

El enfoque que nosotros planteamos está motivado principalmente por la transformación que este campo está experimentado: desde un enfoque de automatización no estructurado hacia una integración de servicios de procesamiento de información y comunicaciones en el marco de las IST (Information Society Thecnologies – Tecnologías de la Sociedad de la Información) [6]. En esta transformación, el Ingeniero en Informática tiene un papel fundamental. La asignatura pretende fomentar esta visión, marcando unos objetivos y contenidos que doten al alumno de conocimiento, habilidades y

Universidad	Asignatura	Créditos	Contenido
U. Oviedo	Domótica y Edificios Inteligentes (2002)	4.5 + 1.5	Nociones generales de electrotecnia. Estudio de componentes y sistemas para la gestión técnica de la edificación
U. Politécnica de Madrid	Domótica y Edificios Inteligentes (1996)	3 + 1.5	Generación y transporte. Red de baja tensión. Iluminación. Electricidad estática. Arquitecturas del sistema de control de un edificio. Sistema de climatización. Sistema de detección y protección contra incendios. Sistema de seguridad. Sistema de control de ascensores. Estudio de proyectos concretos de control de edificios.

Tabla 1. Domótica y Edificios Inteligentes en otras universidades.

actitudes dentro de este campo concreto. El conocimiento que se aporta al alumno, en algunos temas, no es novedoso sino fundamentalmente de aplicación de modelos y técnicas propias de la disciplina informática en el contexto del control inteligente de viviendas.

En los siguientes apartados mostraremos la motivación y los objetivos propuestos para la asignatura, expondremos los contenidos teóricos y prácticos, y por último, finalizaremos el artículo con las conclusiones que se han extraído de la experiencia docente.

2. Motivación y objetivos

La domótica es un campo de naturaleza multidisciplinar donde áreas como la electrónica, la automática, las comunicaciones y la computación aportan parte de su conocimiento. La aplicación de estos cuerpos de conocimiento tiene como objetivo proporcionar servicios en el entorno del hogar y de forma más genérica en la edificación. Este campo se entiende como el área de automatización de las actividades y los sistemas del hogar y de incorporación de inteligencia en los edificios.

El ámbito de la domótica[1] según su enfoque clásico se limita a la automatización puntual de servicios a pequeña escala, lo que lleva asociado una falta de estructura. Sin embargo, el ámbito de la domótica es mucho más amplio y más abierto que la automatización de edificios. El enfoque actual hace énfasis en la integración de servicios generales de procesamiento de información y de comunicaciones dentro del marco de las IST. Esta tecnología debe abordar la automatización de los sistemas del hogar actuales y soportar los servicios futuros relacionados con el trabajo, la salud, la educación, etc (e-work, e-health, e-learning,…)

Las características generales de los sistemas domóticos son:

- Las mismas de los sistemas generales de redes de computadores: estructuración, modularidad, transparencia (encapsulación), escalabilidad.
- Comprenden todos los niveles, desde la capa física hasta la interfaz de usuario.
- Constituyen sistemas informáticos y de comunicaciones multidisciplinares, con concurso de todas las áreas de la informática y de otras áreas.
- Los subsistemas, los dispositivos y los componentes son muy diversos y hay gran heterogeneidad en requerimientos, prestaciones y costes.

La gran evolución que han experimentado las IST, y concretamente la disciplina informática, ha repercutido en todos los ámbitos de la sociedad. Los sistemas informáticos presentan nuevas posibilidades a la hora de abordar cualquier tipo de problema. Estas posibilidades han marcado la evolución de los sistemas domóticos.

El nuevo planteamiento requiere la revisión y diseño de nuevos objetivos. El alumno debe tener conciencia de la evolución de esta tecnología y el papel que el Ingeniero en Informática desempeña. Debe conocer cuales son las tecnologías que existen actualmente y ser capaz de enfrentarse a la especificación y diseño de estos sistemas

[1] Según el texto literal del diccionario de la Real Academia Española, domótica se define como: (Del lat. *domus*, casa, y *informática*). 1. f. Conjunto de sistemas que automatizan las diferentes instalaciones de una vivienda.

domóticos. Concretamente, los objetivos generales de la asignatura Domótica y Edificios Inteligentes los detallamos a continuación:

- Ubicar el área domótica en el marco de las IST.
- Introducir los conceptos generales de la domótica y su ámbito de aplicación.
- Estudiar la metodología general de diseño de los sistemas domóticos.
- Conocer los subsistemas domóticos y los dispositivos de percepción y actuación.
- Estudiar los estándares actuales y las tecnologías de comunicación en edificación.
- Diseñar un sistema domótico basado en agentes.
- Plantear sistemas avanzados y líneas futuras de desarrollo

3. Contenidos teóricos

Con el fin de cubrir los objetivos que se han marcado en la sección anterior, el contenido teórico de la asignatura se desarrolla en tres bloques temáticos. A continuación describimos cada uno de los bloques y los temas que comprende.

Bloque I. Generalidades

Este bloque introduce al alumno en la domótica explicando qué es lo que se espera de esta tecnología y cuál es el papel que desempeña el Ingeniero en Informática en este contexto. También, se cubren los aspectos legales.

Tema 1. Ámbito de la domótica

Objetivos: Introducir las metas de la Sociedad de la Información, revisar las tecnologías integradas en la IST y ubicar la domótica en la IST.

Contenido: Conceptos. Ámbito. Características de los sistemas domóticos. Criterios de diseño.

Tema 2. Marco legal y competencial

Objetivos: Introducir los conceptos generales de la domótica y su ámbito de aplicación, revisar el marco legal de la domótica y ubicar la competencia del Ingeniero en Informática.

Contenido: Marco legal. Competencias. El Ingeniero en Informática y la domótica.

Bloque II. Tecnología domótica

El bloque II se presenta como el de mayor carga de la asignatura contando con más del 50% del total de créditos teóricos. Aquí se analizan los distintos servicios que se esperan de un sistema domótico (confort, seguridad, comunicaciones y ahorro energético) incidiendo en los aspectos tecnológicos de captura de información y actuación. Más tarde, se revisa la arquitectura de control de un edificio mediante las normas y los estándares específicos existentes en el mercado, mostrando, también, los diferentes sistemas implantados con dichas normas. Una vez revisado y valorado el estado actual de esta tecnología, identificando sus virtudes y debilidades, entramos en uno de los temas fundamentales: las redes del hogar. Las comunicaciones juegan un papel muy importante en la evolución de la tecnología domótica. En este tema se introducen las necesidades de las redes del hogar, estudiando las distintas tecnologías existentes: adecuación de las redes de datos clásicas (ethernet) y las redes específicas del hogar (mediante cable eléctrico, teléfono,...). Acabamos el bloque con la integración de distintas tecnologías de comunicaciones y las especificaciones middleware.

Tema 3. Sistemas domóticos y componentes

Objetivos: Estudiar los servicios de los subsistemas domóticos revisando los sensores y actuadores utilizados, y revisar las particularidades de sensorización y actuación de cada subsistema.

Contenido: Análisis del problema. Subsistemas domóticos. Adquisición. Actuación.

Tema 4. Normativa

Objetivos: Entender la automatización y el control como un subconjunto de los servicios de un edificio inteligente, entender la arquitectura del sistema de control de un edificio, revisar las normas y estándares del mercado para el control de edificios, y valorar el estado de la tecnología actual identificando debilidades.

Contenido: Home Automation & Networking. Arquitectura de Sistemas domóticos: topología, soporte físico, nodos de control y protocolos (sistemas por corrientes portadoras y sistemas en bus).

Tema 5. Sistemas domóticos actuales

Objetivos: Estudiar diferentes sistemas implantados en la actualidad con los estándares estudiados y estudiar los servicios de estos sistemas domóticos.

Contenido: Sistemas basados en estándares de buses domóticos. Sistemas propietarios. Sistemas basados en autómatas.

Tema 6. Redes domóticas

Objetivos: Introducir las necesidades de redes en el hogar, discutir los objetivos del diseñador de redes en el hogar, estudiar las tecnologías de comunicación en la edificación, conocer especificaciones de alto nivel para red del hogar, y valorar las ventajas e inconvenientes de las diferentes tecnologías.

Contenido: Justificación e impacto de las redes. Arquitectura de las redes del hogar. Alternativas tecnológicas. Soluciones de alto nivel.

Bloque III. Especificación y diseño

En el último bloque, el alumno ya es consciente de los objetivos y de la evolución domótica, y conoce las herramientas tecnológicas disponibles. Por tanto, es momento de plantear nuevos paradigmas más robustos, expresivos y potentes para poder abordar la concepción y el diseño de este tipo de sistemas utilizando metodologías más cercanas a las IST. Para ello, en el tema siete presentamos el paradigma de agentes como modelo para especificación y diseño de sistemas heterogéneos, proporcionando al alumno las claves para el diseño de un sistema domótico basado en agentes. Finalizamos el contenido teórico de la asignatura pretendiendo que el alumno conciba los servicios del hogar desde una perspectiva global, incidiendo en los sistemas domóticos avanzados y servicios de alto nivel. También, en este último tema, los alumnos exponen los trabajos que han elaborado (en el apartado de evaluación mostraremos la finalidad de los mismos).

Tema 7. Modelado mediante agentes

Objetivos: Entender la problemática en el diseño de un sistema domótico, introducir el modelo de agentes y sistemas multiagentes, y diseñar un sistema domótico basado en agentes.

Contenido: Problemática en el diseño. Agentes. Sistemas multiagente. Modelado del sistema.

Tema 8. e-Home

Objetivos: Estudiar sistemas domóticos avanzados y servicios de alto nivel.

Contenidos: Los contenidos de este último tema versan sobre aplicaciones o modelos avanzados de domótica, ya sean tecnologías específicas o tecnologías existentes en otras áreas que se aplican con éxito en este campo (biometría, realidad virtual, robótica, etc).

En la tabla 2 se muestra la distribución temporal de la materia impartida en teoría.

4. Prácticas de laboratorio

Debido al carácter aplicado del conocimiento que debe poseer el alumno con respecto a la domótica, el componente práctico desempeña un papel importante, concretamente, el 50 % de los créditos totales de la asignatura.

Dentro de los objetivos generales, en el laboratorio se pretende reforzar el contenido descriptivo, haciendo hincapié en las habilidades y actitudes del alumno frente a los problemas que se presentan en la domótica. Para ello se proponen seis prácticas donde el alumno se enfrente a la especificación de requerimientos de un sistema domótico y al diseño de distintas partes del mismo. También, incidimos en los aspectos

Bloque	Tema	Duración
Generalidades	Tema 1. Ámbito de la domótica	2 h.
	Tema 2. Marco legal y competencial	2 h.
	Subtotal	**4 h.**
Tecnología domótica	Tema 3. Sistemas domóticos y componentes	4 h.
	Tema 4. Normativa	4 h.
	Tema 5. Sistemas domóticos actuales	4 h.
	Tema 6. Redes domóticas	6 h.
	Subtotal	**18 h.**
Especificación y diseño	Tema 7. Modelado mediante agentes	4 h.
	Tema 8. e-Home	4 h.
	Subtotal	**8 h.**
	Total	**30 h.**

Tabla 2. Cuadro resumen contenido teórico de Domótica y Edificios Inteligentes

tecnológicos de los estándares existentes en el mercado. Para llevar a cabo estos objetivos utilizamos distintas herramientas y recursos para el modelado e implementación de estos sistemas.

Con el fin de incidir en el diseño general utilizamos Simulink® [7]. Ésta es una herramienta interactiva que permite, a partir de la construcción de diagramas de bloques gráficos, modelar, analizar y simular sistemas dinámicos de tiempo continuo y discreto. Simulink®, concretamente, es una *toolbox* de MatLab®. El concepto de *toolbox* se debe a que es una colección especializada de funciones que permiten trabajar en clases particulares de problemas. Por tanto, Simulink® añade muchas características específicas a los sistemas dinámicos, mientras conserva toda la funcionalidad de propósito general de MatLab®. Sin embargo, el entorno de trabajo que proporciona, un entorno de trabajo visual, permite el uso de esta toolbox sin necesariamente conocimientos previos de MatLab®.

Otra de las herramientas que utilizamos es el sistema operativo QNX® [8]. Las características más destacables de este sistema operativo son: arquitectura de micro-kernel con API POSIX que permite implementar sistemas embebidos, escalabilidad, tiempo real e integración de los recursos de la red. El propósito fundamental de este sistema operativo es proporcionar sistemas abiertos de una forma robusta y escalable adecuada para una amplia gama de sistemas, desde sistemas de recursos restringidos a entornos de computación distribuidos. Todas estas características hacen de QNX® un soporte ideal para la arquitectura de un sistema domótico.

Como caso de estudio de plataforma Middleware se propone JINI™ [9]. La tecnología JINI™ está construida a partir de JAVA mediante librerías de clases. Esta tecnología afronta el problema de computación distribuida utilizando un conjunto simple de interfaces y protocolos. Permite crear *federaciones* de recursos (dispositivos, datos, aplicaciones, etc) disponibles a través de la red de forma transparente al usuario. En general puede ser utilizado para compartir tanto servicios software como hardware, ambos representados como objetos y accesibles mediante Interfaces Java. Las características más importantes y relacionadas íntimamente con los sistemas domóticos son la capacidad para transformar una red heterogénea de dispositivos en una homogénea, ser independiente de la arquitectura de los dispositivos aislados y la adaptación dinámica a los dispositivos.

Como dispositivos, disponemos del IPC@CHIP® Single Chip Embedded-Webserver de Beck [10]. Se trata de un controlador embebido diseñado para aplicaciones en red. El controlador incorpora el hardware y software necesario para tal fin. El hardware consiste en una CPU 186 de 16 bits, memoria RAM y Flash, Ethernet, Watchdog y detección de falta de alimentación. El software preinstalado es un sistema operativo en tiempo real (RTOS) con sistema de ficheros, la pila TCP/IP, un servidor Web, Ftp y Telnet. Este controlador está montado en el kit DK-40 de la misma empresa, mediante el cual tenemos 8 entradas y salidas digitales, 2 interfaces puertos serie (TTL) e interfaz Ethernet (10BaseT). Las características de este controlador permite el uso de interfaces remotas para el control de distintos dispositivos.

Una vez analizadas las herramientas y recursos que utilizamos, a continuación, mostramos el contenido práctico de la asignatura describiendo cada una de las prácticas:

Práctica 1. Recopilación de información

La primera práctica está destinada a que el alumno recopile información sobre la domótica y términos afines. Con esta labor, pretendemos iniciar al alumno en qué es la domótica, cuando y por qué se aplica este término. El estudiante en todo momento ha de ser capaz de valorar la información recopilada.

Para llevar a cabo esta tarea, se plantea la redacción de un informe valorando las referencias recopiladas y clasificándolas según el criterio que estime más oportuno (empresas, organizaciones, estándares, investigación, normativa legal, etc).

Práctica 2. Introducción a Simulink

Antes de abordar el trabajo de la siguiente práctica, es necesario que el alumno disponga de los conocimientos necesarios de la herramienta escogida para realizar el diseño general de un sistema domótico.

Con este objetivo, se dan las nociones básicas de Simulink® sin incidir en MatLab®. Concretamente, se repasan las librerías de que dispone la herramienta, reforzando los bloques que puede necesitar el alumno. A partir de aquí se

realizan una serie de modelos para que el alumno comprenda la metodología de modelado con esta herramienta. Acabamos la descripción de la herramienta con utilidades para la implementación del sistema (máscaras) y con la definición de bloques a partir de funciones de sistema, aplicando un lenguaje imperativo de programación (s-functions).

Práctica 3. Diseño de un sistema domótico mediante Simulink

El objetivo principal de esta práctica es que el alumno sea capaz de analizar el problema de implantar un sistema domótico para una vivienda unifamiliar. Que sea capaz de especificar los requerimientos para esa vivienda en concreto a nivel de los subsistemas de seguridad, confort, comunicaciones y de ahorro energético, y de diseñar el sistema oportuno.

Para la realización, se propone al alumno que diseñe el sistema domótico para una vivienda unifamiliar tipo con dependencias como dormitorios, salón, cocina, jardín, etc. En primer lugar debe especificar los requerimientos del sistema. La fase de diseño se divide en dos partes: la construcción de una librería domótica y el diseño concreto de la vivienda. En la fase de construcción de la librería se pide la construcción de cuatro sublibrerías: generación de eventos, control, adquisición y actuación. Con esto se persigue que el alumno se centre de forma independiente en las magnitudes a controlar (térmicas, eléctricas, ópticas, etc), el propio control (gestión a partir de la adquisición de información), la sensorización y la actuación. El nivel de detalle que se persigue es optativo, condicionado al interés del alumno, por ejemplo si queremos medir la temperatura, el bloque encargado podría realizarse mediante una función matemática que devolviese un valor con respecto a la temperatura, o por ejemplo, que el bloque implementase la adquisición mediante una resistencia NTC. En el diseño concreto de la vivienda se persigue la interconexión de los diferentes bloques diseñados por el alumno. Por último se pide al alumno que implemente las pruebas necesarias que determinen la corrección del sistema diseñado.

Práctica 4. Diseño de un sistema embebido para el control de servicios mediante un bus domótico

Los objetivos que nos marcamos para esta práctica son dos y serán abordados en diferentes niveles de realización. El primero de ellos es la aproximación a la utilización de un protocolo de control basado en bus domótico, que permitirá al alumno extrapolar los conocimientos particulares del protocolo, para comprender la filosofía de funcionamiento de los buses de control domótico. Se pretende estudiar tanto los protocolos de control como los controladores de los sistemas, así como sus interfaces. El segundo objetivo es la introducción al diseño de sistemas embebidos que permitirán el desarrollo de arquitecturas integradas, para la implementación de servicios de control concretos, que podrán ser ofrecidos en diferentes puntos de la red de control domótico. Estos sistemas embebidos podrán implementar tanto el control de la red con el protocolo pertinente como el interfaz con el usuario.

El contenido de la práctica plantea, para el primer objetivo, el estudio de X10, uno de los protocolos de control basados en bus domótico más difundidos. Éste utiliza la línea de corriente para la transmisión de las señales de control. Los controladores permitirán el envío de comandos a través de la red para implementar las funciones deseadas, existiendo interfaces para arquitectura PC que permiten la comunicación, monitorización y configuración del sistema. Los alumnos deberán realizar la implementación del protocolo y la interfaz para el controlador que comunica con el PC. Para el segundo objetivo, utilizando la implementación anterior, se pretende que el alumno sea capaz de abordar el diseño del protocolo y la interfaz para un sistema embebido utilizando QNX®.

Práctica 5. Desarrollo de servicios basados en tecnologías web

En la práctica cinco se fija como objetivo familiarizar al alumno sobre las posibilidades de integración de Internet en el desarrollo de redes de control especializadas en tareas de supervisión y control. Estas tecnologías ofrecen un valor añadido a las actuales soluciones propietarias que ofrece el mercado.

Para ello, se diseñará y realizará un prototipo, proponiendo una solución que utilice los protocolos, interfaces y lenguajes de programación sobre los que se sustentan las soluciones que propone el IPC@CHIP® para

servicios Web. A partir del kit DK-40 que como hemos comentado anteriormente presenta una matriz de entradas y salidas digitales, se propone la conexión de distintos dispositivos para comprobar su correcto funcionamiento.

Práctica 6. Modelado mediante agentes

Esta última práctica tiene como objetivo la especificación y el diseño del sistema utilizando agentes que permitan la conectividad con sistemas generales de procesamiento de información y de comunicaciones.

El alumno ha de ser capaz de especificar y diseñar los posibles agentes involucrados en tal objetivo. En esta práctica hacemos uso de la herramienta JINI™. La implementación de todo el sistema presentaría una amplia dilatación en el tiempo y probablemente fuera de los objetivos docentes, donde se enfatiza en la habilidad y actitud del alumno. Por esto, se pide la implementación de dos o tres agentes representativos del sistema.

Estas dos últimas prácticas que hemos descrito no fueron elaboradas en la primera propuesta de la asignatura. Sin embargo, consideramos oportuno la inclusión de éstas en el segundo año que impartiremos la asignatura para lograr una mayor conexión con el contenido teórico. En la tabla 3 se muestra la distribución de las prácticas de la asignatura y su duración junto con la distribución teórica.

5. Evaluación de la asignatura

El carácter descriptivo de la asignatura en su parte teórica indica la conveniencia de la realización de ejercicios por parte del alumno, que serán entregados para su revisión y evaluación. Este método permite la evaluación continua de la evolución del alumno. Para ello, al finalizar cada sesión de teoría se plantean una serie de casos prácticos (cuestiones y problemas) sobre la materia explicada. Las prácticas de laboratorio serán de asistencia obligatoria para observar la progresión de los alumnos, que además deberán entregar memorias del trabajo realizado. Además se propone la elaboración de trabajos optativos en relación a la materia impartida. Estos trabajos se caracterizan por presentar servicios domóticos de alto nivel que englobamos dentro del tema e-Home. Entre estos trabajos, en el curso anterior se presentaron *Sistemas de Reconocimiento de Voz, Robótica en el hogar y de consumo, Reconocimiento de huellas dactilares en el ámbito de la domótica, Entornos inteligentes (Realidad Virtual), Teletrabajo, etc.*

Toda esta tarea desarrollada de forma continua durante el curso, será evaluada y calificada siguiendo los siguientes criterios: la calificación final C_F de la asignatura se obtendrá de la nota obtenida en teoría, C_T, (evaluación continua y/o examen final), mediante la nota obtenida en prácticas, C_P, y por la realización del trabajo optativo, C_{OPT}.

Teoría	Práctica	Duración
Tema 1. Ámbito de la domótica	Práctica 1. Recopilación de información	4 h.
Tema 2. Marco legal y competencial		
Tema 3. Sistemas domóticos y componentes	Práctica 2. Introducción a Simulink	2 h.
Tema 4. Normativa	Práctica 3. Diseño de un sistema domótico mediante Simulink	6 h.
Tema 5. Sistemas domóticos actuales	Práctica 4. Diseño de un sistema embebido para el control de servicios mediante un bus domótico	6 h.
Tema 6. Redes domóticas	Práctica 5. Desarrollo de servicios basados en tecnologías web	6 h.
Tema 7. Modelado mediante agentes	Práctica 6. Modelado mediante agentes	6 h.
Tema 8. e-Home		

Tabla 3. Diagrama temporal de los contenidos prácticos de la asignatura con respecto al contenido teórico.

$$C_F = 0{,}4C_T + 0{,}4C_P + 0{,}2C_{OPT} \quad (1)$$

De esta forma, al ser una asignatura optativa el alumno puede aprobar la asignatura con la realización de los problemas y las prácticas. Sólo los alumnos que muestren un interés por la asignatura podrán optar a calificaciones de sobresaliente y matrícula de honor.

Aquellos alumnos que no superen la evaluación continua a partir de los supuestos prácticos o no la utilicen podrán realizar un examen final. La calificación de teoría se obtiene mediante la realización del examen final de la asignatura, C_E, o mediante la evaluación continua de los problemas planteados, C_{ECP}, al finalizar las sesiones teóricas.

$$C_T = \max\{C_E, C_{ECP}\} \quad (2)$$

Si el alumno opta por la realización de los casos prácticos, la calificación se reparte de forma pondera a la importancia del tema asociado, proporcional a la duración de los mismos. La calificación de las prácticas, también se reparte de forma pondera a la importancia y duración de las mismas.

6. Conclusiones

En este artículo se ha presentado el enfoque de la asignatura Domótica y Edificios Inteligentes de los planes de estudio de las Ingenierías Informáticas de la Universidad de Alicante. Se ha expuesto la motivación de la incorporación de esta asignatura y los objetivos, orientados por la transformación de la domótica hacia un enfoque con énfasis en la integración de servicios de procesamiento de información y de comunicaciones. Este enfoque da cabida al Ingeniero Informático en un campo donde no había intervenido tradicionalmente. El objetivo a nivel de transmisión de conocimientos tiene un marcado carácter aplicado. A nivel práctico los objetivos enfatizan las habilidades y actitudes del alumno que le permitirán enfrentarse a problemas de ámbito domótico. El método de evaluación propuesto, por su parte, permite una mayor relación profesor-alumno y una evaluación continua de la progresión.

La aceptación de la asignatura en el primer curso en el que se ha impartido ha resultado positiva. En el presente curso académico, la matriculación ha sido superior. Además, alumnos que cursaron la asignatura están realizando el proyecto fin de carrera sobre esta materia. La valoración de la experiencia docente en base a la acogida de los alumnos que está siendo muy positiva, unida a la aceptación a nivel de proyectos fin de carrera, provoca la reflexión sobre el interés de intensificar nuestra actividad investigadora en esta línea.

Referencias

[1] *Plan de Estudios conducente al título de Ingeniero en Informática de la Escuela Politécnica Superior de la Universidad de Alicante.* BOE número: 230-2001.

[2] *Plan de Estudios conducente al título de Ingeniero Técnico en Informática de Sistemas de la Escuela Politécnica Superior de la Universidad de Alicante*. BOE número: 230-2001.

[3] *Plan de Estudios conducente al título de Ingeniero Técnico en Informática de Gestión de la Escuela Politécnica Superior de la Universidad de Alicante*. BOE número: 230-2001.

[4] *Adaptación del Plan de Estudios de Ingeniero Técnico en Informática de Sistemas, de la Escuela Universitaria de Ingeniería Técnica Informática de Oviedo, a los Reales Decretos 614/1997, de 25 de abril, y 779/1998, de 30 de abril*. BOE número: 181-2002

[5] *Resolución de 25 de septiembre de 1996, de la Universidad Politécnica de Madrid, por la que se ordena la publicación del Plan de Estudios para la obtención del titulo de Ingeniero en Informática*. BOE número: 253-1996

[6] Programa de trabajo 2003-2004 del VI Programa Marco de la Unión Europea. *http://www.cordis.lu/fp6/home.html*.

[7] Documentación para productos de Mathworks. *Simulink*. *www.mathworks.es*

[8] Documento técnico. *System Architecture*. QNX RT Platform. *http://www.qnx.com/*

[9] Baker, M.; Juhasz, Z. *Distributed Computing with Java and Jini*. Euro-Par 2001. Manchester. 2001

[10] Schlösser, E.; Gatrost, J. *SC12 Getting Started*. Oct. 2001 *www.bcl.de*

Complejidad Algorítmica: de la Teoría a la Práctica

I. Dorta, C. León, C. Rodríguez, G. Rodríguez, A. Rojas
Universidad de La Laguna
Edificio de Física y Matemáticas
c/ Astrofísico Fco. Sánchez s/n
38271 La Laguna. Tenerife

Resumen

En este trabajo se presentan las herramientas CALL y LLAC que facilitan el análisis de complejidad de aplicaciones tanto secuenciales como paralelas. La descripción del modo de uso se realiza mediante el estudio de un caso práctico. El ejemplo elegido ha sido el Problema de la Mochila 0/1. La resolución del mismo se aborda mediante la técnica de Ramificación y Acotación, presentando dos opciones de implementación: recursiva y paralela.
La principal aportación del uso de estas herramientas en la docencia es establecer un puente entre el análisis teórico de la complejidad de los algoritmos y el análisis práctico de los parámetros de rendimiento de los programas, simplificando la labor de ejecución, recolección y estudio de resultados computacionales.

1. Introducción

En los planes de estudio vigentes en la Universidad de La Laguna se cuenta con asignaturas de contenidos relacionados con la Programación Paralela y Distribuida [4] que se concretan en las asignaturas optativas: *Programación en Paralelo I* y *Programación en Paralelo II* y *Programación Distribuida*.

En ellas, los alumnos deben realizar prácticas de laboratorio que implican el uso de los dos paradigmas de programación de máquinas paralelas estándar: el Paso de Mensajes y la Memoria Compartida. Ello supone la realización de experimentos que permitan confirmar que el algoritmo paralelo que se ha diseñado proporciona alguna ventaja frente al algoritmo secuencial. El uso de las herramientas CALL y LLAC facilita esta labor.

En este trabajo, se presenta un ejemplo para ilustrar cómo se usarían las herramientas CALL y LLAC en la docencia de "*Programación en Paralelo*". El problema que se ha considerado es el Problema de la Mochila 0/1 [3]. Se ha optado por un problema de optimización combinatoria porque tienen una gran trascendencia en todos aquellos sectores productivos y usuarios de software que se dediquen al desarrollo de aplicaciones de ayuda a la toma de decisiones y optimización de recursos. Dado que el objetivo de este trabajo es mostrar la utilidad del software en la docencia de optimización matemática, sólo se incluyen los conocimientos elementales necesarios para empezar a trabajar con ambos paquetes.

En el apartado siguiente se presentan las herramientas CALL y LLAC. En la tercera sección se define el caso de estudio y se presentan dos aproximaciones para su resolución: secuencial y paralela, se describe cómo se instrumenta el código y cómo interpretar los resultados. Finalmente se exponen las conclusiones y trabajos futuros.

2. Las herramientas CALL y LLAC

El análisis de complejidad de un algoritmo produce como resultado una "*función de complejidad*" que da una aproximación del número de operaciones que realiza. La herramienta CALL [5] permite anotar con dicha fórmula de complejidad el código C que implanta el algoritmo. Por ejemplo, para el clásico producto de matrices cuadradas C = A×B, donde A y B son matrices de dimensión N×N, el código de la llamada al procedimiento se anotaría de la siguiente forma:

```
#pragma cll mp mp[0] + mp[1]*N + \
               mp[2]*N*N + mp[3]*N*N*N
  MatrixProduct(A, B, C, N);
#pragma cll end mp
```

Donde C_0 = mp[0], C_1 = mp[1], C_2 = mp[2] y C_3 = mp[3] son las constantes asociadas a la fórmula de complejidad del algoritmo ($C_0 + C_1 N + C_2 N^2 + C_3 N^3$).

Los valores de dichas constantes, para una arquitectura dada, se calculan mediante regresión lineal por LLAC. LLAC es una herramienta estadística basada en R [2] que analiza los datos aportados por la ejecución del código instrumentado mediante CALL.

3. Caso de estudio: El Problema de la Mochila 0/1

Consideremos la siguiente definición del Problema de la Mochila 0/1:

"Se dispone de una mochila de capacidad C y de un conjunto de N objetos. Se supone que los objetos no se pueden fragmentar en trozos más pequeños, así pues, se puede decidir si se toma un objeto o si se deja, pero no se puede tomar una fracción de un objeto. Supóngase además, que el objeto k tiene beneficio b_k y peso p_k, para $k=1,2,...,N$. El problema consiste en averiguar qué objetos se han de insertar en la mochila sin exceder su capacidad, de manera que el beneficio que se obtenga sea máximo".

Su formulación como un problema de optimización es:

$$max \sum_{k=1}^{N} b_k x_k$$

sujeto a: $$\sum_{k=1}^{N} p_k x_k \leq C$$

$$x_k \in \{0,1\} \quad k \in \{1,...,N\}$$

La Ramificación y Acotación [1] es un método general que permite resolver un amplio rango de problemas de optimización combinatoria. La solución de un problema consiste en un vector de enteros que cumple un número de restricciones y optimiza una función objetivo. La función objetivo debe ser maximizada o minimizada.

Dado que el área de soluciones contiene un número de elementos exponencial, es imposible explorar todas las soluciones en un tiempo razonable para problemas grandes. La técnica de Ramificación y Acotación intenta reducir el número de soluciones factible por exploración sistemática del área de solución. Los algoritmos de Ramificación y Acotación dividen el área de solución paso por paso y calculan una cota del posible valor de aquellas soluciones que pudieran encontrarse más adelante. Si la cota muestra que cualquiera de estas soluciones tiene que ser necesariamente peor que la mejor solución hallada hasta el momento, entonces no necesitamos seguir explorando esta parte del árbol. Por lo general, el cálculo de cotas se combina con un recorrido en anchura o en profundidad.

En los párrafos siguientes se presentan dos implementaciones, una secuencial en C y una paralela con MPI, de un algoritmo que resuelve el Problema de la Mochila mediante la técnica de Ramificación y Acotación.

3.1. Implementación Secuencial

El Algoritmo 1 muestra el código para resolver el problema de forma recursiva. El número de objetos a insertar en la mochila se almacena en la variable *N*, en las variables *w* y *p* se guardan los pesos y los beneficios de cada uno de ellos, mientras que *M* representa la capacidad. Puesto que se trata de un problema de maximización se da el valor inicial de -∞ a la variable que almacena la mejor solución encontrada hasta el momento *bestSol* (líneas 1 a 4).

Entre las líneas 8 y 18 se define la función de acotación (*lowerUpper*). Se utiliza la misma función tanto para calcular la cota inferior como la cota superior. La cota inferior se define como el máximo beneficio que se puede obtener para un subproblema dado. Mientras que la cota superior incluye la parte proporcional del beneficio del último objeto que no se pudo insertar en la mochila.

La función *knap* (líneas 20-35) implementa el algoritmo de Ramificación y Acotación de forma recursiva. En la línea 29 se realiza la llamada que estudia la posibilidad de insertar el objeto k, mientras que en la línea 30 se considera su no inclusión. De las líneas 22 a la 26 se implementa la condición que actualiza los valores de la mejor solución (*bestSol*) encontrada hasta el momento.

Estamos interesados en conocer el comportamiento del algoritmo, concretamente la pregunta que nos gustaría responder es ¿cuántos nodos del espacio de búsqueda se visitan para encontrar la mejor solución?.

```
1     /* …ficheros de cabecera */
2     number N, M;
3     number w[MAX], p[MAX];
4     number bestSol = -INFINITY;
5
6     #pragma cll code double numvis;
7
8     void lowerUpper(number k, number C, number P,number *L, number *U) {
9       number i, weig, prof;
10      if (C < 0) {*L = -INFINITY; *U = -INFINITY;  }
11      else {
12        for(i=k, weig = 0, prof = P; weig <= C; i++)
13        {weig += w[i]; prof += p[i];}
14        i--;
15        weig -= w[i]; prof -= p[i];
16        *L = prof;  *U = prof+(p[i]*(C-weig))/w[i];
17      }
18    }
19
20    number knap(number k, number C, number P) {
21      number L, U, next;
22      if (k < N) {
23        lowerUpper(k,C,P,&L,&U);
24        if (bestSol < L) {
25          bestSol = L;
26        }
27        if (bestSol < U) { /* L <= bestSol <= U */
28          next = k+1;
29          knap(next, C - w[k], P + p[k]);
30          knap(next, C, P);
31    #pragma cll code numvis += 2 ;
32        }
33      }
34      return bestSol;
35    }
36
37    int main(int argc, char ** argv) {
38      number sol;
39      readKnap(data);
40    #pragma cll code double numvis = 0.0;
41    #pragma cll kps kps[0]*unknown(numvis) posteriori numvis
42                  /* obj. sig., capacidad rest., beneficio */
43      sol = knap(   0,          M,                0);
44    #pragma cll end kps
45      printf("\nsol = ", sol);
46    #pragma cll report all
47      return 0;
48    }
```

Algoritmo 2. Implementación Secuencial recursiva anotada

Para ello anotamos el código con directivas de CALL. La línea 6 le dice a CALL que se va a definir una variable de tipo `double` para almacenar el número de nodos visitados, *numvis*. A dicha variable se le da inicialmente el valor cero (línea 40). La directiva *code* de CALL proporciona la posibilidad de insertar código fuente al programa anotado. Este código no modifica el programa original, sólo se utiliza en las sentencias CALL. La cadena *cll* que se coloca después de la palabra reservada `#pragma` le indica al compilador de C que se trata de una directiva para el compilador de CALL. Si no se dispone del mismo, simplemente se interpreta como un comentario.

La línea 41 crea un experimento CALL típico. El identificador *kps* especifica su nombre. Este experimento mide el tiempo de ejecución de las sentencias entre las directivas de inicio y finalización (línea 44). El `pragma cll end` va seguido del nombre del experimento y hace que el cronómetro de CALL se pare, se guarden los tiempos y se prepare para la próxima ejecución.

La fórmula que sigue al identificador del experimento *kps* (línea 41) ha de sintetizar la

fórmula de complejidad que nosotros creemos que describe el comportamiento del tiempo. Asociadas a la fórmula se tienen las constantes ksp[0],…,kps[i]. La sintaxis de CALL requiere que se utilice el nombre del experimento para indexar las constantes asociadas con su fórmula de complejidad. Estas constantes pueden ser evaluadas con la herramienta LLAC.

Nuestra intuición nos dice que el tiempo invertido por el algoritmo de Ramificación y Acotación está relacionado con el número de nodos del espacio de búsqueda que se visitan. Así pues, nuestra fórmula se ha de escribir en términos de la variable *numvis* - número de nodos visitados. Sin embargo, el valor final de *numvis* sólo se conocerá una vez resuelto el problema, esto es cuando acabe la llamada a *knap(0,M,0)*. Por lo tanto, en la fórmula de complejidad se indica que dicho valor es desconocido (`unknown`) y que se evaluará al final (`posteriori`).

El número de nodos visitados (*numvis*) se incrementa cuando se tratan los dos nuevos subproblemas que se obtienen a partir del que se esté estudiando en cada momento, esto es, cuando recursivamente se resuelven los problemas que "si" incluyen al siguiente objeto (línea 29) y "no" lo incluyen (línea 30). Este incremento hemos de indicárselo a CALL insertando la directiva correspondiente en la línea 31.

Finalmente, se ha de añadir la directiva `report` después de la especificación de todos los experimentos. Esta directiva le dice al compilador de CALL que genere un fichero con todos los resultados obtenidos durante la ejecución. En nuestro caso, la colocamos justo antes de la sentencia `return` de la función *main* (línea 46). Se ha utilizado el calificador `all` que guarda los resultados de todos los experimentos definidos en el programa. Alternativamente, se puede especificar una lista con los identificadores de los experimentos de los cuales se quieren almacenar los resultados.

Una vez anotado el código fuente con las directivas de CALL se puede compilar con cualquier compilador de C y se seguirá ejecutando exactamente como si no se hubiera modificado. Esto es debido a que el compilador ignora todas las directivas CALL tratándolas simplemente como comentarios.

Supongamos que el Algoritmo 1 está almacenado en el fichero *kpr.c*. Procedemos a procesarlo con el compilador de CALL utilizando el comando:

```
> call kpr.c
```

Como resultado se producen dos ficheros de salida: *kpr.cll.c* y *kpr.cll.h*. Para saber qué código añade CALL al programa original sólo hay que echarle un vistazo a estos ficheros. Ahora se procede a compilar estos ficheros con un compilador de C, indicándole dónde están los ficheros de CALL que es necesario incluir (`/usr/local/CALL/include`):

```
>cc -o kpr kpr.cll.c
```

Al ejecutar el programa resultante (`kpr`) se obtiene un fichero con los resultados del experimento definido: *kpr.c.dat*. La Figura 1 muestra el contenido de este fichero. El programa se ejecutó sobre una mochila de capacidad M = 1.254.202 y número de objetos N = 5000. Fijemos nuestra atención en las dos últimas líneas del fichero. En ellas aparecen los resultados de nuestro experimento. Puesto que se trata de una ejecución secuencial, el número de CPUS usadas es 1. Por defecto, el nombre de la CPU es 0. El número de nodos visitados (*numvis*) ha sido de 261.134 y el tiempo en el que se ha ejecutado la llamada a la función *knap()* ha sido de 0,16 segundos. En el manual de usuario de CALL [5] se puede encontrar una descripción detallada del resto de la información que aparece en el fichero.

```
EXPERIMENT: "kps"
BEGIN_LINE: 115
END_LINE: 119
FORMULA: p 0 p 1 v 0  * +
INFORMULA: kps[0]+kps[1]*numvis
MAXTESTS: 131072
DIMENSION: 2
PARAMETERS:
NUMIDENTS: 1
IDENTS: numvis
OBSERVABLES: CLOCK
COMPONENTS: 1 numvis
POSTFIX_COMPONENT_0: 1
POSTFIX_COMPONENT_1: v 0
NUMTESTS: 1
SAMPLE:
  CPU  NCPUS  numvis    CLOCK
    0      1  261134.0  0.16491100
```

Figura 1. Contenido del fichero knr.c.dat

```
 1        busy[0] = 1; for i = 1, nProcs { busy[i] = 0;}

 2        idle = nProcs - 1;
 3        //Send initial subproblem to first idle slave
 4        auxSp = sp.initSubProblem();
 5        outputPackect.send(firstIdle,
 6                           auxSp,   // initial subproblem
 7                           bestSol, // bestSolution
 8                           sol);    // current solution
 9        idle--;
10        IDLE2WORKING(busy,firstIdle);  // mark this slave like working
11        while (idle < (groupSize-1)) { // while there are working slaves
12           recv(source, flag);
13           while(flag) {
14               if (SOLVE_TAG) {        // receive the final solution
15                 inputPacket.recv(source,
16                                  bestSol,  // best solution
17                                  sol);     // current solution
18               }
19               if (BnB_TAG) {              // receive a slave request
20                 inputPacket.recv(source,
21                                  high,     // upper bound
22                                  nSlaves); // num. of reuiered slaves
23                 if ( high > bestSol){      // problem to branch
24                   total= ((nSlaves <= idle)?nSlaves:idle);
25                   for i = 1, total { idle--; IDLE2WORKING(busy,i); }
26                   outputPacket.send(source,
27                                     total,  // num. of assigned slaves
28                                     bestSol,     // best Solution
29                                     1,..., total // slaves identifiers
30                                    );
31                 }
32                 else { // the problem must be bounded
33                   outputPackted.send(source, DONE);
34                 }
35               }
36               if (IDLE_TAG) {     // signal of an idle slave
37                 inputPacket.recv(source, IDLE);
38                 idle++;
39                 WORKING2IDLE(busy,source); // mark this slave like idle
40               }
41               recv(source, flag);
42           } // while (flag)
43        } // while (idle < (groupSize-1))
44        // Send the ending message
45        for i = 1, groupSize { outputPacket.send(i, END); }
```

Algoritmo 1. Implementación del Maestro

3.2. Resolución Paralela

La versión paralela del algoritmo de Ramificación y Acotación para resolver el problema de la mochila se ha implementado mediante Paso de Mensajes [6] siguiendo un esquema *Maestro/Esclavo*.

El *Maestro* (véase el Algoritmo 2) es el responsable de la coordinación entre tareas, para ello, cuenta con la estructura de datos *busy* donde registra el estado de ocupación de cada uno de los esclavos (línea 1). Al comienzo de la computación todos los esclavos se marcan como ociosos.

El subproblema inicial (*auxSp*), el mejor valor de la función objetivo (*bestSol*) y el mejor vector solución hasta el momento se envía al primer esclavo ociosos (líneas 3-10). Mientras existan esclavos libres el *Maestro* recibe información de ellos y decide la siguiente acción a ejecutar dependiendo de si el problema está o no resuelto (línea 14), si hay una solicitud de esclavos (línea 19) o si los esclavos no tienen nada que hacer (línea 36). Si el problema está resuelto se reciben

```
 1    while (1) {
 2      recv(source, flag);
 3      while (flag) {
 4        if (END_TAG){                 // receive the finishing message
 5          inputPacket.recv(MASTER, END); return;
 6        }
 7        if (PBM_TAG){                // the problem to be branched
 8           inputPacket.recv(source,    // source = slave or master:
 9                            auxSp,     //   the initial subproblem
10                            bestSol,   //   the best solution value
11                            sol);      //   the current solution
12           auxSol = sol;
13           bqueue.insert(auxSp);       // insert in the local queue
14           while(!bqueue.empty()) {
15             auxSp = bqueue.remove();  // pop from the local queue
16    #pragma cll code numvis++;
17            high = auxSp.upper_bound(pbm,auxSol);    // upper bound
18            if ( high > bestSol ) {
19              low = auxSp.lower_bound(pbm,auxSol);   // lower bound
20              if ( low > bestSol ) {
21                bestSol = low;
22                sol = auxSol;
23                outputPacket.send(MASTER,    // send to the Master:
24                                  SOLVE_TAG,    // problem solved
25                                  bestSol,      // best solution value
26                                  sol);         // solution vector
27              }
28              if ( high != low ) {
29                rSlaves = bqueue.getNumberOfNodes();
30                op.send(MASTER,
31                        BnB_TAG,
32                        high,                 // upper bound
33                        rSlaves);             // num. of slaves req.
34                inputPacket.recv(MASTER,
35                                 nfSlaves,   // num. of slaves assig.
36                                 bestSol,    // updated best solution
37                                 rank {1,..., nfSlaves});
38                if ( nfSlaves >= 0) {
39                  auxSp.branch(pbm,bqueue);   // branch
40                  for i=0, nfSlaves{          // send problems to slaves
41                    auxSp = bqueue.remove();
42    #pragma cll code numvis++;
43                      outputPacket.send(rank,  // send to the slave:
44                                        PBM_TAG, //   tag
45                                        auxSp,   //   problem
46                                        bestSol, //   best solution value
47                                        sol);    //   the solution vector
48                  }
49                } // if nfSlaves == DONE the problem is bounded (cut)
50           } } }
51           outputPacket.send(MASTER, IDLE_TAG);  /idle slave
52         }
53         recv(source, flag);
54       } // while (flag)
55    } // while(1)
```

Algoritmo 3. Implementación del Esclavo anotada

el mejor valor de la función objetivo y el vector solución y se almacenan (líneas 15-18). Cuando el *Maestro* recibe una solicitud de "*nSlaves*" esclavos libres, viene acompañada del valor de la cota superior (*high*). Si el valor de la cota superior es mayor que el valor actual de la mejor solución (*bestSol*) la respuesta al esclavo incluye el número de esclavos libres que pueden ayudar a resolver el problema (*total*) - líneas 26-30. En otro caso, la respuesta indica que no es necesario trabajar en ese subárbol de búsqueda (línea 33). Cuando el número de esclavos libres es igual al valor inicial, el proceso de búsqueda finaliza y el Maestro

notifica a todos los esclavos que dejen de trabajar (línea 45).

Cada *esclavo* (Algoritmo 3) trabaja acotando los problemas que recibe (líneas 8-12). Se generan nuevos subproblemas mediante llamadas a la función de ramificación *branch()* (línea 39). El *esclavo* pide información sobre esclavos libres al *Maestro* (líneas 30-33). Si no hay otros esclavos libres que le ayuden en su tarea, el *esclavo* continúa trabajando localmente. En otro caso, elimina subproblemas de su cola local y los envía directamente a los otros esclavos que le hayan asignado (líneas 39-47).

Al igual que en el caso secuencial, queremos estudiar el número de nodos visitados (*numvis*). Ahora este valor está distribuido entre los *esclavos*, porque el *Maestro* no realiza labores de acotación. Por lo tanto, anotaremos el código paralelo exactamente igual que el secuencial, cambiando sólo el lugar donde se incrementa el número de nodos visitados. Las directivas CALL que lo hacen se han añadido en los puntos en los que se extraen problemas de la cola local de cada esclavo ya sea para estudiarlo (línea 16) o para enviárselo a otro esclavo para que lo resuelva (línea 42).

3.3. Estudio de los resultados

La herramienta LLAC al ser una extensión de R, proporciona la posibilidad de hacer un análisis de las muestras que se obtienen al ejecutar los experimentos generados con CALL. Con las mismas muestras se pueden realizar distintos tipos de representaciones para estudiar cuanto se ajustan la fórmula con la que habíamos anotado el programa y los resultados obtenidos.

Las ejecuciones secuenciales del problema de la mochila se ejecutaron sobre un procesador AMD-Duron a 800 MHz y 256 Mb de memoria. El experimento consistió en generar aleatoriamente diez problemas de la mochila con capacidad en el rango [500, 5000]. La Figura 2, generada con LLAC, muestra los resultados obtenidos. Cada uno de los diez puntos redondos asociado a cada tamaño (500-5000) representa el número de nodos que se visitó para resolver ese problema. Las "x" unidas con una línea punteada representan la media de nodos visitados. La línea continua representa a un polinomio de segundo grado, lo que nos indica que el número medio de nodos visitados se aproxima a una parábola. De la gráfica también se concluye que a mayor número de objetos mayor dispersión en el número de nodos visitados. Este comportamiento creemos que está ligado al generador de problemas aleatorio que se está utilizando.

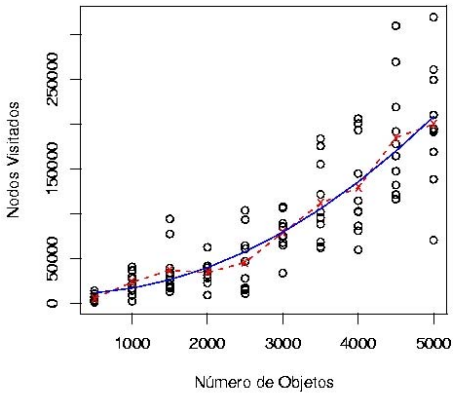


Figura 2. Resultados del Secuencial

Un parámetro muy interesante de estudiar en las implementaciones paralelas de los algoritmos de ramificación y acotación es el "equilibrado de la carga de trabajo" entre los distintos procesadores que intervienen en la ejecución del algoritmo.

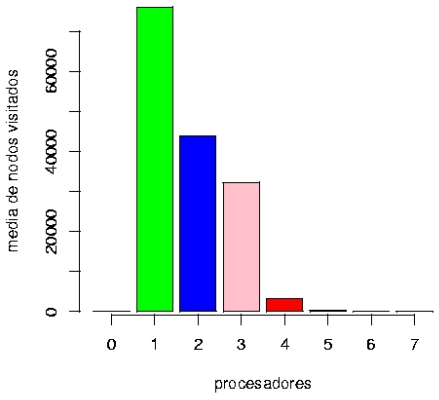


Figura 3. Resultados ejecución paralela

La Figura 3 muestra la distribución media de nodos visitados entre ocho procesadores de cinco ejecuciones de un problema de la mochila generado aletoriamente de tamaño 4000. Se trata de un conjunto de máquinas heterogéneo. Las dos primeras son AMD-Duron a 800 MHz y el resto a 500 MHz, todas con 256 Mb de memoria. Nótese que el procesador cero no visita ningún nodo puesto que se trata del *Maestro*. Creemos que el primero de los esclavos explora la mayor parte del espacio de búsqueda, porque se trata de un procesador más rápido y debido al tamaño del problema, no queda mucho trabajo para los últimos. También hemos notado una gran diferencia entre el número de nodos visitados de una ejecución a otra. La Figura 4 muestra los resultados de los cinco experimentos que dan lugar a la media de la Figura 3. Creemos que el incremento de nodos en la segunda ejecución se debe a que existía algún otro proceso en la máquina uno que consumía muchos recursos. Esto implicaría que la recepción de las cotas que permitirían podar se realiza más tarde, por lo que se explora un espacio de búsqueda mayor.

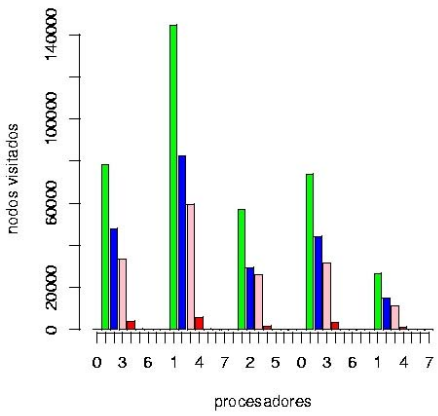


Figura 4. Cinco Ejecuciones Paralelas para N=4000

4. Conclusiones

Se ha presentado a través de un ejemplo el modo de uso de las herramientas CALL y LLAC para el análisis de complejidad de algoritmos.

El ejemplo utilizado forma parte de las prácticas de laboratorio de las asignaturas *Programación en Paralelo I y II* que se imparten en la Ingeniería Superior en Informática en la Universidad de La Laguna. Sin embargo, consideramos que estas herramientas serían de gran utilidad en la docencia de asignaturas que incluyan en sus contenidos la descripción de técnicas algorítmicas. Las herramientas proporcionan la posibilidad no sólo de estudiar implementaciones secuenciales de algoritmos, sino también paralelas.

Además, el algoritmo paralelo propuesto para resolver el problema de la Mochila 0/1, se puede generalizar para resolver mediante Ramificación y Acotación otro tipo de problemas.

Agradecimientos

Este trabajo ha sido financiado parcialmente por los proyectos del ministerio de Ciencia y Tecnología: TIC2002-04498-C05-05 (TRACER) y TIC2002-04400-C03-03 (PELICAN).

Referencias

[1] Dorta I., León C., Rodríguez C., Rojas A. *Parallel Skeletons for Divide-and-Conquer and Branch-and-Bound techniques*. In Proceeding of 11th Euromicro Conference on Parallel, Distributed and Network based Processing, 2003.

[2] Ihaka R., Gentleman R. *R, A language for Data Analysis and Graphics.* Journal of Computational and Graphical Statistics, 5 (3), pp. 299-314, 1996.

[3] Martello S., Toth P. *Knapsack Problems: Algorithms and Computer Implementations*. Jonh Wiley & Sons Ltd, 1990.

[4] Planes de Estudio del CSI. http://www.csi.ull.es

[5] Rodríguez, G. *CALL: a complexity Analysis Tool*. Proyecto Fin de Carrera, Centro Superior de Informática, Universidad de La Laguna, Junio 2002. http://nereida.deioc.ull.es/~call.

[6] Snir M., Otto S., Huss S., Walker D. and Dongarra J. *MPI-the Complete Reference*. 2nd Edition, MIT Press, 1998.

Sistemas operativos

La programación concurrente y el interbloqueo en la asignatura de Sistemas Operativos

Miguel Riesco Albizu, Marián Díaz Fondón
Depto. de Informática
Universidad de Oviedo
e-mail: albizu@lsi.uniovi.es
fondon@correo.uniovi.es

Resumen

La programación concurrente es un tópico dentro de la docencia de los Sistemas Operativos. En este artículo se revisa su papel dentro de la asignatura y su relación, excesivamente estrecha en muchos casos, con otro tema de la misma asignatura como es el del interbloqueo.

1. Introducción

Dentro de los temas típicos que se imparten dentro de la materia de sistemas operativos está el de la programación concurrente. Este tema aparece incluido en los modelos curriculares más utilizados [9, 10], dentro del cuerpo de conocimiento de *Sistemas Operativos* (en concreto en el OS2) en el primer caso y en el módulo 4.3 (*Sistemas operativos y Arquitectura de Computadores I)* en el segundo. Los libros de texto que existen sobre la materia también recogen este tema, con mayor o menor profusión.

Lo mismo ocurre, como no podía ser menos, con la inmensa mayoría de los programas de las asignaturas de Sistemas Operativos que se imparten en las distintas Universidades.

Ante semejante unanimidad, parece lógico considerar que la inclusión de esta materia en el programa de una asignatura de este estilo está fuera de toda duda o discusión. En este artículo comenzaremos discrepando de esta opinión, por muy generalizada que esté, intentando aportar las razones que nos llevan a disentir.

Por otra parte, otro tema que también se incluye en todos los libros y temarios de Sistemas Operativos es el del interbloqueo. Sin discutir en este caso su adecuación con la materia, sí discrepamos de su posición temporal en el desarrollo de la docencia de la asignatura, generalmente muy ligado al tema de programación concurrente, tanto que muchas veces aparece incluso incluido en él.

2. Programación concurrente y Sistemas Operativos

El tema de programación concurrente, como su propio nombre indica, es un tema eminentemente de programación. Trata de explicar las peculiaridades que presenta la realización de programas concurrentes, los problemas más habituales que aparecen y los medios más utilizados para resolverlos.

Ante este contenido puede surgir la pregunta de por qué se incluye este tema dentro de la asignatura de Sistemas Operativos, dado que, aparentemente, no tiene ninguna relación directa con los contenidos esperados de esta asignatura. A esa pregunta se suelen dar varias respuestas:

1. Con la inclusión de la multiprogramación, el sistema operativo debe manejar procesos concurrentes, con lo que un estudiante de sistemas operativos debe conocer los problemas que pueden aparecen en estos casos, así como la manera de solucionarlos.
2. El sistema operativo suele incluir mecanismos de sincronización y de comunicación entre procesos, por lo que se debe conocer para qué y cómo se van a utilizar estos mecanismos.
3. El propio sistema operativo se implementa muchas veces como un conjunto de funciones concurrentes, con lo que se debe ser capaz de entender, y en su caso desarrollar, programas concurrentes correctos.

Todas estas razones hacen referencia al carácter instrumental de esta materia. Es decir, es necesa-

rio conocer programación concurrente para estudiar y comprender la materia específica de Sistemas Operativos, que no es otra que estudiar la estructura interna y el funcionamiento de un sistema operativo. Podemos resumir, por tanto, esas razones para la inclusión de programación concurrente en una sola: "Para la docencia de Sistemas Operativos nos hace falta la programación concurrente; como no se imparte en otro sitio, hay que hacerlo aquí".

Esta razón podría considerarse académicamente válida si este tema fuera única y exclusivamente de utilidad dentro de la asignatura de Sistemas Operativos. Sin embargo, hay otras asignaturas, donde también son necesarios en mayor o menor grado los conceptos básicos de programación concurrente. Centrándonos en el primer modelo de currículo de los antes citados [9] son necesarios conceptos de programación concurrente en los cuerpos de conocimiento de *Programming Fundamentals* (PF7, en concreto), *Programming Languages* (PL11), *Information Management* (IM5 e IM6) y *Net-Centric Computing* (NC7), además de en el *Operating Systems* (OS2) ya citado.

Por otro lado, además de en las asignaturas que desarrollan estos cuerpos de conocimiento, en muchos Centros se considera la materia como digna de figurar como asignatura independiente en sus planes de estudio, tal y como recomienda el modelo curricular UNESCO-IFIP[10].

Volviendo al razonamiento de "inclusión por necesidad", si lo aplicamos a otras partes de la asignatura podíamos incluir también un tema sobre programación, dado que es necesaria para entender los algoritmos que se explican, otro sobre estructuras de datos, para poder comprender las estructuras de datos, muchas veces complejas, que usan los sistemas operativos. O incluso podríamos, como se hace en [5] introducir un tema sobre modelado analítico y teoría de colas, que puede ser muy interesante para estudiar y comparar el comportamiento de las políticas que se emplean en distintas partes de un sistema operativo. Como obviamente esto no parece muy adecuado, podemos concluir que el tema de "programación concurrente" no debería estar incluido en la asignatura de Sistemas Operativos.

3. ¿Dónde debería tratarse la programación concurrente?

Centrándonos en las directrices del Consejo de Universidades para las Ingenierías Técnicas en Informática [3, 4] (y podemos extenderlo a las Ingenierías Superiores [2]), podremos ver que el tema de "programación concurrente" no aparece en el descriptor de ninguna asignatura, con lo que podrá impartirse legalmente en cualquiera donde tenga sentido incluirse.

Lo que se pretende con este artículo es iniciar un debate para decidir dónde podría ser más adecuado impartir una introducción a los conceptos básicos de programación concurrente.

Nosotros proponemos que sea en alguna de las asignaturas en la que se estructure la materia de "Metodología y Tecnología de la Programación", por tres motivos:

1. Es la materia donde más conceptos relacionados con programación concurrente podemos incluir de los expuestos en [9].
2. Tiene casi el triple de carga docente asignada en las directrices del Consejo de Universidades [2, 3, 4] que la de Sistemas Operativos.
3. Independientemente de currículos y de directrices, el tema de programación concurrente es un tema (como su propio nombre indica) de programación.

4. Otro problema: el interbloqueo

Otro tema en principio relacionado con la programación concurrente es el del interbloqueo. En este apartado estudiaremos esta relación, así como el lugar más adecuado para la manera de abordar su docencia en la asignatura de sistemas operativos.

El interbloqueo es una anomalía que puede ocurrir durante la ejecución de procesos concurrentes debido a la competencia por los recursos. Si bien es cierto que prácticamente ningún sistema operativo real incorpora mecanismos de tratamiento de interbloqueo, esto es debido a una cuestión de la pérdida de rendimiento que conlleva su tratamiento para la baja probabilidad que hay de que ocurra. En un sistema operativo ideal, sin embargo, sí deberían incluirse mecanismos para su tratamiento, dado que existen y son bien conocidos.

Podemos considerar, sin temor a equivocarnos, que el tema del interbloqueo, además de ser un clásico en todos los libros, modelos de currículos, etc., sobre sistemas operativos es un tema que por sus características merece estar incluido en el temario de esta asignatura.

Si bien no se discute, por tanto, su adecuación a la asignatura, sí nos vamos a permitir cuestionar su posición en el desarrollo de la materia. La práctica totalidad de los libros de texto [5, 6, 7, 8], los distintos modelos de currículos [9, 10] y los programas de las distintas asignaturas que sobre el tema se imparten colocan este tema inmediatamente después del correspondiente a programación concurrente. Más aún, en algunas ocasiones [9] se llega a incluir el interbloqueo dentro del tema de programación concurrente.

Este hecho nos parece un importante error, por los motivos que expondremos a continuación, que hace por un lado que no se comprenda del todo bien su objetivo, y que, por otra parte, se complique innecesariamente un tema tan árido para los alumnos como es la programación concurrente.

5. La programación concurrente y el interbloqueo

Uno de los problemas con los que se puede encontrar el alumno cuando empieza a realizar sus primeros programas concurrentes es el del interbloqueo. En este contexto el interbloqueo que se produce es debido a una mala programación. Por ejemplo, en los clásicos intentos de solución al problema de la exclusión mutua se muestra este problema debido a un mal planteamiento del problema.

La única solución que hay para solventar los interbloqueos de este tipo es eliminar el error en la lógica del programa para que los procesos implicados no estén esperando el uno por el otro.

La utilización de mecanismos de sincronización de alto nivel facilita la realización de programas concurrentes libres de errores, de la misma manera que el desarrollo de un programa en un lenguaje de alto nivel es más fácil e introduce menos errores que si lo realizamos en ensamblador. Pero tanto en un caso como en otro sólo una buena programación puede asegurarnos tener éxito en nuestro propósito de construir programas correctos.

Lo que está claro es que, en ninguna circunstancia, podremos utilizar ninguna técnica que nos asegure la realización de un programa concurrente libre de interbloqueo ante un problema dado. Igualmente, tampoco podremos desde el sistema operativo evitar que ese interbloqueo se produzca si los procesos se empeñan en caer en él.

6. Interbloqueo y gestión de recursos

Otro de los campos típicos donde se presenta el problema del interbloqueo es en la gestión de recursos. En este caso varios de los procesos que se ejecutan concurrentemente en el sistema (procesos probablemente independientes entre sí) compiten por el uso de recursos no compartibles, de tal manera que hasta que el proceso que tiene asignado el recurso no lo libere ningún otro podrá utilizarlo. Si otro proceso necesita utilizarlo para continuar su trabajo, lo típico que ocurre es que este proceso suspende su ejecución hasta que el que ocupa el recurso lo libera.

En este contexto el interbloqueo se produce cuando los procesos suspendidos retienen recursos que son necesarios para la continuación de otros procesos, que a su vez retienen los recursos que necesitan los primeros. En definitiva, el problema se debe a que se han asignado los recursos que han ido necesitando los procesos sin ningún criterio, sumado al propio comportamiento de los procesos.

Este tipo de interbloqueo no se da únicamente en los sistemas operativos. Es muy típico también en los sistemas de gestión de bases de datos, por ejemplo, donde los recursos son registros o tablas de una base de datos que los procesos bloquean mientras los manejan si precisan de un acceso exclusivo.

Esta clase de interbloqueo está perfectamente estudiado y puede tratarse sin demasiada dificultad utilizando distintas técnicas, clasificadas normalmente en tres categorías: *prevención*, *evitación* y *detección y recuperación*. El problema que tienen estas técnicas es que sacrifican algo a cambio de solucionar el interbloqueo (la asignación de recursos será más lenta, al tener que comprobar si se puede permitir; se exige seguir una serie de normas a los procesos a la hora de solicitar recursos; etc.)

Este es el motivo de que no se suelan apenas utilizar en sistemas reales. Pero en cualquier caso

es un problema muy interesante de gestión de recursos que debe explicarse dentro del tema de Sistemas Operativos[1].

7. Relación entre el interbloqueo, la programación concurrente y los Sistemas Operativos

Como ya hemos indicado anteriormente, en la mayoría de libros, currículos y temarios de asignaturas de Sistemas Operativos el tema del interbloqueo está muy ligado al de programación concurrente. Tanto es así que habitualmente el tema de interbloqueo sigue inmediatamente al de programación concurrente, o incluso en algunos casos [9] aparece incluido en él como un apartado más. A nuestro juicio esto es un grave error, dado que hace parecer que son temas íntimamente relacionados cuando en realidad no es así. El interbloqueo que aparece cuando tratamos con programación concurrente es radicalmente distinto en su origen del que se presenta debido a la gestión de recursos:

- Los procesos implicados en un interbloqueo del primer tipo son procesos cooperantes, que comparten información y se sincronizan para lograr un objetivo común. En el segundo caso los procesos son independientes, y sólo comparten la necesidad de utilizar recursos comunes.
- El primer tipo de interbloqueo es causado por una mala sincronización, debido a una mala programación, al no considerar la situación que lleva al interbloqueo. En el segundo caso el problema aparece por una falta de control en la asignación de recursos que solicitan los procesos por parte del sistema operativo.
- El segundo tipo de interbloqueo sí es un problema que debe resolver el sistema operativo, mientras que el primer tipo es un problema meramente de programación.

De la misma manera, la solución al problema también es completamente distinta:

- El primer tipo de interbloqueo se soluciona mediante la corrección de los programas implicados en los mismos.
- El segundo tipo de interbloqueo se soluciona realizando la asignación de recursos a los procesos según alguna de las técnicas diseñadas al efecto (previniendo o evitando el interbloqueo, fundamentalmente).

En resumen, el segundo tipo de interbloqueo sí entraría dentro de la materia de sistemas operativos, mientras que el primero debería resolverse dentro del campo de la programación concurrente.

El modelo curricular de la UNESCO [10] sí parece recoger esta sensación, dado que incluye el interbloqueo en dos temas distintos: en el tema de procesos (donde se incluye también el de programación concurrente) en el módulo de *Sistemas Operativos y Arquitectura de Computadores I*, así como en el tema de entrada/salida del módulo de *Sistemas Operativos y Arquitectura de Computadores II*.

8. ¿Y cuál es el problema?

El contenido del tema del interbloqueo que se imparte en la asignatura de Sistemas Operativos se centra única y exclusivamente en el segundo tipo de interbloqueo, como no podía ser de otro modo.

Parece, por tanto, no demasiado adecuada la situación del tema de tratamiento del interbloqueo en el temario de la asignatura, dado que al situarlo a continuación del tema de programación concurrente puede dar la impresión, errónea, de que en ese apartado vamos a ver métodos para resolver los problemas de interbloqueo que nos han ido surgiendo en nuestros programas concurrentes, como en la "Cena de los filósofos" o en el "Problema de la sección crítica". Esto hace que el alumno no comprenda bien el objetivo del tema y su utilidad dentro de un sistema operativo, creyendo en un principio que los métodos de tratamiento que se exponen van a ser la panacea a los problemas que se le presentan en su incipiente vida de desarrollador de programas concurrentes.

El porqué se sitúa en este lugar el tema del interbloqueo no puede explicarse más que por inercia (porque a alguien se le ocurrió en su momento a hacerlo así y tuvo éxito), dado que, como hemos visto, a pesar de haber "interbloqueos" en programación concurrente éstos nada tienen que ver con los que se tratan en este tema.

[1] No hay que olvidar que una definición clásica de "sistema operativo" lo caracteriza como "gestor de recursos"

9. ¿Dónde debería situarse el tema del interbloqueo?

Dentro de los sistemas operativos los posibles problemas de interbloqueo se suelen dar por la utilización de recursos periféricos. Los distintos modelos curriculares suelen incluir un tema de "Gestión de dispositivos" (o "Gestión de entrada/salida"), donde se explica la forma que tiene el sistema operativo de manejar los distintos dispositivos periféricos.

Nuestra propuesta es, por tanto, situar el tema del interbloqueo tras el tema de "Gestión de dispositivos", para entender cómo se gestionan los dispositivos periféricos antes de ver cómo se trata el problema del interbloqueo debido a su compartición.

Con esta medida conseguimos, asimismo, alejarlo del tema de programación concurrente, lo cual nos aporta dos ventajas adicionales:

- Aumentamos el tiempo para que los alumnos asimilen los conceptos de programación concurrente, completamente nuevos para ellos, con lo que podrán comprender mejor las diferencias entre los dos tipos de interbloqueo.
- Evitamos confusiones innecesarias al enfatizar la diferencia entre los dos tipos de interbloqueo.

El modelo curricular UNESCO/IFIP [10] apoya, además, esta decisión.

10. Conclusiones

A pesar de que en buena parte de la bibliografía consultada se considera el tema de programación concurrente como una parte importante de la docencia de Sistemas Operativos, consideramos que no debe ser así. Sólo el hecho de que no se quiera incluir en ninguna otra asignatura, donde probablemente sería más adecuado hacerlo, puede justificar que se imparta en ésta, ante la necesidad que tiene de conocer esa materia para comprender los conceptos que le son propios.

En cualquier caso, si se incluye la programación concurrente dentro del temario de la asignatura de Sistemas Operativos consideramos que debe separarse claramente del tema de "tratamiento del interbloqueo", dado que, por una parte, no tiene prácticamente ninguna relación con él, y que, por otro lado, induce a los alumnos a confundir las dos clases de interbloqueo que se ven en la asignatura, complicando innecesariamente, además, la comprensión del tema de programación concurrente, bastante complejo de por sí.

Referencias

[1] Carretero Pérez, J. y otros. *Sistemas Operativos. Una visión aplicada*. McGraw Hill, 2001

[2] Consejo de Universidades. *Directrices generales propias de los planes de estudios conducentes a la obtención del título oficial de Ingeniero en Informática*. Boletín Oficial del Estado, número 278, pag. 34401 a 34403

[3] Consejo de Universidades. *Directrices generales propias de los planes de estudios conducentes a la obtención del título oficial de Ingeniero Técnico en Informática de Gestión*. Boletín Oficial del Estado, número 278, pag. 34403 y 34404

[4] Consejo de Universidades. *Directrices generales propias de los planes de estudios conducentes a la obtención del título oficial de Ingeniero Técnico en Informática de Sistemas*. Boletín Oficial del Estado, número 278, pag. 34404 y 34405

[5] Deitel, H. M. *Sistemas Operativos. Segunda Edición.* Addison-Wesley Iberoamericana, 1993

[6] Sinlverschatz, A. y Galván, P.B. *Sistemas Operativos.* Quinta Edición. Addison Wesley Longman, 1999.

[7] Stallings, W. *Sistemas Operativos.* Cuarta Edición. Prentice Hall, 2001.

[8] Tanenbaum, A. y Woodhull A. *Operating Systems Design and Implementation*, 2ª edición. Prentice-Hall, 1997

[9] The Joint Task Force on Computing Curricula, *Computing Curricula 2001* IEEE Computer Society, Association for Computing Machinery, 2000

[10] UNESCO/IFIP, *A Modular Curriculum in Computer Science*. UNESCO, 1994.

Simulador didáctico de gestión de memoria con interfaz de autoaprendizaje basado en WWW

Félix Buendía, Julio Sahuquillo, Juan-Carlos Cano
Departamento de Informática de Sistemas y Computadores
Escuela Técnica Superior de Informática Aplicada
Universidad Politécnica de Valencia
e-mail: {fbuendia, jsahuqui, jucano}@disca.upv.es

Resumen

La enseñanza de las materias relacionadas con los Sistemas Operativos cubre un conjunto de contenidos esenciales en cualquier currículo universitario de informática. Las diferentes asignaturas impartidas, incluyen temas tales como la gestión de procesos y gestión de memoria, con una fuerte componente teórica. Aunque dichos conceptos son ampliamente tratados en numerosos libros de texto, no resulta sencillo encontrar entornos adecuados al nivel de conocimientos del alumno, que permitan realizar actividades donde poner en práctica los conceptos introducidos de forma teórica. En este trabajo, se describe una herramienta de simulación realizada de forma "*ad hoc"* para mostrar el funcionamiento de los principales aspectos relacionados con la gestión de memoria en un sistema operativo. Las características de dicha herramienta permiten que el alumno pueda interactuar con el simulador, introduciendo ejemplos de carga y configurando parámetros del sistema tales como el tamaño de la memoria y el algoritmo de gestión de la misma. La herramienta esta desarrollada con tecnología de desarrollo basada en WWW lo que facilita su amplia difusión y utilización.

1. Introducción

Las asignaturas relacionadas con los Sistemas Operativos son un elemento fundamental en cualquier currículo universitario de informática. Desde las propuestas de ACM/IEEE de 1991 [1] hasta las más recientes del 2001 [2], existe un consenso sobre la importancia de esta temática. Las directrices del MEC establecen una serie de descriptores que hacen referencia a aspectos como la gestión de procesos y ficheros así como los recursos de memoria y entrada/salida. Se trata de aspectos con una fuerte componente teórica pero que deben encontrarse apoyados con actividades prácticas. Sin embargo, no resulta sencillo encontrar entornos donde realizar este tipo de actividades, que permitan al alumno fijar con la práctica los conceptos tratados de forma teórica. Tradicionalmente, se han realizado actividades prácticas directamente sobre los sistemas operativos reales, donde el acceso a los mecanismos de gestión de recursos es complejo.

En algunos casos, se dispone de sistemas operativos reducidos como Nachos [4] que aunque presentan una amplia funcionalidad, para asignaturas troncales básicas tienen como principal inconveniente su orientación hacia aspectos de diseño e implementación.

Otras herramientas como RCOS [3] simulan el funcionamiento de un sistema operativo mediante animaciones. Sin embargo, la propuesta de RCOS se centra demasiado en los aspectos visuales y no profundiza en la aplicación de aspectos teóricos. En [5], Magee y Kramer presentan una herramienta que, de manera similar a la que se presenta en esta ponencia, se centra en la demostración visual de aspectos concernientes a aspectos relacionados con los conceptos de concurrencia.

En este trabajo, se presenta un simulador utilizado para mostrar el funcionamiento de la gestión de memoria de un sistema operativo. El simulador incorpora un entorno gráfico que visualiza la actividad del gestor de memoria. Este entorno junto con un menú detallado de ayuda facilita, sin duda, el autoaprendizaje de la materia al alumno.

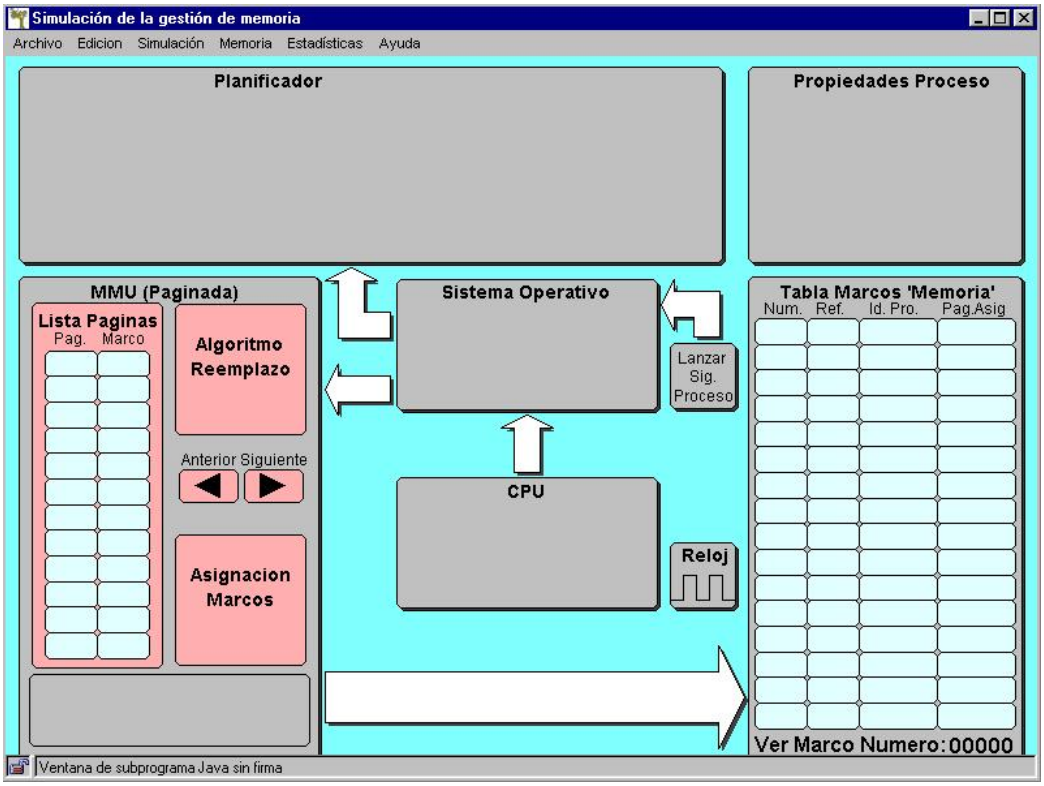


Figura 1. Esquema del simulador de gestión de memoria.

De hecho, son muchos los que durante el presente curso académico lo han utilizado para constatar los resultados de ejercicios propuestos en clase. Para facilitar esta tarea de aprendizaje, el simulador se encuentra disponible en [8].

El resto del trabajo se organiza como sigue. La Sección 2 describe la herramienta de simulación, mostrando su funcionamiento general. La Sección 3 detalla algunos ejemplos de ejercicios basados en la utilización de la herramienta. Finalmente, la Sección 4 presenta las conclusiones del trabajo.

2. Simulador de gestión de memoria

La herramienta de simulación ha sido desarrollada en lenguaje Java. Una primera versión de la misma dio lugar a un proyecto final de carrera [6], dirigido por el primer autor del artículo y enmarcado dentro de la Escuela Técnica Superior de Informática Aplicada de la Universidad Politécnica de Valencia. Nuevas versiones con interfaz mejorada se han ido incorporando como parte del trabajo realizado en el ámbito de un proyecto europeo de educación a distancia [7]. La Figura 1 muestra la pantalla principal de la última versión de la herramienta, que es la que se utiliza para realizar prácticas en las asignaturas de Sistemas Operativos.

2.1. Introducción al simulador

El Simulador de gestión de memoria tiene por objeto mostrar el funcionamiento de algunos aspectos de la gestión de memoria que intervienen en la ejecución de uno o más procesos en un entorno de multiprogramación. Concretamente, la herramienta describe la ejecución de un conjunto

de procesos mediante el algoritmo de planificación *round-robin* y cómo estos procesos generan durante su ejecución accesos a direcciones lógicas qué serán traducidas a direcciones en memoria física. El esquema de traducción de direcciones se basa en una técnica de paginación combinada con algoritmos de reemplazo, lo que se conoce globalmente como memoria virtual mediante paginación bajo demanda.

2.2. Inicio de una sesión

El primer paso en la ejecución de una simulación de gestión de memoria consiste en cargar el fichero de simulación (menú Archivo). Para describir un determinado programa, se ofrece un fichero denominado "Simulacion.inf" que describe las características de los diferentes procesos. La Figura 2 muestra un ejemplo.

```
Proceso1 50000 0
5000
0
100
0
456
4000
1050
end
0
Proceso2 50000 0
2048
7999
2248
3100
230
end
0
Proceso3 60000 0
2148
50000
3477
6000
2340
end EOF
```

Figura 2. Ejemplo de carga.

Este fichero contiene una secuencia de direcciones lógicas agrupadas por cada proceso que se incluye en el sistema. Para ello se utiliza una cabecera con tres campos: el primero hace referencia al identificador del proceso, el segundo el tamaño asignado a dicho proceso y el tercero a la prioridad, que identificará la cola a la que pertenece el proceso. Por ejemplo, la primera línea del fichero contiene la siguiente declaración:

```
Proceso1 50000 0
```

que define el Proceso1, con un tamaño de 50000 bytes y una prioridad 0. A partir de esta línea aparecen valores numéricos separados en diferentes líneas, cada uno de los cuales representa una dirección lógica. Las direcciones lógicas contenidas en un mismo proceso finalizan cuando aparece la etiqueta "end". A continuación, y antes de empezar la especificación de un nuevo proceso, aparece un valor numérico que especifica el desplazamiento temporal en la activación del siguiente proceso respecto al anterior. En este caso, se utiliza un valor 0 que indica que el Proceso1 y el siguiente (Proceso2) se activan al mismo tiempo. Un valor de 2 significaría que deben transcurrir dos instantes de tiempo antes que se active el Proceso2. La aparición de una etiqueta "EOF" a continuación de la última marca "end" indica la finalización del fichero de simulación.

Dentro de la opción "Ver Ventana de Direcciones Lógicas" (menú Edición) se puede editar el contenido del fichero de simulación, para modificar, por ejemplo, alguna de las direcciones lógicas o el tamaño de un proceso.

2.3. Configuración del sistema

Una vez que se ha cargado el fichero de simulación, se puede proceder a la configuración del sistema mediante una opción con dicho nombre del menú Archivo. La Figura 3 muestra un ejemplo. Se puede observar, como esta opción permite asignar diferentes atributos del sistema tales como el tamaño de la memoria, el tamaño de la página, los algoritmos de asignación y reemplazo del sistema.

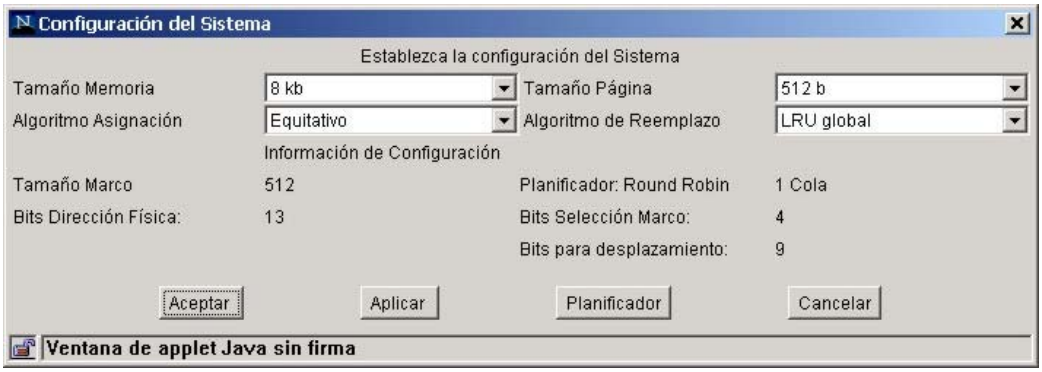


Figura 3. Ventana de configuración del sistema.

- Tamaño de memoria: indica el número de bytes disponibles en la memoria principal. El rango permitido oscila desde 8KBytes hasta 1Mbyte.
- Tamaño de página: indica la capacidad en bytes de cada una de las páginas en que se divide un proceso. Puede seleccionarse entre un rango de valores que van desde 512 bytes a 4 KBytes.
- Algoritmo de reemplazo: determina el método que se utiliza para seleccionar la página víctima de un proceso cuando se produce un fallo de página y no existe un marco de página disponible en memoria. Los posibles métodos a elegir son: LRU: local y global y FIFO: local y global.

En caso de utilizar un algoritmo de reemplazo local, habrá que elegir un determinado algoritmo de asignación, el cual determina el método utilizado para repartir los marcos de página disponibles en memoria entre los procesos del sistema. Existen tres métodos posibles:

- Equitativo.
- Proporcional.
- Prioritario.

La selección de atributos del sistema permite determinar aspectos tales como el tamaño del marco de página, el número de bits de la dirección física y cómo estos se distribuyen (los bits de mayor peso se destinan a la selección del marco y los de menor peso al desplazamiento dentro del marco).

Por otra parte, tal y como se muestra en la Figura 3, mediante la opción "Planificador" también pueden configurarse ciertos parámetros referidos a la Planificación del Sistema. Los parámetros que se pueden configurar son los siguientes:

- Algoritmo de planificación: actualmente, sólo se encuentra implementado un algoritmo de planificación del tipo "round-robin" (turno rotatorio). Cabe matizar, que existen 2 proyectos fin de carrera en fase de escritura destinados a implementar otros algoritmos.
- Número de colas: indica el número de estructuras de datos para almacenar procesos activos. Se pueden elegir de 1 a 5 colas mediante los identificadores cola 0 a cola 4, donde la cola 0 es la más prioritaria.
- Quantum: permite definir para cada cola el "quantum" o cantidad de tiempo que se asocia a un proceso en su turno de ejecución. En este caso, dicho tiempo se evalúa en base al número de direcciones lógicas. Por defecto su valor es 1, es decir, se accede a una dirección lógica en cada instante de tiempo.

2.4. Comienzo de la simulación

Una vez cargado el fichero de simulación y realizada la configuración del sistema, para simular el comportamiento se elige la opción "Comenzar Simulación" dentro del menú de Archivo.

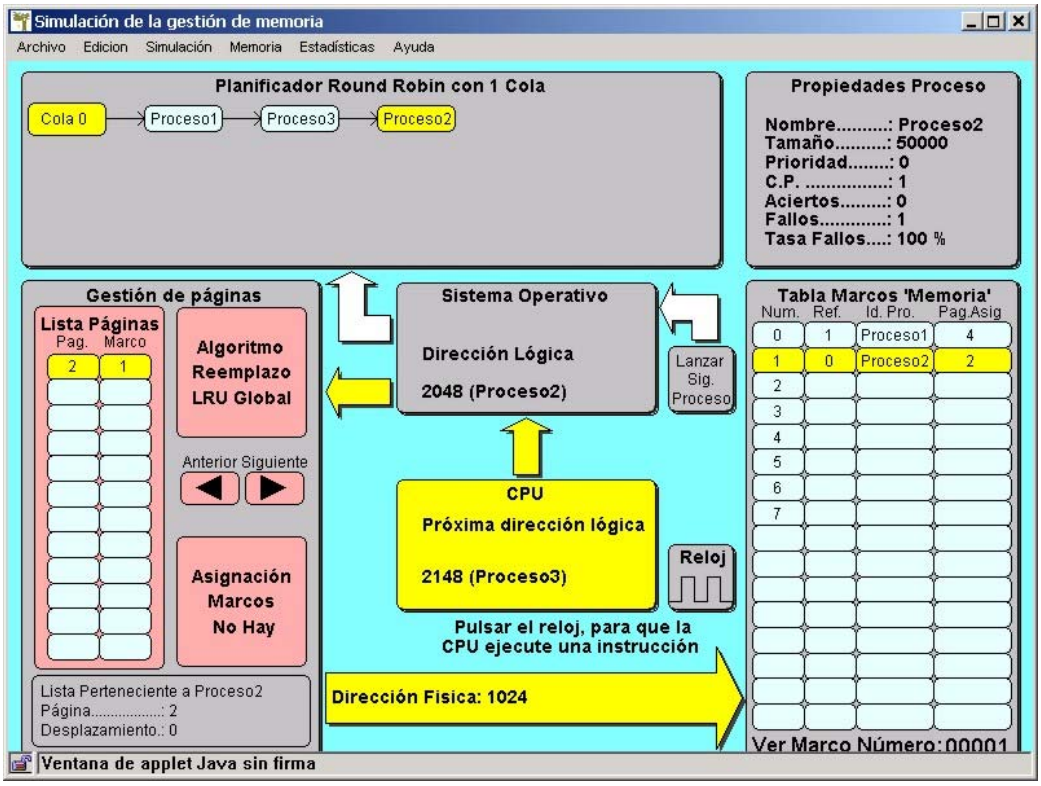


Figura 4. Ejemplo de ejecución del simulador.

Una vez iniciada la simulación, se puede observar la evolución de la misma a través de diferentes zonas de la pantalla mostradas en la Figura 4. Existen dos formas de controlar la simulación, las cuales se pueden seleccionar desde el menú de Simulación:

- Modo paso a paso: permite visualizar los resultados de cada paso (unidad temporal) en que se realiza cada acceso a memoria o la activación de un nuevo proceso. Para ello, el usuario debe pulsar el icono de Reloj que aparece en la parte central de la pantalla.
- Modo automático: en este caso la ejecución de los procesos y por tanto, el acceso a direcciones de memoria se realiza de forma automática sin intervención del usuario.

La parte superior izquierda de la pantalla muestra una representación gráfica de los procesos activos en cada cola. En el ejemplo se utiliza una única cola (Cola 0) que contiene los tres procesos definidos. Además, cada vez que se accede a la dirección lógica de un proceso, éste se pondrá al inicio de la cola y cambiará el color del símbolo que lo representa, para indicar que es el proceso en ejecución.

La parte superior derecha de la pantalla informa del estado del proceso en ejecución. Para ello, se incluye el contador de direcciones lógicas (C.P.) que define el orden relativo de la dirección lógica accedida por el proceso en ese momento y que se actualiza con cada nuevo acceso. Además, se indica el número de accesos realizados hasta el momento actual, diferenciando entre fallos de página (la página que contiene la dirección actual no está en memoria principal y tiene que ser transferida desde disco) y aciertos (la página accedida está ya en memoria). También se incluye el valor de la tasa de fallos.

La parte inferior izquierda representa la gestión de páginas e incluye información como:

- El algoritmo de reemplazo seleccionado.
- El tipo de asignación de marcos si lo hubiera.
- La lista de páginas utilizada por el correspondiente algoritmo de reemplazo para elegir la página víctima y decidir el marco que va a ser liberado. Esta estructura incluye unos iconos ("Anterior" y "Posterior") para poder manejar los elementos de la lista que excedan el tamaño dispuesto en la pantalla (12 páginas). También se incluye información de la dirección lógica actual, como la página y el desplazamiento dentro de ella. Esta lista no representa la tabla de páginas ya que sólo almacena parte de su información (en este caso el número de marco asociado a una página).

La parte inferior derecha de la pantalla representa la Tabla de Marcos de la memoria física. En ella se muestra la siguiente información:

- El número de marco: su valor va desde 0 hasta N, siendo N+1 el número de marcos disponible. En caso de sobrepasar el número máximo de elementos de la tabla en pantalla (16), se podrán utilizar una serie de controles que se activan al colocar el cursor en alguno de los números de la opción "Ver Marco Número". Cuando se activan estos controles aparece el símbolo de una mano y al pulsar el botón izquierdo del ratón el número se incrementa, mientras que al pulsar el botón derecho, se decrementa. Ello permite el acceso a la totalidad de la estructura de la tabla de marcos de página.
- El campo referencia define la antigüedad de las páginas que se almacenan en memoria y sirve para decidir el marco ocupado por la página a reemplazar, en caso de fallo. Por tanto, depende del tipo de algoritmo de reemplazo elegido.
- El identificador de proceso: indica el proceso que accede a la página.
- Página asignada: se trata de la página almacenada en el marco referido.

Por último, la parte central de la pantalla define la relación entre las estructuras descritas previamente. Indica en cada momento, que proceso ha sido elegido por parte del planificador para su ejecución y la dirección lógica implicada en este paso. Dicha dirección puede ser seleccionada de forma automática (modo "Automático") o interactiva a través del icono Reloj (modo "Paso a paso"). Asimismo, se puede activar un nuevo proceso mediante el control "Lanzar Siguiente Proceso", sin que se tenga que esperar al plazo de tiempo definido en el fichero de carga.

También se encuentran en esta parte de la pantalla tres etiquetas que contienen la siguiente información:

- Sistema operativo: indica la dirección lógica que actualmente está siendo traducida.
- CPU: representa la próxima dirección a traducir.
- Dirección física: contiene la dirección física resultado de la traducción actual.

2.5. Resultados de la simulación

Mediante la opción "Ver Ventana de Direcciones Físicas" del menú Edición, se obtiene una traza de la simulación. Esta traza muestra un listado de direcciones físicas como consecuencia del proceso de traducción. La Figura 5 muestra un ejemplo de dicha traza.

```
**************************************
Proceso1  Dir.Lógica: 5000     Dir. Física: 904
Proceso2  Dir. Lógica: 2048     Dir. Física: 1024
Proceso3  Dir. Lógica: 2148     Dir. Física: 2148
Proceso1  Dir. Lógica: 0     Dir. Física: 3072
Proceso2  Dir. Lógica: 7999     Dir. Física: 4927
Proceso3  Dir. Lógica: 50000     Dir. Física: 5968
Proceso1  Dir. Lógica: 100     Dir. Física: 3172
Proceso2  Dir. Lógica: 2248     Dir. Física: 1224
Proceso3  Dir. Lógica: 1024     Dir. Física: 6144
Proceso1  Dir. Lógica: 0     Dir. Física: 3072
Proceso2  Dir. Lógica: 4096     Dir. Física: 7168
Proceso3  Dir. Lógica: 1024     Dir. Física: 6144
Proceso1  Dir. Lógica: 456     Dir. Física: 3528
Proceso2  Dir. Lógica: 2134     Dir. Física: 1110
Proceso3  Dir. Lógica: 3477     Dir. Física: 405
Proceso1  Dir. Lógica: 4000     Dir. Física: 2976
Proceso2  Dir. Lógica: 3100     Dir. Física: 4124
Proceso3  Dir. Lógica: 6000     Dir. Física: 6000
Proceso1  Dir. Lógica: 1050     Dir. Física: 7194
**********************************
```

Figura 5. Ventana de direcciones físicas

Tras finalizar la simulación y descargar los procesos de la memoria, se pueden obtener estadísticas de la ejecución de la simulación. La Figura 6 muestra un ejemplo de dichos resultados, los cuáles se pueden obtener utilizando las opciones "Tasa de Fallos" y "Rendimiento" dentro del menú "Estadísticas".

```
**********************************************
Tasa de Fallos
**********************************************
El Proceso Proceso3 ha efectuado 7 accesos a memoria
obteniendo 1 aciertos y 6 fallos de página
Con una tasa de fallos del  85  %

El Proceso Proceso2 ha efectuado 7 accesos a memoria
obteniendo 2 aciertos y 5 fallos de página
Con una tasa de fallos del  71  %

El Proceso Proceso1 ha efectuado 7 accesos a memoria
obteniendo 3 aciertos y 4 fallos de página
Con una tasa de fallos del  57  %
**********************************************
Rendimiento del sistema
**********************************************
Se han producido un total de 21 accesos a memoria
contabilizando un total de 6 aciertos de página y
15 fallos de página. Con una tasa de fallos del  71  %
```

Figura 6. Estadísticas de la simulación.

3. Aplicación del simulador a las prácticas de Sistemas Operativos

En la sesión práctica relacionada con la gestión de memoria, se sugiere al alumno que siga los pasos mencionados en el apartado anterior. A partir de un fichero de simulación, el alumno debe configurar el sistema de varias formas interpretando los resultados obtenidos. El alumno debe realizar las siguientes tareas.

- Cargar el fichero de simulación.
- Configurar el sistema. Se sugiere el uso de un tamaño de memoria de 8 KB, un tamaño de página de 1 KB y alguno de los algoritmos de reemplazo global (LRU o FIFO). Respecto el planificador, se puede considerar al principio, una única cola (cola 0).
- Comenzar la simulación. Para un mejor control, se puede utilizar un modo "paso a paso" y la visualización simultánea de los resultados en la "Ventana de Direcciones Físicas". Se sugiere al usuario que antes de acceder a la siguiente dirección lógica, analice la información disponible y determine el proceso y si provocará un fallo de página, comprobando posteriormente qué página ha sido reemplazada y la dirección física generada.
- Evaluar los resultados de la simulación. Una vez finalizada la simulación se trata de comprobar los resultados obtenidos que reflejan la tasa de fallos de cada proceso y el rendimiento global del sistema.

Con la información obtenida se plantean ejercicios relacionados con la práctica realizada, y cuyo objetivo es analizar el nivel de adquisición de conocimientos del alumno. A continuación se muestra un posible ejercicio:

- Determinar el estado final de la memoria según una configuración de tamaño de página de 1 KB, algoritmo LRU de reemplazo GLOBAL y tamaño de memoria de 8 KB. Para especificar la solución, se le ofrece al alumno la tabla adjunta.

Marco	Pid	Pag.
0		
1		
2		
3		
4		
5		
6		
7		

Tabla 1. Ejemplo de ejercicios basados en tablas.

Aunque dicho ejercicio, es relativamente "mecánico" de utilización del simulador, también se sugiere al alumno que previamente se realice de forma completamente manual, de forma que el alumno pueda contrastar su conocimiento de la materia, utilizando el simulador para solucionar posibles dudas. También pueden plantearse actividades de tipo reflexivo que permitan validad el nivel de conocimientos y el trabajo del alumno.

A continuación se muestran algunos ejemplos de estos ejercicios propuestos al alumno.

- Indique si existe alguna diferencia en la aplicación de ambos algoritmos (LRU y FIFO) y en qué consiste. Justifique la existencia de una misma tasa de fallos de páginas en ambos ejemplos.
- Encuentre un ejemplo de carga donde la aplicación de un algoritmo LRU devuelva un menor número de fallos de página, respecto un algoritmo FIFO.

4. Conclusiones

En este trabajo, se ha presentado una herramienta dirigida a la simulación de algoritmos de gestión de memoria. Dicha herramienta ha sido realizada en lenguaje Java, lo que facilita su utilización en entornos Web y su portabilidad a diferentes sistemas operativos. Asimismo, proporciona una interfaz visual que favorece su uso por parte de los alumnos. Tras tres años de experiencia los alumnos han aceptado de buen grado y manifestado la utilidad de dicho simulador como herramienta de autoaprendizaje.

La herramienta, además de reproducir información en un formato atractivo, invita al alumno a interactuar con la aplicación y permite fácilmente la modificación de algunos de los parámetros de entrada como el tamaño de la página o de la memoria principal. La herramienta es de gran utilidad para el alumno principalmente para la comprobación de ejercicios planteados y resueltos en clase. La experiencia, en líneas generales, ha resultado positiva a pesar del elevado coste de desarrollar una herramienta *ad hoc* de uso específico.

Como trabajo en vías de realización, se está considerando la integración de:

- Algoritmos de planificación basados en prioridades.
- Aspectos teóricos en el funcionamiento del simulador
- Algoritmos de reemplazo de páginas alternativos.

Referencias

[1] A.B. Tucker et al., ACM/IEEE-CS Joint Curriculum Task Force. Computing Curricula 1991, ACM Press; IEEE Comp. Soc. Press. , 1991.

[2] ACM/IEEE Task Force on the Year 2001 Model Curricula for Computing , Computing Curricula 2001(CC-2001), obtenido el 10/11/1999 de la página web: http://www.computer.org/education/cc2001/.

[3] David Jones, Andrew Newman, A constructivist-based tools for operating systems education, Proceedings of EdMedia'2002, Denver, Colorado, June 2002.

[4] W A Christopher et a. (1993), The Nachos Instructional Operating System. Proceedings of the Winter 1993 Usenix Technical Conference, pp 481-489.

[5] Jeff Magee and Jeff Kramer. Concurrency: State Models & Java Programs.

[6] Llopis Mengual, J.Espinosa Rodilla. Simulador de gestión de memoria. Proyecto final de carrera (PFC1), 1999.

[7] Innovations for Education in Information Technology through Multimedia and Communication Networks. Proyecto Socrates de Red Temática. http://www.eui.upv.es/ineit-mucon/.

[8] Programa de Simulación de Gestión de memoria. http://www.eui.upv.es/ineit-mucon/ Applets /mem_simulator/Applet.htm.

Tecnologías de la información en la gestión empresarial

Gestión de Clientes en el marco de los Portales Corporativos. Ensayo de enseñanza interdepartamental en la EUEEZ. Integración de las visiones empresarial y tecnológica.

Javier Gutiérrez
E.U..Estudios Empresariales
Dept. de Informática e Ingeniería de Sistemas
Universidad de Zaragoza
e-mail: adsogu@posta.unizar.es

María Jesús Lapeña
E.U..Estudios Empresariales
Dept. de Informática e Ingeniería de Sistemas
Universidad de Zaragoza
e-mail: mlape@posta.unizar.es

Pilar Urquizu
E.U..Estudios Empresariales
Dpto. de Econocmía y dirección de Empresa
Universidad de Zaragoza
e-mail: purquizu@posta.unizar.es

Resumen

En la Escuela Universitaria de Estudios Empresariales de Zaragoza estamos llevando a cabo una experiencia interdepartamental para la didáctica integrada de aspectos empresariales y tecnológicos en relación a la gestión de clientes en el marco de los portales corporativos. Esta experiencia, se basa en un caso común, de índole práctica, planteado en cada asignatura implicada desde su punto de vista específico. Estos puntos de vista van desde el análisis y estrategias de segmentación de mercado, a la implementación de dichas estrategias en forma de bases de datos relacionales y consultas SQL, que finalmente se integran en un portal de empresa basado en tecnologías de Internet.

1. Gestión de clientes en el marco de los Portales corporativos

Los Portales Corporativos [3] o Portales de Empresa están adquiriendo una importancia fundamental como vertebradores del sistema informático de las empresas [1 y 4], al unificar el acceso a información, aplicaciones, trabajo cooperativo y todo tipo de recursos y servicios. Esta visión es la consecuencia final de la evolución de intranets y extranets, redes privadas virtuales, sedes web y sistemas de información tradicionales [3]. El entorno más adecuado para el desarrollo de estos Portales lo proporciona sin duda la tecnología procedente de Internet.

Como entorno unificado, en estos portales se integran diversos aspectos de gestión empresarial y toma de decisiones. No sólo proporcionan el marco de trabajo unificado, sino además la posibilidad de trabajar de forma integrada con información procedente de diversos departamentos o secciones de la empresa.

El caso de la gestión de clientes no es una excepción. Desde el emergente campo del llamado CRM (Customer Relational Management) surgen propuestas y puntos de vista de mayor alcance de los hasta ahora considerados en el mundo del marketing. La tecnología disponible hoy en día es responsable, en buena parte, de esta evolución, al permitir una gestión mucho más depurada de la que se venía haciendo hasta este momento y el acceso unificado a información de clientes procedente de distintos departamentos.

Debido a este estrecho maridaje entre marketing y tecnología, creemos fundamental que el alumno de empresariales conozca no sólo los aspectos de análisis de mercados sino también posibles tecnologías subyacentes. En este artículo presentamos una experiencia didáctica basada en el seguimiento de un caso supuesto. Este caso es extremadamente sencillo, tanto desde el punto de vista del marketing como desde el punto de vista informático. La pretensión con esta experiencia es que el alumno pueda acceder a distintos puntos de vista sobre un mismo problema: el punto de vista del marketing relacional, el de las bases de datos y el de la programación de portales corporativos.

Este artículo se estructura como sigue. En primer lugar se plantea el problema de la gestión

de clientes en el marco de los portales corporativos.

En el segundo apartado presentamos la didáctica de esta gestión en nuestra escuela, didáctica que desarrollamos mediante un caso de segmentación de mercado, de forma integrada desde dos departamentos y cuatro asignaturas. En este capítulo se destaca además la importancia de integrar conocimientos conceptuales del dominio de la empresa con conocimientos tecnológicos de Sistemas de Información e Internet. Los apartados tercero, cuarto y quinto presentan las diferentes asignaturas implicadas en esta experiencia, así como la relevancia del problema en cada una de las asignatura y particularmente cómo se enmarca en ellas el caso de ejemplo. En el apartado sexto hacemos una valoración de nuestra aproximación didáctica y presentamos nuestras ideas sobre futuras actuaciones en didáctica integrada.

2. Presentación del caso de ejemplo

En nuestra didáctica integrada suponemos la existencia de una empresa mediana dedicada a la venta de material para deportes de aventura y de naturaleza, y a la organización de actividades guiadas en ese ámbito.

Esta empresa tiene un portal corporativo donde se integra un módulo de toma de decisiones y en particular un modulo de gestión de clientes y segmentación de mercado. Nuestro interés en este momento se centra en agrupar a los clientes de nuestro negocio en dos segmentos, con el objetivo de difundir selectivamente información sobre actividades guiadas. Nos referiremos a estos dos segmentos como "clientes de turismo activo" y "clientes de deporte de riesgo".

Para realizar esta segmentación tendremos en cuenta diversas características de nuestros clientes, tales como las actividades en las que hayan participado previamente, o el tipo de material que hayan adquirido en nuestra tienda. Para ello utilizamos información procedente de diversos departamentos de la empresa: tienda, actividades y gestión de clientes (esta integración de información procedente de actividades diversas de la empresa es una de la características fundamentales, si no la más importante, de los portales corporativos.)

El alumno deberá establecer rangos de valores para las variable determinantes de los segmentos de mercado, tras haber comprendido por qué se han escogido esas variables y no otras. Esto lo realizará en el ámbito de las asignaturas del Departamento de Economía y Dirección de Empresa. En las asignaturas del Departamento de Informática e Ingeniería de Sistemas realizará una implementación de estas reglas. En la asignatura Informática II el alumno creará una base de datos adecuada para recoger la información de clientes, productos y actividades, así como las sentencias SQL y formularios necesarios para implementar las reglas y la asignación de valores a las variables. Finalmente, en la asignatura de Lenguajes de Programación integrará esta base de datos como parte del Portal Corporativo, permitiendo que cualquier usuario autorizado del Portal pueda ejecutar las consultas correspondientes a las reglas y en su caso modificar las propias reglas o sus variables.

3. Gestión de clientes (visión empresarial)

En la docencia impartida desde el Departamento de Economía y Dirección de Empresas tienen una relevancia particular los aspectos relacionados con la aplicación de las nuevas tecnologías al marketing, fundamentalmente porque permiten profundizar en el conocimiento y tratamiento de la información sobre los clientes para realizar una gestión más depurada.

Estos conocimientos se desarrollan fundamentalmente en dos asignaturas: *Procesos Básicos de Producción* y *Análisis de Mercados*. En este marco se concede especial importancia a la Gestión Relacionál de Clientes (CRM, Customer Relational Management) o Marketing Relacional. En este sentido resulta especialmente importante conocer cómo realizar una adecuada segmentación de mercado. El propósito de esta segmentación es realizar un marketing selectivo que fidelice a los clientes y permita aumentar la cuota de mercado.

Hay que destacar la importancia de definir correctamente el problema y comprender la necesidad de la tecnología para implementar segmentaciones adecuadas que permitan personalizar las estrategias de Marketing y crear relaciones a largo plazo con los clientes.

El alumno deberá comprender, guiado por el profesor, cuáles son las variables a tener en cuenta para realizar dicha segmentación, y determinar qué intervalos de valores son los que permitirán realizar dicha segmentación de forma adecuada.

A continuación mostraremos un ejemplo de uno de los posibles resultados de dicha segmentación. A este resultado se ha llegado después de un análisis en el que se han determinado cuáles son las variables que es preciso tener en cuenta (propuesto por el profesor) y posteriormente qué rango de valores va a ser el que nos proporcione la segmentación deseada. Análisis más completos y realistas darán resultados más complejos. Hay que tener en cuenta que se trata de un ejemplo muy sencillo, debido sobre todo a que se pretende que el alumno pueda encontrar el hilo de continuidad de su análisis con su confección informática como una base de datos en un portal corporativo.

Determinación del segmento de "clientes de turismo activo"

- Clientes que han contratado cualquier actividad comprendida en las modalidades de : *"Excursión ecuestre", "Itinerario botánico", "Senderismo"*
- Clientes que han adquirido productos de alguna de estas secciones: *"Senderismo"*

Determinación del segmento de "clientes de deporte de riesgo"

- Clientes que han contratado cualquier actividad comprendida en las modalidades de: *"Escalada", "Esquí de montaña", "Expedición"*
- Clientes que han adquirido productos de alguna de estas secciones: *"Escalada", "Esquí"*

4. Bases de datos (nivel tecnológico)

En la asignatura Informática II abordaremos todo lo concerniente al diseño y desarrollo de bases de datos referente al caso que nos ocupa. De hecho, esta asignatura está centrada en el estudio de las bases de datos, ya que es obvia su importancia en el mundo de la empresa actual.

La asignatura tiene una parte teórica, en la que se explican conceptos y nociones fundamentales y se tratan todos los aspectos teóricos relacionados con el análisis y diseño de bases de datos. Tiene también una parte práctica, que se desarrolla en paralelo a la parte teórica, que tiene como objetivo concreto la implementación de bases de datos en Access. Aunque de forma simplificada y siempre en un nivel básico acorde a nuestras necesidades y posibilidades, se abordará cada una de las fases del ciclo de vida de las bases de datos: análisis, diseño, desarrollo, pruebas y mantenimiento. Se insistirá siempre en la importancia de seguir un método, y en la necesidad de hacer un control de calidad y generar una correcta documentación.

El enfoque que damos al estudio de las bases de datos viene dado por el entorno en el que se encuadra la asignatura: la diplomatura en empresariales; por ello, los ejemplos y casos que se desarrollan corresponden al entorno empresarial, tratando de buscar el equilibrio entre problemas clásicos de gestión y problemas emergentes como CRM.

En el caso del ejemplo (Gestión de Clientes en el marco de los Portales Corporativos) los alumnos construirán una base de datos y almacenarán los datos correspondientes a clientes, productos y actividades en sendas tablas, con la siguiente estructura (a modo de ejemplo):

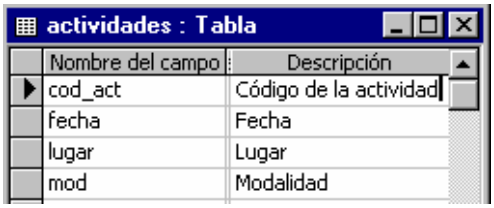
clientes : Tabla

Nombre del campo	Descripción
cod_cli	Código del cliente
apel1	Primer apellido
apel2	Segundo apellido
nom	Nombre
correo_e	Correo electrónico
tel	Teléfono
dir	Dirección
edad	edad

productos : Tabla

Nombre del campo	Descripción
cod_prod	Código del producto
denom	Denominación
marca	Marca
seccion	Sección

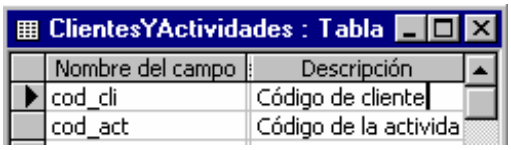
Los posibles valores de la sección son:, "Senderismo", "Escalada" y "Esquí".

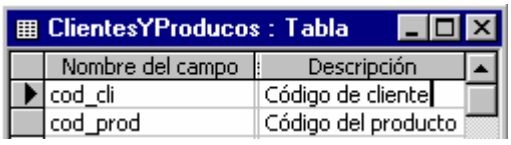


Los posibles valores de la modalidad son: "Excursión ecuestre", "Itinerario botánico", "Senderismo", "Escalada", "Esquí de montaña" y "Expedición".

El diseño de estas tablas y los datos a considerar será el primer problema a resolver por los alumnos, una vez hayan decidido las variables a tener en cuenta para llevar a cabo la segmentación de mercado.

Tienen que construir, además, otras dos tablas que representan las relaciones de los clientes con las actividades y con los productos. El alumno creará las restricciones de clave foránea necesarias.





Asimismo, los alumnos deberán diseñar e implementar consultas y formularios sobre dichas tablas, construyendo soluciones integradas para la gestión de clientes. Como tema colateral, a lo largo de todo el desarrollo, se insistirá particularmente en considerar con especial interés el tema de la interfaz de usuario, aplicando normas de estilo que nos lleven a diseños de calidad.

A modo de ejemplo, la implementación como SQL de la primera de las cuatro reglas del apartado anterior sería así:

```
SELECT clientes.*
FROM    clientes    INNER    JOIN
(actividades      INNER      JOIN
ClientesYActividades            ON
actividades.cod_act              =
ClientesYActividades.cod_act)   ON
clientes.cod_cli                 =
ClientesYActividades.cod_cli
WHERE
(((actividades.[mod])="Excursión
ecuestre"                        Or
(actividades.[mod])="Itinerario
botánico"                        Or
(actividades.[mod])="Senderismo")
);
```

Una vez diseñadas las consultas, se construirán formularios como éste que nos permitan seleccionar los datos de los clientes requeridos:

Hay que decir además que el desarrollo de esta base de datos en el entorno de la asignatura Informática II va más allá de lo estrictamente necesario para la experiencia interdepartamental del caso ejemplo. Así, todas las consultas y formularios se integrarán en una completa aplicación de gestión de clientes, para cuya construcción los alumnos han de poner en práctica todos los conocimientos que han ido adquiriendo en la asignatura.

5. Programación de Portales (nivel tecnológico)

En la asignatura de Lenguajes de Programación se estudia el desarrollo de portales como herramienta integradora. Se trata de una asignatura predominantemente práctica, en la que el desarrollo de un portal justifica la exposición teórica de los temas propuestos para la asignatura.

En una primera unidad didáctica se enmarca el problema de la programación y la creación de portales en el ámbito de la empresa, justificando las distintas tecnologías estudiadas desde su uso para marketing, gestión de decisiones, venta on-line, recopilación de información de clientes y trabajo cooperativo. En sucesivas unidades se hace una introducción tecnológica a Internet, Buscadores Web, codificación HTML y uso de editores web. Como complemento a estos conceptos se introducen nociones de multimedia y estilo de diseño. Otro capítulo importante del programa consiste en la presentación de una Metodología de Desarrollo de Sedes Web, que resulta de la simplificación de las principales Metodologías de Sistemas de Información utilizadas hoy en día.

La parte más directamente implicada en el problema de la programación en entornos de red trata sobre arquitecturas cliente servidor y la programación en ambos lados, ofreciéndose conocimientos básicos de JavaScript para la programación del lado cliente. En cuanto a la programación de lado servidor se centra la atención en el problema de las Bases de Datos, orientándose hacia la tienda virtual y las bases de datos de clientes y de control de productos. Para ello se estudian formularios web y nociones muy básicas de servidores web y ASP.

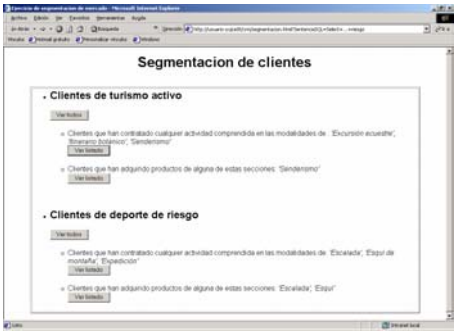
La práctica del curso consiste en el desarrollo guiado mediante especificaciones detalladas, de una Sede Web en sentido muy amplio, que puede ser concebida como un pequeño portal. Esta Sede está inspirada en un caso real de negocio virtual de gran resonancia nacional, *barrabes.com*. En esta Sede o portal se desarrollan varios módulos, funcionando a un tiempo como Intranet, Extranet y Sede Web. Se concede gran importancia al problema del acceso a bases de datos a través de Internet.

En este contexto el caso práctico que presentamos en este artículo aparece como una práctica más de las realizadas durante el curso, consistiendo en el trabajo con los elementos necesarios parta realizar el acceso: creación de formularios web, comprensión de un pequeño script ASP que permite la consulta a la base de datos y finalmente la colocación de la base de datos Access en el servidor web (IIS). A partir de este momento el alumno puede efectuar las consultas preparadas, a través de una página web integrada en el portal.

En el desarrollo de esta práctica el alumno desarrollará una página HTML con formularios, uno para cada consulta deseada. Cada uno de estos formularios tiene un campo oculto, con una sentencia SQL. A modo de ejemplo presentamos el formulario correspondiente a la sentencia SQL mostrada en el capítulo anterior

```
<FORM      action="http://usuario-
yujce0t/crm/consulta.asp"
method="post">
 Clientes que han contratado
 cualquier actividad comprendida
 en las modalidades de:
 <I>'Excursión ecuestre',
 'Itinerario
  botánico',Senderismo' </I> <BR>
 <input type="hidden"
 name="SentenciaSQL"
 value="SELECT  clientes.* FROM
 (actividades INNER JOIN
 ClientesYActividades ON
 actividades.cod_act =
 ClientesYActividades.cod_act)
 INNER JOIN clientes ON
 ClientesYActividades.cod_cli =
 clientes.cod_cli WHERE
 (((actividades.mod) = 'Excursión
 ecuestre')) Or
 (((actividades.mod) =
 'Itinerario botánico')) Or
 (((actividades.mod) =
 'Senderismo'))">
 <input type="submit" value="Ver
        listado">
</FORM>
```

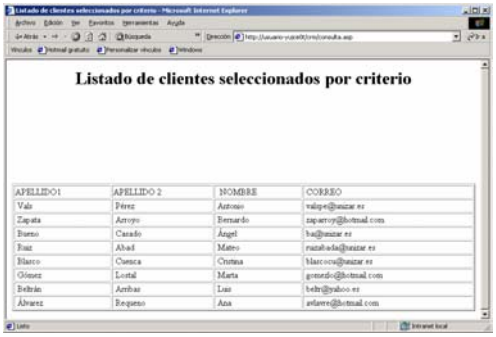
Las sentencias SQL utilizadas para construir los formularios de esta página serán las que el alumno ha desarrollado en la asignatura Informática II. El aspecto de la página Web que debe desarrollar es el siguiente:



Los formularios de esta página llaman a un script ASP que establece una conexión con la base de datos Access. El script recoge la sentencia SQL que le ha enviado el formulario Web y la envía a la base de datos. Con el resultado de la consulta, el script genera una página Web que es devuelta al cliente. La base de datos consultada es la que el alumno ha creado en la asignatura Informática II. No se le pide al alumno que programe este script, sino sencillamente que lo comprenda y sepa realizar alguna pequeña modificación en él:

```
<HTML><HEAD><TITLE>Listado de
clientes seleccionados por
criterio</TITLE></HEAD><BODY>
<%
Dim laSentencia

laSentencia =
Request.Form("SentenciaSQL")

Set conexion =
CreateObject("ADODB.Connection")
conexion.Open
"PROVIDER=MICROSOFT.JET.OLEDB.4.0
; DATA SOURCE=
C:\WebShare\Wwwroot\crm \crm.mdb"
Set misDatos =
conexion.Execute(laSentencia)
Response.Write
("<CENTER><H1>Listado de clientes
seleccionados por
criterio</h1><table BORDER=2
WIDTH='100%'> </CENTER>")
Response.Write("<tr><td>" & "
APELLIDO1  " & "</td><td>" & "
APELLIDO 2 "  & "<td> " & "
NOMBRE " & "</td>" & "<td> "
& "CORREO" & "</td></tr>")

while (not misDatos.Eof)

Response.Write("<tr><td> " &
misDatos("apel1") &
"</td><td> " &
misDatos("apel2") & "<td> "
& misDatos("nom") & " </td>"
& "<td> " &
misDatos("e_mail") &
" </td></tr><br>")
      misDatos.MoveNext
wend

Response.Write ("</TABLE>")

conexion.Close
Set conexion = Nothing
%>
</BODY></HTML>
```

De esta forma el alumno ha utilizado las sentencias SQL que ya había realizado, insertándolas en formularios Web para consultar, gracias al uso de un script sencillo, a la base de datos también creada previamente. El resultado final para cada una de las consultas es una página Web, generada por el propio script. Esta página contendrá los datos de los clientes seleccionados, como podemos ver en la imagen:



6. Conclusiones

Pensamos que la experiencia didáctica que presentamos en este artículo supone una efectiva

innovación didáctica, basada fundamentalmente en las siguientes ideas:

- Se trata de una **experiencia interdepartamental e interasignatura**, que permite mostrar al alumnos la cohesión e interdependencia de diversas asignaturas de su currículo.
- Se trata de una **experiencia internivel**, que permite al alumno comprender la continuidad entre el nivel teórico de los análisis de mercado y el nivel de implementación informática. Se transmite así al alumno la idea de que los conocimientos tecnológicos que adquiere en la diplomatura están cohesionados con la formación teórica a la que dan soporte; en el caso que nos ocupa, mostrando la necesidad de las tecnologías de la información para desarrollar las estrategias de Marketing.
- Asimismo nuestra orientación sigue **la línea de las metodologías docentes del caso** [5], al presentar al alumno un problema equivalente a un problema real típico que en distintas prácticas debe ir resolviendo bajo el asesoramiento y orientación del profesor. Creemos que se trata de una perspectiva adecuada, ya que, como se indica en [5], "para enseñar el uso de una ciencia es conveniente comenzar por casos en que al educando le resulte relativamente fácil estructurar el problema en términos de las variables de esa ciencia. lo que significa utilizar casos más bien próximos a problemas teóricos"
- Finalmente, se trata de una experiencia inspirada en un caso real, el de una empresa prototípica en nuestra comunidad, barrabes.com, que ha evolucionado desde un pequeño negocio familiar en un pueblo de montaña ("donde se acaba la carretera") hasta llegar a ser una empresa líder en comercio electrónico a nivel nacional, e incluso a ser parte constituyente de la moderna plataforma de e-business Walqa [2].
- Creemos que esto supone un poderoso incentivo didáctico para el alumno de empresariales de nuestra comunidad

No obstante, esta experiencia se encuentra todavía en pleno desarrollo en este curso 2002 / 2003, lo que nos impide efectuar una evaluación y un análisis restrospectivo de su efectividad. Se trata en todos los casos de asignaturas anuales. La respuesta, tanto a nivel de profesorado como de alumnos está siendo muy positiva; los alumnos agradecen la presentación de casos que les acerquen a su futura realidad profesional y que requieren la puesta en práctica de conocimientos correspondientes a materias de distinta índole. Hemos previsto efectuar una encuesta a final de curso, que nos permita medir el nivel de satisfacción de alumnos y profesores.

Creemos que se trata de una verdadera innovación docente en los Estudios Empresariales, que, en general, siempre han adolecido de formación en Informática; desconocemos la existencia de experiencias en esta línea, que presenta grandes perspectivas y posibilidades docentes y que requiere, por tanto, un impulso para su desarrollo.

En cuanto a líneas futuras de actuación integrada en la Escuela de Empresariales de Zaragoza, hemos de destacar que en el nuevo Plan de Estudios pendiente de aprobación, está prevista una línea de especialización que, bajo el título "Marketing y Nuevas Tecnologías", pretende englobar contenidos de Marketing y de Informática con objeto de ofrecer al alumno una formación que le capacite para la realización de análisis de clientes, fidelización de clientes y comercialización de productos a través de la red. Concretamente, en dicha especialidad se integran las asignaturas de Informática "Bases de Datos y Sistemas de Información" y "Nuevas Tecnologías de la Información".

Referencias

[1] Aguila, Ana Rosa del, *Comercio electrónico y estrategia empresarial*, Madrid, Rama, 2000

[2] Arnal, José Carlos, *Sueños Electrónicos*, Zaragoza, Institución Fernando el Católico, 2002

[3] Joyanes Aguilar, Luis, *Portales de 3ª generación*, en DATA.TI, Diciembre 2002, p. 54 58

[4] O'Brien, James A., *Sistemas de información gerencial*, McGraw Hill, 1999

[5] Orti González, Ana María, Metodologías para la Formación de Emprendedores: *El Método del Estudio del Caso, el Método de las Situaciones y el Estudio de Incidentes Críticos*,http://www.unsam.edu.ar/unsam/secyt/DVE2001/index.htm, 2001

Telemática

Experiencia en la aplicación de un entorno virtual como apoyo a la docencia de laboratorios presenciales

Javier Macías, José Javier Martínez, José María Gutiérrez, Roberto Barchino
Dpto. Ciencias de la Computación
Universidad de Alcalá
28871 Alcalá de Henares
e-mail: {javier.macias, josej.martinez, josem.gutierrez, roberto.barchino}@uah.es

Resumen

Se presenta una experiencia de aplicación de un entorno virtual en un laboratorio presencial, indicando las razones de tal selección, la configuración del entorno, su explotación y los resultados obtenidos.

1. Introducción

Existe en la actualidad un número significativo de Universidades españolas que utilizan entornos virtuales, como por ejemplo la UNED [2], la UOC [4], la Universidad de las Islas Baleares [11] o el consorcio ADA-Madrid [10] (en el cual participan seis universidades madrileñas).

Sin embargo, la mayoría de las veces estos entornos sólo se explotan en asignaturas impartidas a distancia o de una forma semipresencial, ya sea por tratarse de una universidad a distancia, como es el caso de la UNED o la UOC, por la especial casuística de los alumnos a los que van destinadas, como ocurre en la Universidad de las Islas Baleares debido a los problemas de desplazamiento que provoca su insularidad, o por la propia naturaleza de las asignaturas, como es el caso de las asignaturas ofertadas por ADA-Madrid, que son de libre elección y su objetivo es ampliar el horizonte académico de los alumnos.

Además, debido a la importancia estratégica que supone el uso de las nuevas tecnologías de la información y la comunicación [6], la puesta en marcha de los entornos virtuales se ha hecho en ciertos casos más como inversión pensando en el futuro que con vistas a una explotación inmediata eficiente.

Por otro lado, los nuevos entornos producen desconfianza en muchos profesores, al tratarse de una herramienta para la enseñanza que nunca experimentaron como alumnos [9].

A todo ello debemos añadirle el hecho de la juventud en que todavía se encuentran las plataformas de soporte de estos entornos virtuales y el proceso de desarrollo en que todavía se encuentran los estándares que permitan su interconexión y, por tanto, respalden las inversiones realizadas en la producción de material educativo. En cualquier caso, es de destacar la cada vez mayor convergencia de funcionalidades de estos entornos y la tendencia a un agrupamiento de plataformas [13].

2. Objetivos

Aprovechado la disponibilidad en el Departamento de algunos de los entornos virtuales más utilizados y los conocimientos y experiencia de los autores en dichos entornos, se decidió la puesta en marcha en el curso 2001/02 de uno de ellos con los siguientes objetivos:

- El entorno se emplearía en una asignatura de carácter troncal u obligatorio.
- La asignatura debería tener unas características tales que el uso del entorno produjera unas ganancias significativas en uno o más aspectos docentes.
- El entorno debería ser integrador, en el sentido de ser capaz de reunir la mayor cantidad de herramientas de apoyo a la docencia basadas en las tecnologías de la información y la comunicación, incluidas aquellas ya utilizadas en cursos anteriores si procediera

(como el correo electrónico o la transferencia de ficheros).
- El esfuerzo requerido tanto al profesor como al alumno para el aprendizaje y uso del entorno tendría que ser razonable.
- Aproximadamente a mitad de curso, se realizaría una evaluación del entorno virtual, tanto para poder realizar una valoración del funcionamiento del mismo como para tomar las medidas correctoras oportunas en caso de que se detectaran problemas que hubieran pasado desapercibidos.

3. Elección de la asignatura y la plataforma

Una vez establecidos los objetivos, las primeras acciones a desarrollar consistían en la elección de la asignatura y la plataforma.

Como asignatura se seleccionó "Laboratorio de Metodología de la Programación", impartida en el segundo cuatrimestre a los alumnos de primer curso de Ingeniería Técnica Informática de Sistemas e Ingeniería Técnica Informática de Gestión de la Universidad de Alcalá. Dicha asignatura es de carácter obligatorio y tiene una carga lectiva de tres créditos.

Las razones por las cuales se eligió una asignatura de laboratorio de programación fueron las siguientes [5, 12]:

- Distribución de enunciados. Los enunciados suelen variar total o parcialmente cada año, entre otras razones para evitar que el alumno copie las prácticas entregadas en cursos anteriores. Frecuentemente los enunciados van acompañados de código que el alumno debe modificar o utilizar y/o de ficheros con datos de prueba. Asimismo, es reseñable que en el caso de enunciados de nueva creación, existe mayor posibilidad de que, durante el desarrollo de la práctica por parte de los alumnos, se descubran errores de diverso tipo que pasaron desapercibidos al profesor. Todo ello hace difícil la edición de libros de prácticas y justifica el disponer de una plataforma que permita una comunicación ágil entre profesores y alumnos.
- Entrega y corrección de prácticas. Usualmente, los alumnos deben entregar una o más prácticas en las fechas que se establezcan. El profesor, tras recoger las prácticas, deberá realizar acciones como la corrección manual o automática, el uso de aplicaciones detectoras de plagio y la comunicación a los alumnos del resultado. Todo ello genera una cantidad importante de trabajo, parte del cual se puede calificar como trabajo administrativo.
- Problemática de horarios. Los laboratorios suelen ser semanales, de unas dos horas de duración (como ocurre con la asignatura seleccionada). En ciertos casos, los horarios de laboratorios no son adyacentes a los de las clases teóricas, lo que puede dificultar la asistencia de los alumnos, en particular de aquellos que también trabajan. Esto ocasiona un número de faltas mayor al esperable.
- Deficiencias en infraestructura. Uno de los objetivos generales de todo laboratorio suele ser el trabajo en equipo. Sin embargo, la mayoría de las veces los laboratorios sólo disponen de un ordenador por equipo de alumnos, lo cual supone que dos o tres personas estarán trabajando sobre el mismo ordenador en el momento de codificar. Este hecho, además de ser contrario a la propia naturaleza de la programación modular, ocasiona un aprovechamiento deficiente del tiempo de clase, por lo que no suele ser suficiente para terminar las prácticas. Como consecuencia, frecuentemente los alumnos deben continuar trabajando en casa, dificultándose la comunicación entre los componentes del grupo y con el propio profesor.
- Demanda de tutorías. Aunque, al igual que para las tutorías de clases teóricas, existe el problema común de posible coincidencia de éstas con el horario de otras clases, en el caso de los laboratorios el problema se acentúa, ya que las tutorías para laboratorios suelen ser más frecuentadas que las tutorías para clases teóricas. Las razones más obvias para esta mayor utilización vienen dadas por un menor contacto con el profesor de prácticas en horario de clase (una vez por semana) y por la entrega periódica de prácticas, que marcan frecuentemente los ritmos y picos de

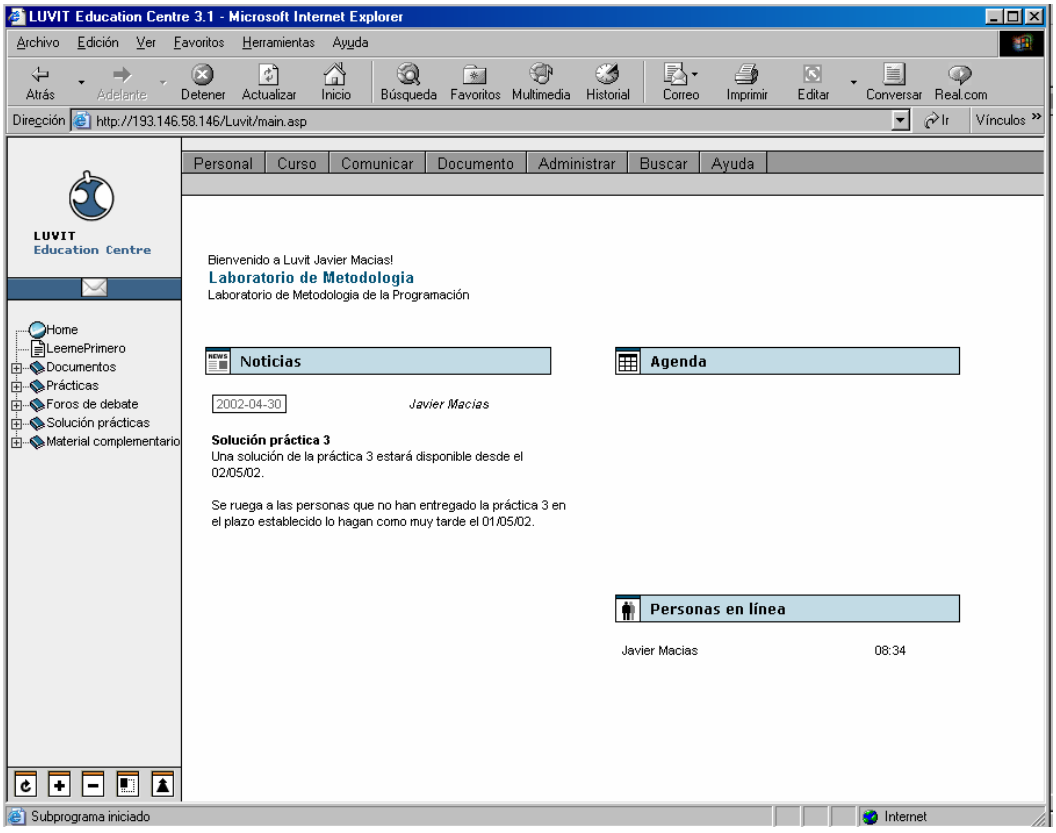


Figura 1. Página inicial del curso Luvit de Laboratorio de Metodología

utilización de dichas tutorías. En particular, los alumnos demandan una rápida respuesta a sus cuestiones sobre programación, ya que una sola duda o problema puede ocasionar que una práctica completa no funcione, o, incluso, que no puedan ni llegar a la codificación de una sola línea por dificultades relacionadas con el análisis o el diseño.

Los alumnos de la asignatura seleccionada estaban divididos en ocho grupos por titulación de 30 a 35 alumnos por grupo, dado el tamaño físico y número de máquinas de los Laboratorios. Se decidió incluir en el entorno a sólo dos grupos (denominados A2 y A5), por lo que el total de alumnos incluidos en la experiencia era de aproximadamente 65. Para simplificar la gestión del entorno y mantener una mayor animación en los foros, se acordó incluir a ambos grupos en el mismo entorno virtual. Además, bastó un solo profesor para atender el curso durante su explotación.

Respecto a la elección de la plataforma se buscó, en consonancia con los objetivos perseguidos, una de fácil uso y con disponibilidad inmediata. Ello nos llevó a descartar tanto cualquier diseño propio a medida (aun usando componentes de alto nivel) como soluciones externas que pudieran requerir una instalación compleja, necesitaran adaptación o no dispusieran del soporte técnico adecuado. Por tanto, nos decidimos por utilizar una de las plataformas disponibles en el mercado de calidad contrastable.

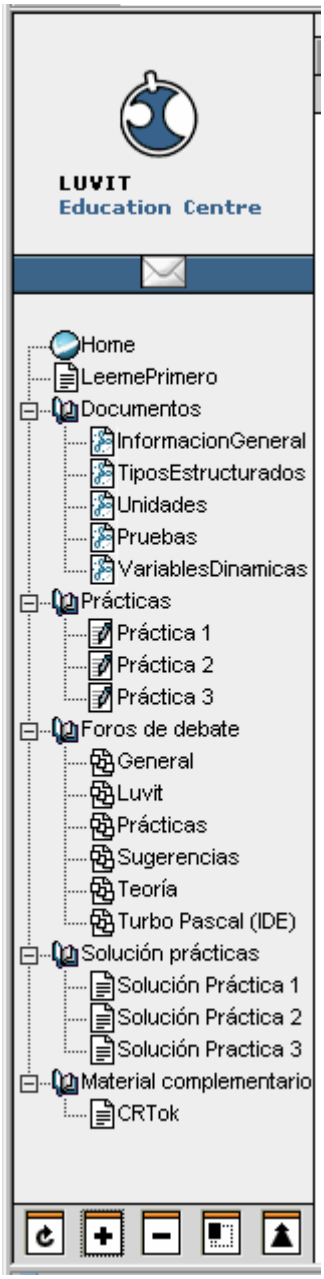


Figura 2. Árbol de navegación de Luvit

En concreto, elegimos Luvit® [7], debido a su disponibilidad en el Departamento, la experiencia que ya poseíamos con ella y a su claridad (Figura 1) y facilidad de uso, fundamentada en el árbol de navegación (Figura 2) y similar al árbol del Explorador de Windows®. Una comparativa de Luvit con otras plataformas se encuentra en [3].

4. Fases

A continuación se detallan las fases desarrolladas en la experiencia aquí presentada.

4.1. Preparación del curso

La preparación del curso exige cierto trabajo, aunque parte de él puede ser aprovechado en los siguientes años.

En primer lugar, y suponiendo que la plataforma que dará soporte a los cursos está ya instalada, hay que dar de alta (crear) el curso, indicando entre otros datos el nombre del curso, el del profesor y las fechas de inicio y fin del mismo.

Seguidamente se procede a dar la estructura inical del curso. Se inserta un mensaje de bienvenida y un documento de introducción al curso (denominado generalmente como “Leeme Primero” o de alguna otra forma similar) y se crean las secciones principales, incluyendo los distintos foros y un mensaje inicial en cada uno de ellos indicativo del cometido del mismo. Aunque la plataforma lo permitía, se decidió no crear una sección específica para charlas planificadas (*chat*), ya que al tratarse de una herramienta síncrona, podría generar una cantidad de trabajo no asumible, al menos no el primer año.

Posteriormente, se cargan de alumnos en el curso. En Luvit los alumnos se identifican mediante una cuenta de correo electrónico (todos tienen al menos una proporcionada por la Universidad), siendo obligatorio también el nombre del alumno. Para obtener estos datos, los alumnos escribieron durante una clase su correo y nombre en un mismo fichero texto uno a uno (bastaron unos pocos minutos para que todos los alumnos pasaran por el ordenador donde debían introducir sus datos). Este fichero fue cargado fácilmente en Luvit tras un pequeño preproceso.

4.2. Presentación del curso y puesta en marcha

En una de las clases de Laboratorio, se realiza una breve presentación del curso a los alumnos, ya que debido a que se trata de alumnos de informática y que la asignatura pertenece al segundo cuatrimestre, se presupone que tienen suficiente destreza para manejar el entorno. Se les explica la forma de acceso, cómo recibirán la contraseña de entrada (en nuestro caso se decidió enviarla por correo electrónico, cosa que puede hacer automáticamente Luvit) y se les instó a que consultaran en primer lugar el documento "LeemePrimero" del curso donde se les daban las instrucciones necesarias para su manejo. Además, se les insistió en que podían recurrir a las tutorías presenciales para explicarles el manejo del entorno a aquellos que tuvieran más problemas.

Se sugirió a los alumnos que formaran parejas en las que al menos uno de los componentes tenga acceso a Internet desde su casa, aunque también podían acceder al entorno desde las aulas de informática de la Universidad.

Tras la clase, se mandaron las contraseñas a los alumnos para que quien así quisiera pudiera utilizar el entorno desde el primer día.

4.3. Explotación

Una vez enviadas las contraseñas, se procede a la explotación del curso. Regularmente se accede al entorno para contestar las preguntas del foro y del correo, añadir nuevo material (como documentos, prácticas y soluciones), corregir las prácticas y comunicar noticias de interés para los alumnos (fechas de disponibilidad de documentos, enunciados y soluciones, fechas de entrega, fechas de exámenes y revisiones, etc.). También se mantuvo alguna charla interactiva con ellos cuando se coincidió en línea (en concreto, el número de charlas en las que participó el profesor fueron cuatro).

Es fundamental que el tiempo de respuesta a las cuestiones planteadas por los alumnos sea mínimo, idealmente inferior a 48 horas. Esta recomendación fue seguida fielmente durante el desarrollo del curso. Para lograr estos tiempos de respuesta, además de atender el curso durante el tiempo de tutoría presencial en los momentos en que no asistían alumnos, también se aprovechaban otros instantes en que el profesor se conectaba a Internet por alguna otra razón, ya fuera dentro o fuera de la Universidad. La carga de trabajo que ello supuso no fue excesiva en ningún momento, debido a que los mensajes de los alumnos se encontraban bastante repartidos a lo largo del cuatrimestre.

Como anécdota resaltar que un problema eléctrico acaecido en mitad de las vacaciones de Semana Santa hizo que se perdiera el acceso al curso. La caída se detectó y solucionó rápidamente (en menos de 48 horas) y se mandó un correo electrónico a los alumnos para indicarles por qué no habían podido acceder e informándoles del restablecimiento del curso.

4.4. Evaluación

Cuando el curso llevaba funcionando aproximadamente dos meses, se realizó una evaluación tal y como estaba prevista. Para ello se recopilaron datos objetivos obtenidos directamente del entorno virtual (número de mensajes en el foro, correos enviados, etc.). Asimismo, se pasó una encuesta anónima en papel a los alumnos durante una de las clases [8].

5. Secciones

Conviene realizar un repaso, siquiera somero, de las distintas secciones que se crearon en el curso y de su cometido. Para quién desee ver más a fondo el curso, se ha habilitado un usuario en el mismo [1].

- *LeemePrimero*. Documento en HTML que contiene las instrucciones básicas para el curso.
- Documentos. En esta sección se fueron añadiendo los distintos documentos con la teoría complementaria necesaria para poder realizar las prácticas. Asimismo se incluyeron las soluciones a los exámenes de junio y septiembre justo después de terminar éstos. Todos los documentos se proporcionaban en formato PDF.
- Prácticas. En esta sección el alumno accedía a los enunciados de las prácticas, junto con los archivos de código o prueba cuando

procedía. También desde aquí entregaba las prácticas y recibía las correcciones.

- Foros. Los foros son una herramienta muy importante, ya que constituyen la forma básica de realización de las tutorías remotas y de comunicación y ayuda entre los propios alumnos, con la ventaja de que una respuesta a una cuestión planteada por un alumno puede ser aprovechada por el resto de compañeros. Se crearon 6 foros, denominados *General* (para cuestiones generales), *Luvit* (para cuestiones relativas a la plataforma), *Prácticas* (para cuestiones relativas a las prácticas que había que entregar), *Sugerencias* (para plantear opiniones, críticas, sugerencias, etc.), *Teoría* (para cuestiones teóricas) y *Turbo Pascal (IDE)* (para cuestiones relativas al entorno de desarrollo utilizado en la asignatura).
- Solución prácticas. En esta sección se publicaban las soluciones propuestas para las prácticas, una vez terminado el plazo de entrega, para que el alumno pudiera compararla con su solución.
- Calificaciones. Esta sección se añadió tras el examen de junio para publicar las notas de los alumnos.
- Material complementario. Esta sección fue necesaria crearla durante el desarrollo del curso para incluir material que no tenía cabida en ninguna de las secciones anteriores. Por ejemplo, en esta sección se colocó un fichero que solucionaba ciertos problemas que surgían al ejecutar en ordenadores rápidos los programas compilados mediante el IDE utilizado en la asignatura.

6. Evaluación y datos finales

En el momento de la evaluación del curso se estimó que el número de alumnos que seguían regularmente la asignatura, ateniéndonos al criterio de haber tenido tres o menos faltas de asistencia, era de 21 en el grupo A2 y 24 alumnos en el grupo A5. Es de resaltar que en el recuento del grupo A5 se ha incluido a un alumno que, aunque no pudo asistir ningún día a clase debido a una operación que le imposibilitaba el andar, siguió el curso a través del entorno virtual y entregó sus prácticas regularmente.

Las medidas objetivas obtenidas en ese momento del entorno virtual, centradas principalmente en el uso de los foros, se muestran en la Tabla 1. Es de destacar que muchas de las personas que habían enviado mensajes (en concreto, cerca de un 60%), lo habían hecho más de una vez, por lo que pudimos deducir que su satisfacción por la respuesta obtenida fue alta, ya que de otra forma no hubieran vuelto a usarlos. Como dato particular, el alumno que no podía asistir había enviado tres mensajes hasta ese momento.

Descripción	Valor
Mensajes de alumnos a los foros	26
Alumnos que enviaron mensajes a los foros	14
Alumnos que enviaron más de un mensaje a foros	8
Máximo número de mensajes enviados a los foros por un mismo alumno	4
Correos recibidos por el profesor	3

Tabla 1. Medidas objetivas del curso en el momento de la evaluación

Respecto a la encuesta realizada, fue contestada por 31 alumnos, 18 de los cuales pertenecen al grupo A2 y 13 al grupo A5. Se debe subrayar que de los 31 alumnos, tan sólo uno indicó no haber utilizado la herramienta Luvit por problemas de conexión (problemas que posteriormente solucionó), quedando así excluido como individuo estadístico a analizar. Conviene decir que dicho alumno formaba pareja para las prácticas con otro alumno que sí se conectaba, por lo que en todo momento pudo obtener los documentos y enviar las prácticas.

Aunque el número de alumnos que ha contestado la encuesta no permite realizar un análisis estadístico determinante, sí que permite apreciar tendencias. En particular, cabe destacar los siguientes resultados:

- A la mayoría de los encuestados les ha parecido que la utilidad de Luvit en la asignatura "Laboratorio de Medodología de la Programación" era alta o muy alta. Sólo un alumno contestó que la utilidad le parecía baja, si bien se debe notar que este alumno

reseñó que tenía frecuentemente problemas al conectar con Luvit. Además, a la mayoría también les pareció útil la idea de extender el uso de la herramienta a otras asignaturas de laboratorio, aunque el porcentaje descendió hasta la mitad cuando se les preguntó por su extensión a asignaturas de teoría.

- A la mayoría les resultó sencillo aprender su manejo, aunque a dos alumnos les resultó muy difícil o difícil. Más de la mitad consideraron que no era necesario o que era poco necesario el haber utilizado una clase para enseñar su manejo. Además, todos consideran que, tras haber aprendido, no tienen dificultades para utilizarlo.
- Valoran positivamente la asincronicidad que la herramienta les proporciona a la hora de realizar consultas, descargar documentación y para la entrega de prácticas. Es también de destacar que más de la mitad de los encuestados califica de positiva la iniciativa de establecer una hora en la semana en la cual poder mantener charlas con los compañeros y el profesor.
- Dos terceras partes de los encuestados habían hecho uso de los foros, mientras que sólo un quinto de ellos habían utilizado las tutorías presenciales. Dos de las razones aportadas por los alumnos del porqué no utilizaban las tutorías presenciales fueron los horarios de las mismas y que les era más cómodo resolver dudas específicas utilizando los foros. Es importante señalar que casi todos los que hicieron uso de los foros habían encontrado útil alguna respuesta a preguntas planteadas por otros compañeros.
- Como era esperable, los alumnos que más favorablemente valoran la herramienta son en general los que disponen de conexión a Internet desde su casa, en particular los tres que disponen de ADSL. Seis de los alumnos no disponen de conexión desde su casa, aunque afirmaron que no tenían problemas para acceder ya que usaban las aulas de la Escuela Politécnica, aunque se quejan de su saturación. También expresaron la dificultad de recordar el nombre de la página de entrada a Luvit por lo que, para solucionar el problema, se creó un enlace directo a Luvit desde la página web del Departamento.

A final de curso se realizó una medida final resumen sobre algunos de los aspectos principales del curso (Tabla 2). Cabe remarcar, teniendo en cuenta el bajo tiempo de respuesta por parte del profesor, que en tres ocasiones los alumnos contestaran a preguntas de sus compañeros.

Descripción	Valor
Alumnos que entraron al menos una vez	45
Documentos publicados por el profesor	9
Enunciados	5
Soluciones propuestas por el profesor	5
Material complementario	3
Otros documentos	3
Noticias	20
Correos recibidos por el profesor	9
Charlas en las que participó el profesor	4
Mensajes enviados al foro por alumnos	35
Mensajes enviados al foro por el profesor	35
Mensajes de alumnos en los foros en respuesta a mensajes de otros compañeros	3

Tabla 2. Medidas finales del curso

Dada la especial importancia de los foros, se realizó un recuento detallado de los mensajes publicados en los mismos (Tabla 3). Como era de suponer, el mayor número de mensajes se concentró en el foro de prácticas. Es de destacar que el foro de teoría, aunque sólo tuvo tres mensajes de alumnos, éstos fueron de gran calidad, y, posiblemente, no hubieran sido planteados de no disponerse de este curso.

Foro	Mensajes		
	De alumnos	Del profesor	TOTAL
General	2	2	4
Luvit	4	2	6
Prácticas	24	21	45
Sugerencias	1	2	3
Teoría	3	5	8
Turbo Pascal	1	3	4
TOTAL	35	35	70

Tabla 3. Medidas detalladas de los foros

7. Conclusión

La experiencia aquí presentada muestra que la utilización de un entorno virtual para apoyar la docencia de laboratorios presenciales es bien recibida por los alumnos y ampliamente utilizada. Esta gran utilización es achacable en parte a que las funcionalidades proporcionadas por este tipo de herramientas se adaptan especialmente bien a los requisitos exigidos por las asignaturas de laboratorio.

El alumno valora positivamente la accesibilidad que esta herramienta le proporciona, tanto en lo que respecta al material como al profesor, incrementándose el número de dudas planteadas y resueltas. En concreto, prefiere usar los foros del entorno virtual para dudas puntuales y recurrir a las tutorías presenciales sólo cuando se trata de dudas generales.

Por otro lado, el trabajo que el uso del entorno virtual supone al profesor no es excesivo, en particular si no se hace uso extensivo de las herramientas síncronas, de las cuales la charla es su mayor exponente. Es de destacar que parte del trabajo ocasionado por el curso, en concreto, el mantenimiento de los foros, revierte debido a que una misma respuesta puede solucionar una duda común a un número significativo de alumnos. Además, es posible reutilizar la estructura del curso y posiblemente gran parte del material para años siguientes.

8. Futuras líneas de trabajo

Tras el éxito inicial obtenido, el siguiente paso consiste en ampliar la experiencia a todos los grupos de laboratorio de la asignatura. Ello implicará, entre otras cuestiones, la participación de más profesores, que deberán ser formados para que puedan obtener un rendimiento satisfactorio de la plataforma.

Posteriormente, y tras una valoración de los resultados y conocimientos obtenidos de esta experiencia a mayor escala, se podría ampliar la utilización de los cursos virtuales al resto de laboratorios relacionados con la programación u otros laboratorios o asignaturas cuyas características maximicen las ganancias del uso de este tipo de herramientas.

Referencias

[1] *Curso "Laboratorio de Metodología de la Programación" (plataforma Luvit)*. Dpto. Ciencias de la Computación, Universidad de Alcalá. http://193.146.58.146/Luvit/entrance /entrance.asp?cid=17 (usuario: invitado@jenui2003.es, contraseña: invitado).

[2] *Cursos virtuales de la UNED*. UNED. http://virtual0.uned.es/

[3] *Evaluation of Learning Management Systems*. University of Fribourg. http://www.edutech.ch/edutech/tools/ev2.php

[4] *La Universidad Virtual. Campus Virtual*. UOC. http://www.uoc.edu/web/esp /launiversidad /comoseestudia/campus.htm

[5] Labra, J., Morales, H. y Turrado, R. *Plataforma de enseñanza de lenguajes de programación a través de Internet: Proyecto IDEFIX*. Actas de las VIII Jornadas de Enseñanza Universitaria de la Informática (JENUI, 2002).

[6] *Ley Orgánica de Universidades*. Boletín Oficial de las Cortes Generales. Número 45-13, 26 de diciembre de 2001.

[7] *Luvit (página de inicio de la web de Luvit en inglés)*. Luvit. http://www.luvit.com /P00.m4n?language=en

[8] Macías, J. *Apéndice del curso Luvit*. Dpto. Ciencias de la Computación, Universidad de Alcalá, 2002. http://www.cc.uah.es/jmacias /jenui2003/apendice.pdf

[9] McKeachie, W. *Teaching tips*. Houghton Mifflin, 1999.

[10] *¿Qué es ADA-Madrid?*. Comunidad de Madrid. http://adamadrid.uc3m.es /campusvirtual/Idioma1 /proyectoadamadrid.htm

[11] *Presentación de Campus Extens*. Universidad de las Islas Baleares. http://campusextens.uib.es:2200/portal/pages /cast/ques.htm

[12] Rodríguez, J. *Gestión Automática de entrega de Prácticas vía web*. Actas de las VIII Jornadas de Enseñanza Universitaria de la Informática (JENUI, 2002).

[13] Ruipérez, G. *Las plataformas de gestión de aprendizaje (PGA)*. UNED, 2002. http://www.uned.es/euva/euvaflash/contenidos /articuloruiperez.htm

Desarrollo de actividades en grupos coordinados sobre el modelado y simulación del proceso de transmisión de datos

José Oliver, Alberto Bonastre, José L. Poza
Dpto. de Informática de Sistemas y Computadoras
Universidad Politécnica de Valencia 46022 Valencia
e-mail: {joliver, abonastre, jopolu}@disca.upv.es

Resumen

En este artículo se presenta la realización de actividades en grupo como metodología docente, y los resultados obtenidos dentro del marco de una asignatura sobre sistemas de transmisión de datos.

Durante el desarrollo de estas actividades se simula el proceso de transmisión de datos punto a punto a nivel físico. Cada grupo se centra en realizar el modelado de un elemento del sistema, que puede estar a nivel de codificación de datos, de modulación o de medio físico. Para realizar las actividades, los grupos deben coordinarse entre ellos, tanto para resolver problemas comunes (comunicación horizontal), como para definir claramente el interfaz para el paso de información que circula entre niveles contiguos (comunicación vertical). Por último, se plantea una serie de itinerarios en los que varios grupos de alumnos se agrupan formando grupos mayores y realizan una prueba global de todo el sistema.

1. Introducción

La visión clásica del proceso enseñanza-aprendizaje se basa en clases magistrales apoyadas por el enunciado y resolución de ejercicios en clase. Este planteamiento ofrece poca interacción profesor-alumno e invita escasamente a la participación del alumnado, que suele adoptar una posición de comodidad y paulatina pasividad, lo que resulta en un menor índice de éxito en el aprendizaje.

En 1988, la Universidad Politécnica de Valencia puso en marcha el Plan de Innovación Educativa (PIE). En aquel momento representaba una propuesta atrevida y avanzada encaminada a incentivar las mejoras del sistema enseñanza-aprendizaje en la docencia. Este nuevo plan y la reforma de los planes de estudio de Informática en 1996 aumentaron considerablemente el número de sesiones de docencia en laboratorio, donde el alumno resuelve problemas más reales y maneja instrumentación relacionada con la materia.

Doce años después, la Universidad Politécnica de Valencia propone un nuevo proyecto que recoge el testigo del antiguo PIE, adaptándose a las nuevas necesidades de la Universidad moderna. Este plan se denomina Proyecto EUROPA (Una Enseñanza ORientada al APrendizaje) [4] [5] y toma del PIE parte de sus objetivos, ampliándolos en algunos casos y complementándolos con nuevas propuestas en otros. En concreto se propone la inclusión de técnicas de aprendizaje innovadoras, como es el uso de seminarios en clase, con debates, mesas redondas y foros, que facilitan el aprendizaje mediante la participación interactiva del alumnado, y otras técnicas como las actividades, en las que los alumnos forman grupos de trabajo para realizar diversos ejercicios.

En este artículo presentamos las actividades propuestas para la asignatura optativa Sistemas de Transmisión de Datos, y el resultado obtenido por parte de los alumnos. Siguiendo el espíritu del proyecto EUROPA, nuestro objetivo no ha sido únicamente el desarrollo de habilidades en el marco de la temática de la asignatura, sino que otras capacidades no curriculares, como el trabajo en grupo y la capacidad de coordinación y liderazgo, también han sido potenciadas.

2. Sistemas de Transmisión de Datos

La asignatura Sistemas de Transmisión de Datos (STD) [1] se centra en los primeros niveles del modelo ISO/OSI (nivel físico y enlace de datos),

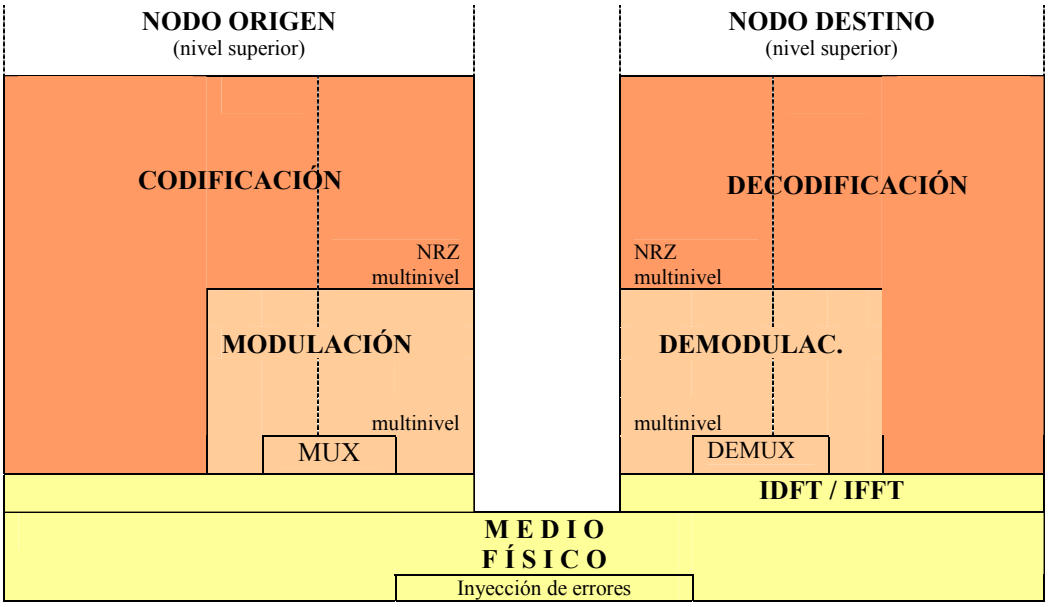


Figura 1. Proceso de comunicación punto a punto a modelar

profundizando en las técnicas de transmisión existentes.

Es una asignatura optativa, correspondiente a los planes de estudios de 1.996, para la obtención de los títulos de Ingeniero en Informática (5º curso, con 6 créditos) e Ingeniero Técnico en Informática de Sistemas /de Gestión (3º curso, con 4.5 créditos).

La asignatura se desarrolla en dos bloques temáticos diferenciados, cuyos temas son los expuestos a continuación.

Bloque I. Fundamentos de Transmisión de Datos.
1. Sistemas Teleinformáticos
2. Medios Físicos de Transmisión
3. Señal. Fundamentos teóricos
4. Modems
5. Transmisión serie

Bloque II. Sistemas de transmisión
6. ADSL-Modem cable
7. Tarjetas de Red. Ethernet
8. Transmisiones inalámbricas

La división en dos bloques se realiza para separar los fundamentos teóricos de las principales aplicaciones reales. Esta división proporciona dos visiones de los sistemas de transmisión de datos, una teórica donde se adquiere la base para comprender la segunda visión, la práctica y real.

3. Actividades propuestas

Las actividades consisten en la realización obligatoria por parte de grupos de alumnos de trabajos teórico/prácticos relativos a la asignatura. Dichos trabajos son tutorizados por el profesor de prácticas, y expuestos durante las últimas sesiones de teoría.

Los contenidos de dichos trabajos tienen una parte teórica (investigación bibliográfica) y otra práctica (implementación, simulación y evaluación de distintos componentes del nivel físico), y es necesaria la interacción entre los grupos tanto en su desarrollo como para la comprobación del funcionamiento adecuado de todo el sistema.

3.1. Coordinación de los grupos

Cada grupo de actividades está formado por tres componentes. En un grupo, cada uno de los

integrantes debe asumir una de las siguientes responsabilidades de coordinación:

- Responsable de coordinación interna del grupo. Se encarga de planificar los objetivos según etapas, y de que estos plazos se cumplan en la medida de lo posible.
- Responsable de coordinación externa. Es la persona responsable de comunicación con los otros grupos, tanto para resolver dudas y necesidades comunes, como para realizar las pruebas del sistema final.
- Responsable de coordinación con el profesor. Su labor es resolver las dudas que tienen, tanto los componentes del grupo hacia el profesor, como el profesor hacia los alumnos, manteniendo en todo momento una realimentación para que el profesor conozca la marcha de las actividades y los principales problemas observados.

Las responsabilidades que se plantean no son más que *roles* o papeles que se invita al alumno a interpretar. El objetivo es acercar el funcionamiento de las actividades al trabajo en grupo y la coordinación entre grupos que se exige en el mundo laboral.

3.2. Temática de los trabajos

Las actividades se centran en el área del temario situada dentro del Bloque I de la asignatura, por lo que una vez estudiados los primeros temas, el alumno ya se encuentra preparado para desarrollar las actividades. La segunda parte de la asignatura se cubrirá por medio de seminarios y estudios de caso. La distribución de créditos de las nuevas metodologías introducidas en este artículo, comparadas con las clásicas sesiones de teoría y prácticas, puede consultarse en [4]. En concreto se pretende realizar un entorno de simulación de los sistemas de transmisión punto a punto a nivel físico. Los trabajos que se realizan son parte de un entorno global, aunque se centran en un aspecto concreto del sistema.

La parte teórico de las actividades consiste en que el alumno se documente sobre el componente a implementar, sepa en qué consiste, su posible construcción hardware, cómo se puede simular su comportamiento, etc. La parte práctica del trabajo consiste en la implementación de código que, a partir de la entrada de un archivo de datos, los transforme de manera adecuada, de forma que según el nivel al que se esté trabajando, los codifique, module o simule su transmisión por un medio físico, volcando el resultado en otro archivo de datos.

El esquema general de la comunicación punto a punto que se pretende modelar es el indicado en la figura 1, en el que los datos se transmiten de un nodo origen a otro destino. Estos datos se inyectan en la red codificados, y posiblemente modulados. Otro aspecto que se pretende que el sistema pueda manejar es la multiplexación de datos. Para esto se deberá tener en cuenta que varias fuentes de datos, modulados a distintas frecuencias, se puedan transmitir un mismo medio físico mediante multiplexación en frecuencia.

Posteriormente, los datos procesados por el nodo origen, una vez codificados y modulados, se inyectan físicamente en la red. Para caracterizar el comportamiento del medio físico es necesario tener una representación espectral de los datos, para lo que se utiliza la transformada discreta de Fourier. A parte de los típicos efectos introducidos por el medio de transmisión (atenuación, distorsión por retardo de grupo, etc) se pueden inyectar errores en la red; algunos afectarán a su comportamiento en frecuencia (por ejemplo, un típico ruido introducido por la red eléctrica funcionando a 50hz) y otros serán puntuales en el tiempo (ruidos impulsivos, posibles ruidos blancos, etc.). Por último, la información tiene que ser procesada correctamente para que se recupere en el nodo destino, por lo que los alumnos también deberán implementar los procesos inversos de demultiplexación, demodulación y decodificación.

En la figura 2 se puede observar el conjunto de trabajos que se proponen, situados cada uno en su nivel correspondiente. Esta figura no es más que un desglose de la figura 1, detallando los posibles trabajos que se puede implementar en cada nivel.

3.3. Implementación de las actividades

A la hora de poder coordinar unas actividades de estas características resulta fundamental especificar el formato de los datos que las aplicaciones generan y reciben. En nuestras actividades utilizamos un formato básico muy

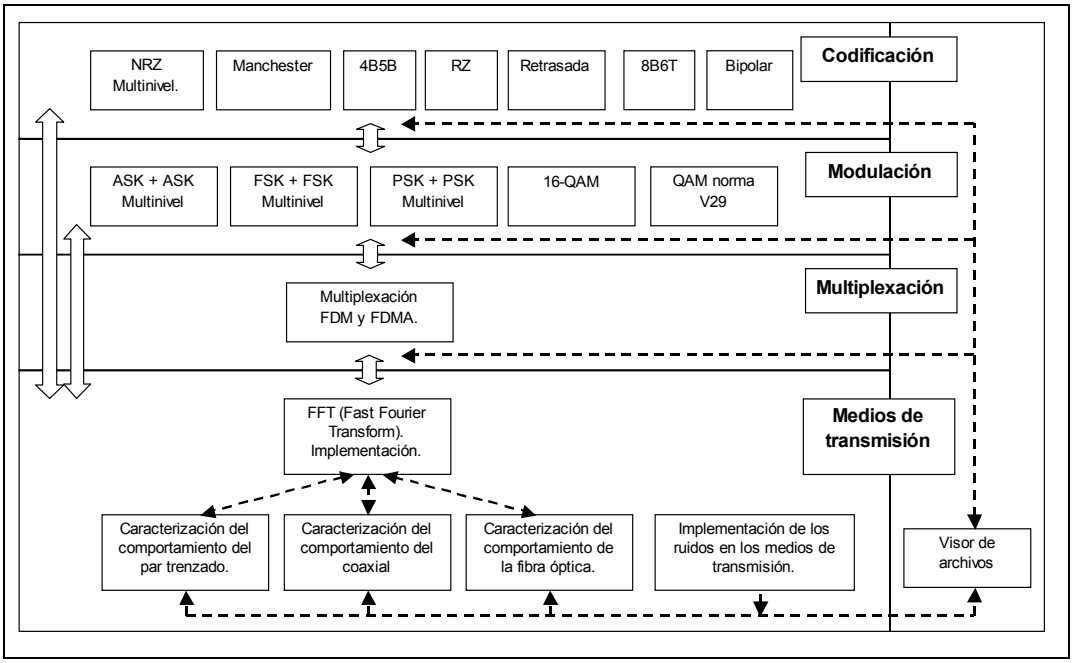


Figura 2. Conjunto de actividades

simple. Representamos una señal como un conjunto de niveles de tensión resultantes de muestrear esa señal en cada instante de muestreo. El resto de parámetros específicos de cada nivel se introducen por teclado cuando se ejecuta dicho módulo.

Un fichero que representa a una señal tiene como primera componente la frecuencia de muestreo usada, después se indica el número de muestras obtenidas y por último se encuentra el vector de muestras, representada cada una de ellas por un valor en coma flotante en simple precisión. Con este sencillo formato se puede representar, a partir de los valores de tensión en una serie de instantes discretos o muestras de la señal, el estado aproximado de una señal en todo momento. A su vez, el hecho de tratar la señal de manera discreta (o discretizada) nos permite un procesamiento digital que simplifica su análisis y modelado.

Para la realización de las actividades los alumnos no necesitan partir de cero, ya que se les ofrece una serie de esqueletos de programa para cada nivel, que implementan tareas básicas como la lectura/escritura de ficheros de muestras, la petición de datos al usuario y la definición básica de cabeceras e interfaces.

Además, para facilitar la comprensión por parte del alumno de todo el sistema que se pretende modelar, y permitir encuadrar bien el trabajo que se está desarrollando, se les facilita un ejemplo compilado de un módulo por cada nivel, de forma que ellos pueden hacer pruebas con un sistema que se encuentra ya resuelto. Por otra parte también se dispone de un visor de datos que permite leer varios ficheros de muestras y mostrarlos en pantalla de manera simultánea, con lo que se pueden observar las transformaciones que va sufriendo la señal en cada uno de los módulos.

4. Comunicación entre grupos

Cuando el alumno finalice su formación deberá haber desarrollado ciertas habilidades de trabajo en grupo, comunicación y coordinación, así como la capacidad de asumir su *rol* dentro del grupo de trabajo de la empresa. Por ello, se ha planteado dar un paso más en las actividades por medio de la

coordinación dentro del propio grupo e intergrupal.

La novedad de incluir una acción formativa de trabajo en grupo no es única ni innovadora, sin embargo sí que se ha considerado como novedad el hecho de que estos grupos deban comunicarse entre ellos, tanto a nivel vertical como a nivel horizontal [3]:

- *Comunicación vertical.* En el proyecto de actividades expuesto anteriormente, los grupos escogen un área en la que trabajar: codificación, modulación o medios de transmisión. Entre estos niveles o capas de las actividades deberá existir una comunicación para consensuar el formato de los archivos y paso de datos.
- *Comunicación horizontal* Gran parte del código que deba desarrollar cada grupo, así como de la documentación, será común a todos los grupos que trabajen dentro de la misma capa. Por ello, entre los grupos deberá existir una comunicación para no repetir trabajo y repartirse labores comunes.

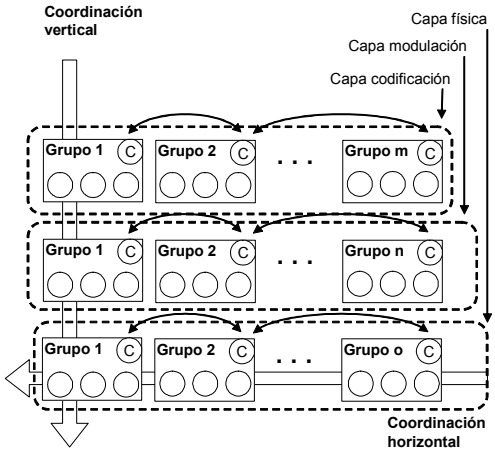


Figura 3. Relaciones de comunicación entre los diversos grupos de las actividades.

Con este sistema se pretende estudiar la interactuación entre grupos cuando el trabajo de unos depende de otros [2]. De esta forma la comunicación entre coordinadores, tal y como se ve en la figura 3, será muy importante. Dicha comunicación se llevará a cabo durante reuniones periódicas.

5. Resultados

La experiencia propuesta se ha llevado a cabo durante dos cursos diferentes. En un primer año tan sólo se desarrollaron los trabajos propuestos, sin llegar a hacer una prueba global de todo el sistema. De esta manera, sólo hubo ocasión de realizar coordinación horizontal, mediante el apoyo entre grupos que trabajaban en proyectos similares pertenecientes al mismo nivel. En un segundo año se pudo poner en marcha unas sesiones adicionales para realizar labores de coordinación vertical, intentando integrar todos los trabajos de manera que el objetivo final de colaboración entre proyectos a distintos niveles quedara totalmente cubierto.

5.1. Comunicación horizontal: resultados por niveles

Al iniciar el desarrollo del trabajo, cada grupo se centra en documentarse e implementar una actividad concreta. En esta primera fase, la comunicación predominante será la horizontal, ya que distintos grupos que trabajan al mismo nivel se encuentran con problemas similares, que podrán resolver de manera coordinada. Aun así la comunicación vertical seguirá presente, ya que hay ciertos parámetros del interfaz de la aplicación que dependerán de cómo estén implementando su actividad los grupos de otros niveles.

Los grupos que estén trabajando en la capa de codificación son los que inician el proceso de la simulación. Deben tomar los datos a partir de una secuencia de bits y realizar su representación según la codificación a aplicar. Para esto se hace un muestreo de la señal resultante a partir de cada uno de los valores binarios de la fuente de datos. Sabemos que la duración de un bit dependerá directamente de la velocidad de transmisión. Por otra parte, el número de muestras por bits, una vez establecida la duración del bit, dependerá del periodo de muestreo, determinado por la frecuencia de muestreo. Por tanto, el problema de obtener una secuencia de muestras a partir de un flujo de bits es común para todos los grupos que trabajen a nivel de codificación, y podrá resolverse de manera coordinada. Posteriormente, bastará con realizar una asignación distinta de los

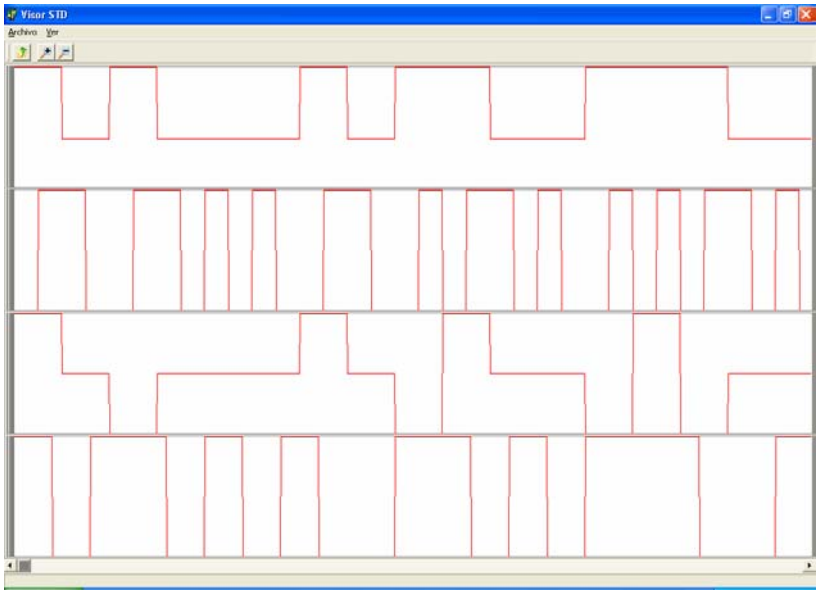


Figura 4. Resultado de distintas actividades en la capa de codificación.

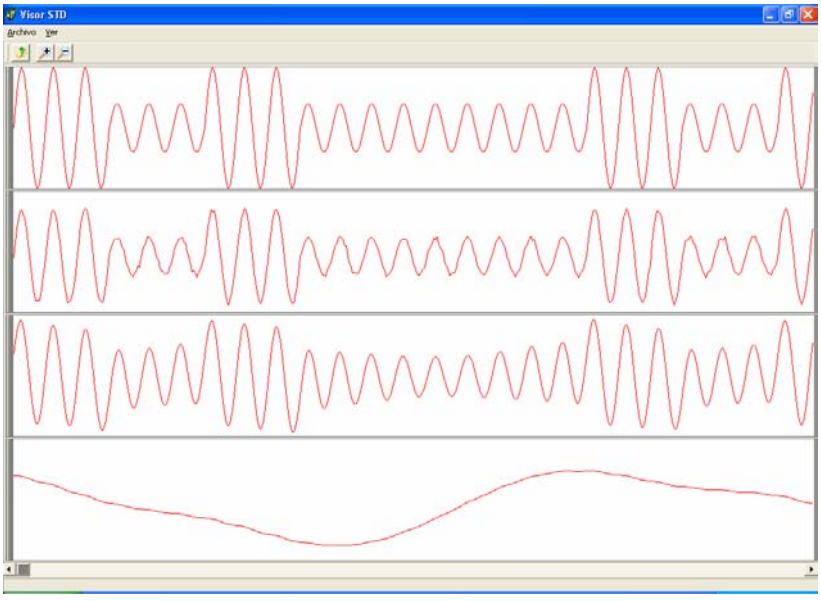


Figura 5. Resultado de las actividades en la capa de medio físico de transmisión.

niveles de tensión en función del valor del bit y el tipo de codificación implementada.

En la figura 4 tenemos el resultado de codificar el mismo flujo de bits utilizando distintas codificaciones implementadas por los grupos de actividades. La primera de todas, situada en la parte superior de la pantalla del visor, se corresponde con una sencilla NRZ, de

manera que el grupo encargado de implementar este tipo de codificación simplemente debía asignar un nivel alto de tensión cuando el bit a codificar fuera un uno, y un nivel bajo en caso contrario. La siguiente codificación es una Manchester. En este caso, para codificar un bit a uno se debe especificar un flanco de subida, codificando la primera mitad del intervalo del bit como nivel bajo y la segunda como nivel alto. La tercera codificación se corresponde con una bipolar en la que un cero binario se implementa con un nivel de tensión de cero voltios, y los bits a uno iteran sucesivamente de un nivel de tensión determinado positivo +V a otro –V, balanceando de esta forma el nivel de continua. Por último se encuentra una codificación del tipo 4B5B, en la que la asignación de valores a las muestras ya no se realiza bit a bit de manera directa, sino que existe una asignación previa de 5 bits por cada medio byte de la fuente original. Una vez realizada esta conversión, el tipo de codificación será similar a la NRZ, pero utilizando los valores indicados en la tabla de conversión 4B5B.

Una vez resuelto el problema común de asignar valores muestras en función de los bits de entrada, implementar una u otra codificación resulta sencilla, de manera que los alumnos han aprendido como trabajar en grupo para resolver un problema les resulta más productivo que enfrentarse al mismo problema por separado.

5.2. Comunicación vertical: pruebas de itinerarios

Tal y como está planteada hasta ahora la experiencia resulta enriquecedora, ya que promueve el trabajo en equipo y entre equipos, pero sin embargo aun puede completarse un poco más. Con este objetivo, en un segundo año, se incluyó la necesidad de realizar pruebas conjuntas entre equipos que trabajan a distintos niveles, de manera que se potenciase aun más la coordinación entre grupos de distintas capas.

De esta forma, se plantean una serie de itinerarios que agrupan una actividad de cada nivel, de manera que, una vez terminado, depurado y entregado el trabajo a realizar por un grupo, se llevan a cabo una serie de pruebas conjuntas en las que cada grupo aporta su módulo. Así se observa cómo la señal se transforma en función del tipo de medio utilizado y otros factores, como la longitud del cable, su categoría, o el nivel de ruido térmico o impulsivo introducido en el medio. La figura 5 muestra un ejemplo de cómo la señal modulada (arriba) varía al sufrir un ligero ruido térmico (segunda señal), se atenúa ligeramente al utilizar un cable de categoría cinco de 500 metros (tercera señal), o casi por completo al utilizar un cable de inferior categoría y una distancia de 5000 metros. Hay que resaltar que la última señal ha tenido que ser ampliada y reescalada, ya que la mayor parte de frecuencias se encuentran atenuadas.

En este tipo de pruebas conjuntas resulta muy interesante una actividad concreta como es la multiplexación en frecuencia. En la figura 6 encontramos un ejemplo de itinerario en el que se ha aplicado multiplexación de dos fuentes de datos distintas, que han sido codificadas con NRZ y moduladas aplicando ASK OOK. Estas dos señales, que se encuentran representadas en la parte superior del visor, se multiplexan en un mismo medio siendo el espectro resultante el que se muestra en la tercera posición de esta figura. Una vez transmitida la señal por el medio se filtra en el nodo destino utilizando dos filtros paso-banda, de manera que se puede volver a recuperar ambas fuentes por separado. Algunas componentes frecuenciales se mezclan de forma inevitable, y por tanto las señales variarán ligeramente respecto a su representación en el nodo original, como se observa en las señales situadas en la parte inferior del visor.

Por otra parte, se pueden realizar pruebas de itinerarios más reales, como por ejemplo simular la transmisión de datos por un canal telefónico utilizando la norma v.21. Para realizar esta prueba disponemos de todos los ingredientes necesarios. A nivel de codificación y modulación, usamos un NRZ a 300 bps con modulación FSK, utilizando portadoras de 1080 hz con desviaciones de 100 hz a ambos extremos para el canal inferior, y otra portadora de 1750 hz con la misma desviación para el superior. Una vez obtenidas ambas señales de subida y bajada de datos, se utiliza el módulo de multiplexación para combinar ambas fuentes que, una vez multiplexadas, se pasan al módulo de medio físico de transmisión que modela el canal telefónico. Este ejemplo es mucho más próximo al mundo real y por esto el alumno se encuentra más motivado. Además, se les plantea ejemplos de empresas que, para realizar el diseño de sus

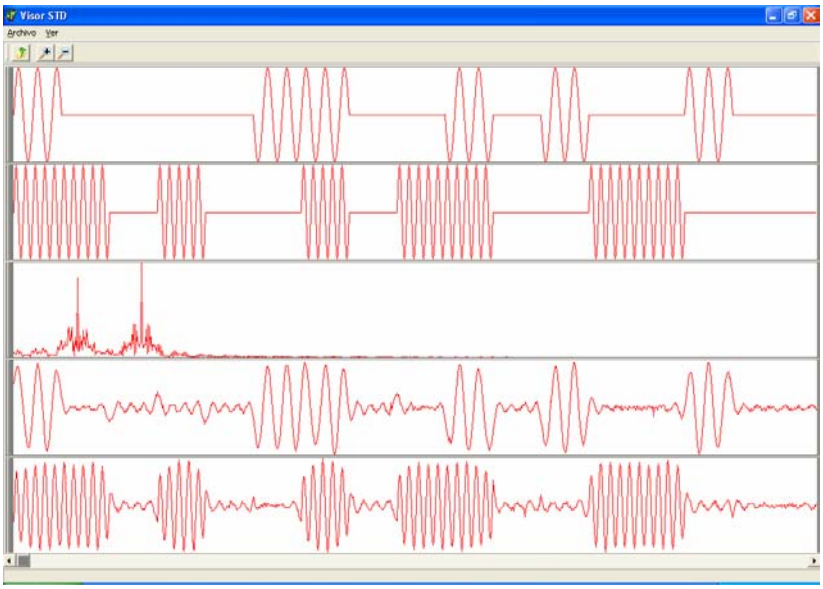


Figura 6. Resultado de un itinerario con multiplexación en frecuencia.

tarjetas de red, hacen estudios de comportamiento del medio similares a los que ellos llevan a cabo, aunque con un nivel de complejidad mayor.

Para realizar las pruebas de itinerarios se dedicaron dos sesiones prácticas de dos horas. En ellas los alumnos ya no sólo cooperaron entre los miembros de un mismo grupo, sino que realizaron las pruebas con personas que estaban trabajando con otros grupos de otros niveles, con lo que adquirieron una visión más global, adoptando distintos puntos de vista, y se favoreció la percepción de la importancia de la coordinación vertical para lograr el correcto funcionamiento del trabajo que se estaba realizando.

6. Conclusiones y trabajo futuro

En este artículo se ha presentado la aplicación de actividades como una nueva metodología docente en una asignatura sobre sistemas de transmisión de datos. Los resultados obtenidos por parte de los alumnos han sido satisfactorios, tanto en el desarrollo de los trabajos en sí, como en la mejora de habilidades para el trabajo en grupo.

Además, mediante una encuesta que se ha pasado a los alumnos, se refleja como el índice de satisfacción por parte de los alumnos es bastante alto, obteniendo una calificación final de 8,1, medio punto por encima del resultado obtenido al preguntarles por las metodologías docentes más clásicas, como son las sesiones de teoría o las prácticas en el laboratorio.

Como trabajo futuro, y a partir de las experiencias obtenidas durante este año, pretendemos potenciar desde el principio la coordinación entre grupos del mismo itinerario.

Referencias

[1] Alberto Bonastre Pina, Félix Buendía García, Manuel Pérez Malumbres. *Equipos y Sistemas de Transmisión de Datos*. SPUPV. 1994

[2] Contreras J.M. *Cómo trabajar en grupo*. Ed. San Pablo. 1997.

[3] Musitu, G. *Psicología de la comunicación*. Nau Llibres. 1987.

[4] José Luis Poza, Alberto Bonastre, Jose Oliver. *Aplicación de las directivas EUROPA en la asignatura Sistemas de Transmisión de Datos*. VIII Jornadas de Enseñanza Universitaria de la Informática, 2001

[5] Vicerrectorado de coordinación académica y alumnado. *Una enseñanza orientada al aprendizaje: proyecto EUROPA*. UPV. 2001

Demostraciones

Desarrollo de un simulador de programación del microprocesador Intel 8085

Manuel Rodríguez Álvarez, Ángel Manuel Gómez García,
Pedro Mesas García, José Ignacio Ruiz Núñez, Alberto Prieto.
Departamento de Arquitectura y Tecnología de Computadores.
Escuela Técnica Superior de Ingeniería Informática. Universidad de Granada
C/ Daniel Saucedo Aranda s/n. 18071 Granada
e-mail: mrodriguez@atc.ugr.es ; manolo@goliat.ugr.es

Resumen

El objeto de este trabajo es la concepción, desarrollo y realización de un simulador de microprocesador 8085 diseñado por Intel, al objeto de ser utilizado con fines docentes en la Universidad de Granada en diferentes asignaturas de las titulaciones de Ingeniero en Informática o Ingeniero Técnico en Informática de Gestión y Sistemas en la Escuela Técnica Superior de Ingeniería Informática o bien de Ingeniería Electrónica, Ingeniería Industrial e Ingeniería de Telecomunicaciones. La idea principal del mismo es disponer de un entorno amigable de simulación que permita al estudiante adquirir un profundo conocimiento de la programación de dicho microprocesador. Obviamente el uso de dicho simulador es libre y gratuito al objeto de facilitar su uso y difusión.

En este trabajo, después de una breve introducción pasaremos a describir la funcionalidad del simulador y su aspecto.

1. Introducción

Un simulador de un microprocesador es un programa que se comporta de igual forma que el microprocesador simulado.

El simulador, como cualquier otro programa, se apoya en el microprocesador de la maquina en la que se ejecuta, el cual no tiene porque parecerse en nada al microprocesador simulado y siendo en el caso que nos ocupa, el microprocesador de la máquina donde se ejecuta mucho más avanzado que el simulado. En nuestro caso, el simulador se comporta como un traductor que comprende un programa diseñado para 8085 y que genera las instrucciones necesarias para que el procesador sobre el que se ejecuta se comporte de forma similar a un 8085. La simulación trata de proveer la misma apariencia externa que el elemento que se simula. En cualquier simulación existen una serie de aspectos sobre los que nos centramos y otra serie de aspectos que obviamos. La simulación puede ser más amplia o menos según en la cantidad de aspectos en los que centre, y más o menos precisa según cuanto profundice en ellos. Obviamente una simulación más amplia y detallada, implicará una mayor complejidad del simulador.

Podemos centrar la simulación del 8085 a muchos niveles. Desde un nivel operativo, en el que el microprocesador realiza una determinada tarea, hasta un nivel electrónico, en el que se simula la evolución de todos los componentes electrónicos del procesador.

Dado que el objetivo de este trabajo es obtener un simulador didáctico para aprender a programar un 8085, sólo nos interesarán ciertos aspectos de la simulación, obviando el resto y centrando el simulador en la programación del 8085.

2. Funciones del simulador

El simulador del 8085 que se ha desarrollado, ha sido a nivel de programación. Por tanto, las unidades que se han simulado son las instrucciones, las cuales afectan a los registros, los bits de estado, la memoria y los puertos. También se ha incluido la opción de que el simulador disponga de dispositivos externos de salida (pantalla de texto y gráfica, panel de leds y visualizadores de 7 y 15 segmentos) y de entrada (panel de interruptores y un teclado externo). Esto lleva a que también se ha simulado cómo las instrucciones afectan a los dispositivos externos.

El desarrollo del trabajo comienza con el estudio de las diferentes posibilidades en cuanto al simulador, el cual se ha desarrollado para que se

pueda trabajar bajo un entorno de Windows 98 o Windows 2000 en un computador Personal Compatible.

Para llevar a cabo la interfaz con el usuario se ha empleado un entorno de programación que ha permitido el desarrollo de una aplicación en un entorno gráfico de ventanas con la posibilidad de utilizar menús, botones, barras deslizantes, etc.. Para tal fin se ha utilizado un lenguaje de tipo visual el cual permite diseñar un entorno agradable de interfaz con el usuario. Dicha interfaz presenta, una serie de ventanas donde se muestra, entre otras funciones, el contenido de la zona de memoria de datos, memoria de programa, memoria de pila, los registros internos del microprocesador y los bits de estado.

Asimismo, se permite trabajar en tiempo real con el simulador, introduciendo datos y programas directamente desde el teclado, o bien leyendo los mismos desde un fichero previamente editado. En este último caso, el simulador permite la utilización de un editor de textos, que ha sido diseñado específicamente para el mismo, el cual contempla además la posibilidad de edición de programas en un sencillo lenguaje ensamblador para 8085. Asimismo, se pueden grabar los resultados de una simulación en un fichero.

Por último, se desarrolló en hipertexto un manual de ayuda en línea, integrado en el propio simulador, utilizable dentro de él o bien independientemente.

3. Aspecto del simulador

En la Figura 1 se muestra el aspecto que presenta la interfaz con el usuario del simulador de 8085. En ella pueden distinguirse las siguientes zonas:

3.1. Registros de la CPU

El 8085 cuenta con varios registros internos, (situados en la parte superior derecha del simulador) B, C, D, E, H y L, el A (acumulador) y el F (bits de estado), de 8 bits cada uno, y los registros SP (puntero de pila) y PC (contador de programa), de 16 bits. Los registros AF, BC, DE y HL, se presentan por parejas, pues muchas de las instrucciones del 8085 usan estas agrupaciones de registros. Situando el puntero del ratón sobre los bits de cada registro es posible cambiar su valor.

3.2. Bits de estado

Existen para el 8085 cinco bits de estado, signo (S), cero (Z), acarreo auxiliar (Ac), paridad (P) y acarreo principal (C). Aparecen en forma de pequeños leds en la parte central del simulador. Si un led está encendido indica que el bit está activo y si está apagado, el bit estará inactivo. Situando el puntero del ratón sobre los leds es posible cambiar su estado.

3.3. Puertos de Entrada y de Salida

El 8085 cuenta con 256 puertos de entrada y 256 puertos de salida de 8 bits cada uno que se sitúan en la parte inferior central del simulador. Todos los puertos aparecen en una misma lista, en la que en la columna izquierda se indica el número de puerto correspondiente.

3.4. Memoria de instrucciones

El 8085 dispone de una memoria de 65536 bytes. En esta memoria se cargan las instrucciones de los programas y los datos. Las instrucciones ocupan un byte, dos bytes y tres bytes. En la zona de memoria de instrucciones (parte izquierda del simulador) aparecen tres columnas, que contienen la siguiente información:

- *Dirección:* indica la posición de memoria en hexadecimal y puede ir entre 0000_h y $FFFF_h$.
- *Nemotécnico:* indica el nemotécnico de la instrucción ubicada en esa dirección de memoria.
- *Código:* contiene el código de la instrucción en hexadecimal y puede ir entre $00)_h$ y $FF)_h$.

También se puede apreciar que dos de las filas están iluminadas con distinto color que las demás:

- En color rojo se señala la posición del contador de programa (PC).
- En color azul se indica una posición seleccionada para ser editada por el usuario.

Pulsando con el botón derecho del ratón sobre la cabecera de la ventana de instrucciones aparecen tres opciones:

- *Ir a dirección de PC:* se salta a la posición en la que se encuentra situado el contador de programa.

- *Ir a dirección de comienzo de programa:* se salta a la dirección de comienzo del programa.
- *Ir a dirección...:* Aparece una ventana en la que puede introducir una dirección de la memoria de instrucciones.

3.5. Memoria de datos y memoria de pila

Estas dos listas que aparecen en la parte inferior derecha del simulador tienen una estrecha relación con la memoria de instrucciones. En realidad todas ellas contienen los mismos datos. La casilla en color azul corresponde a la selección de la posición actual. Para cambiar cualquier posición de las memorias o situarse en una posición determinada, basta con picar sobre ella con el cursor del ratón. En la memoria de pila siempre aparece una posición resaltada con el color verde indicando la posición del puntero de pila. Si se pulsa con el botón derecho del ratón sobre la cabecera de la memoria de datos o pila, aparece un menú emergente con 2 opciones:

- *Ir a dirección SP:* Esta opción no es igual para la memoria de pila y para la memoria de datos.
  - En la memoria de pila, muestra la posición a la que apunta el puntero de pila.
  - En la memoria de datos, esta opción permite ir a la primera dirección donde se encuentran los datos del programa.
- *Ir a dirección...:* Tanto para el caso de la memoria de datos como para la memoria de pila, se puede introducir una posición de memoria.

3.6. Control de ejecución

Situado en la parte central del simulador, contiene 4 botones desde los que es posible controlar la ejecución de sus programas. Cada botón realiza una tarea distinta:

- *Botón Step:* Ejecuta la siguiente instrucción paso a paso y se detiene.
- *Botón Over:* Tiene la misma función que el botón anterior salvo que, cuando se llega a una instrucción de llamada a subrutina, el simulador ejecuta todas las instrucciones pertenecientes a la subrutina, dejando el contador de programa en la siguiente instrucción a la instrucción de llamada.
- *Botón Run:* Ejecuta en modo continuo.
- *Botón Stop:* Para la ejecución en modo continuo.

3.7. Panel de interrupciones

El 8085 permite detener la ejecución normal de un programa mediante una serie de interrupciones. Los parámetros que controlan las interrupciones se han representado por casillas (situadas en la parte inferior izquierda del simulador) que pueden tomar dos posibles valores, ☑ (activa) o ☐ (inactiva). Existen 4 tipos de interrupciones, TRAP, RST 7.5, RST 6.5 y RST 5.5. Además existe un elemento más, llamado INTR, el cual le indica si las interrupciones están permitidas o no. Cada interrupción tiene asociadas dos casillas que tienen el siguiente significado:

- La casilla de la izquierda indica si una interrupción está permitida o no. La interrupción TRAP siempre está permitida.
- La casilla de la derecha permite realizar las peticiones de interrupción. En el momento se que quiera realizar una petición de interrupción basta con pulsar con el ratón en la casilla correspondiente.

3.8. Entrada / Salida serie

En la esquina inferior izquierda aparece un pequeño panel con dos casillas, que le permitirá controlar la entrada serie (SID) y salida serie (SOD).

- Casilla SID: Si está activa, cuando se ejecute la instrucción RIM, se cargará en el bit 7 del acumulador el valor 1. En caso contrario, este bit contendrá el valor 0.
- Casilla SOD: Cuando se ejecute la instrucción SIM, si los bits 6 y 7 del acumulador están puestos al valor 1, la casilla se activará. En caso contrario la casilla permanecerá inactiva.

4. Conclusión

Los objetivos más relevantes que se han conseguido con el simulador de programación de microprocesador Intel 8085 han sido:

- Acercar al estudiante al 8085.
- Ayudar al profesorado a dar a conocer el 8085.
- Facilitar el aprendizaje del 8085.
- El programa trabaja bajo un entorno de Windows 9x o Windows 2000 en un computador Personal Compatible lo que facilita enormemente su difusión.

- La interfaz con el usuario se ha llevado a cabo en entorno de programación que permita el desarrollo de una aplicación en un entorno gráfico amigable entre el usuario y el simulador.
- La interfaz presenta una serie de ventanas donde se muestra el contenido de la zona de memoria de datos, memoria de programa, memoria de pila, los registros internos del microprocesador y los bits de estado.
- El simulador incorpora un fichero de ayuda en línea, así como un editor de textos integrado para editar, compilar y hacer funcionar los programas de simulación.

Referencias

[1] Alecop. *Monografía nº 20. Microprocesador 8085*. 1987.

[2] José María Angulo. *Microprocesadores y Microcontroladores 8085, MCS-51 y ST-6*. Paraninfo, 1996.

[3] Timothy Budd. *Introducción a la programación orientada a objetos*. Addison-Wesley Iberoamericana. 1994.

[4] Antonio Cañas. *Apuntes de la asignatura Estructura de los Computadores I y II*. Escuela Técnica Superior de Ingeniería Informática. Universidad de Granada, 1999.

[5] Francisco Charte. *C++ Builder 4*. Anaya Multimedia, 1999.

[6] C. H. Pappas, W. H. Murria. *Manual de Borland C++*. McGraw-Hill, 1993.

[7] Alberto Prieto, Antonio Lloris y Juan Carlos Torres. *Introducción a la Informática*. McGraw-Hill, 2001.

[8] Alberto Prieto, Julio Ortega, Antonio F. Díaz y Antonio Cañas. *Estructura y funcionamiento de microprocesadores*. 3ª Ed. Copistería La Gioconda. Granada, 1999.

[9] Manuel Torres. *Microprocesadores y Microcontroladores aplicados a la industria*. Paraninfo, 1991.

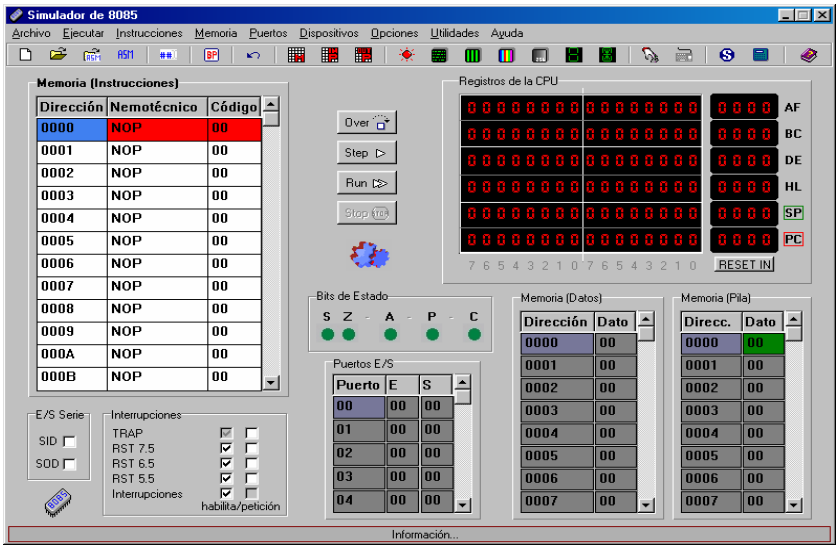


Figura 1. Aspecto inicial del Simulador de 8085.

Jlógica, un entorno de diseño de sistemas digitales

José A.Alvarez, V.G.Ruíz, J.F.Sanjuan, Javier Roca P.
Depto. Arquitectura de Computadores y Electrónica.
Universidad de Almería.
04120 Cañada de San Urbano.
jaberme@ual.es

Carlos A. Bermejo
Área de desarrollo – S. Informática.
Universidad de Almería.
04120 Cañada de San Urbano.
cbermejo@ual.es

Resumen

Jlógica es el resultado de una visión docente del lenguaje de programación Java. Con Jlógica se dispone de una herramienta de diseño y simulación de circuitos homogénea, intuitiva y sin restricciones, capaz de facilitar el aprendizaje en asignaturas tales como Estructura de Computadores, Tecnología de Computadores y Laboratorio de Arquitectura.

1. Introducción

Jlógica es una herramienta Java pensada para homogeneizar el entorno de trabajo en las prácticas de varias asignaturas, de esta manera módulos realizados para una determinada asignatura podrían ser reutilizables a medida que el alumno vaya adquiriendo conocimientos de mayor nivel con la consiguiente ventaja del conocimiento y experiencia práctica obtenida con el entorno de trabajo. Una de las principales ventajas de la herramienta es la independencia que otorga Java con respecto a la plataforma, así como la capacidad de integración con la red. La primera es importante ya que no restringe el sistema operativo usado en los equipos de los laboratorios. La segunda ventaja es que se hace extremadamente sencilla su utilización ya que una vez instalado en un servidor Web, el acceso puede realizarse tanto desde el laboratorio como desde cualquier punto conectado a la red.

Por otra parte, Jlógica dispone de un módulo generador de código Java a partir del diseño gráfico del circuito, el cual facilita el análisis del comportamiento del mismo mediante la modificación de parámetros básicos. Las asignaturas para las que se está desarrollando esta aplicación son Tecnología de Computadores y Estructura de Computadores.

En este artículo se detallan las características de utilización de esta herramienta. En el apartado 2 se describe la interfaz y en el apartado 3 su utilización como entorno de simulación de circuitos. En el apartado 4 se detalla la utilización de esta herramienta como generador de código a partir del diseño gráfico de un circuito.

Además de haber optado por Java para la implementación por las razones antes aducidas, portabilidad e integración en la red, se pueden argumentar razones secundarias tales como que Java dispone de librerías de clases (Swing [4] y Awt [2]) que facilitan el desarrollo de la interfaz además de sobrados componentes visuales libres[1] y reutilizables [1].

2. Descripción de la interfaz.

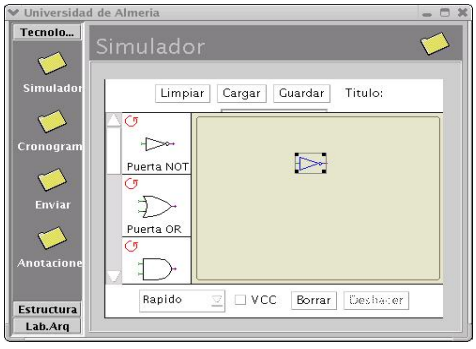


Figura 1. Aspecto de la interfaz Jlógica.

[1] Libre ha de entenderse que el propietario del software cede el componente para su uso de manera desinteresada.

Jlógica cuenta como punto de entrada con una interfaz muy similar a la interfaz ofrecida por Outlook para relacionar la aplicación con las asignaturas que harán uso de ella y separar adecuadamente la funcionalidad de la aplicación. El panel de la izquierda dispone las áreas:

- Tecnología Computadores.
- Estructura de Computadores.
- Laboratorio de Arquitectura.

Dentro de cada una de estas áreas se encuentra la funcionalidad requerida para las asignaturas, sea el caso mostrado en Fig.1 en el que para la asignatura de Tecnología se dispone de opciones como *Simulador* que inicia la aplicación, *Enviar* para mandar al profesor resultados, etcétera. Pero la parte importante de la aplicación no es la interfaz que la relaciona con las asignaturas sino el propio simulador (Fig.2).

Figura 2. Aspecto del simulador.

El simulador dispone de un juego de puertas lógicas básicas (AND, OR, NOT) y de una serie de piezas de utilidad como conectores de entrada/ salida y de empalme (Fig. 3), las cuales están dispuestas en la barra de puertas a la izquierda del área de trabajo Jlógica. Con estas puertas y piezas de utilidad el alumno puede comenzar a montar circuitos fácilmente. Además Jlógica cuenta con dos barras de herramientas (superior e inferior) que suman funcionalidad a la herramienta. La barra superior cuenta con funciones aplicables a todas las puertas del circuito, estas son:

- Limpiar: Elimina el diseño del circuito por completo.
- Cargar/ Guardar[2]: Exportación / importación del circuito a disco.
- Título: Nombra un diseño de circuito para usos posteriores.

En la barra inferior contamos con funciones aplicables a un subconjunto de puertas del circuito tales como:

- Borrar: Elimina las puertas seleccionadas así como sus conexiones con otras puertas.
- Deshacer: Restaura el diseño del circuito anterior a una modificación.
- Iconizar: Construye una nueva puerta con el diseño del circuito que esté en ese momento activo en el panel. Esta nueva puerta se añade con el nombre que se haya usado en la opción "Título" para que pueda ser usada como un componente en otros diseños.

Además de estas opciones aparecen otras dos que tienen que ver con la simulación en sí, estas son el desplegable en el que se puede elegir la velocidad con la que se verán la transición entre estados de una tabla de verdad y el checkbox[3] *VCC* que es el que inicia la simulación haciendo las veces de fuente de alimentación.

3. Áreas de aplicación

Se entiende por áreas de aplicación a las asignaturas que pueden, en principio, hacer uso del simulador para adaptar parte de sus lecciones prácticas. Entre ellas se encuentran las que a continuación se enumeran.

3.1. Tecnología de computadores.

En esta asignatura Jlógica le permite al alumno comprobar de forma fácil cómo los resultados obtenidos tras una minimización correcta de funciones coinciden con las versiones canónicas de los mismos, bien expresada en su forma de tabla lógica o circuito combinacional.

[2] El circuito se copia en un fichero binario al que se le añade un identificador para que el simulador sólo abra los ficheros que él ha generado y no cualquier otro tipo de fichero.

[3] Opción que se activa o desactiva marcando con un tick en el lugar habilitado.

La herramienta simula al panel-entrenador que con carácter general es utilizado por los alumnos en este tipo de prácticas, disponiendo de los elementos fundamentales integrados en el panel (interruptores, leds, conectores, etc.).

La disponibilidad de puertas lógicas, flip-flops u otro tipo de circuito lógico se obtiene a partir de la posible combinación de los elementos básicos (inversor, puerta AND y puerta OR) que se hayan integrados (Fig.2) con el desarrollo de la herramienta. De esta forma la tarea del alumno, ante el testeo de cualquier circuito propuesto, consiste en el ensamblaje apropiado de estos tres elementos hasta conseguir la función necesitada. Toda función diseñada anteriormente puede encapsularse como elemento de circuito y reutilizarse posteriormente en diseños más complejos, confeccionando de esta forma una librería tan extensa como se necesite. A continuación se describe un ejemplo.

La Fig.3 muestra cómo construir un circuito para implementar una determinada función.

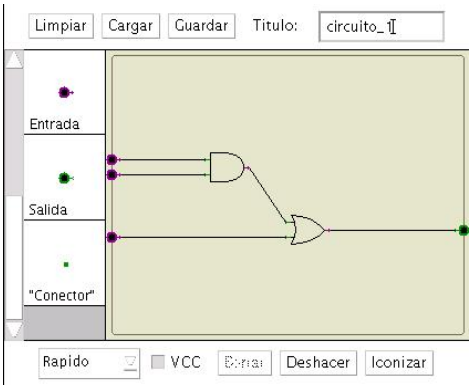


Figura 3. Diseño de circuito.

En Fig.4 se ve un ejemplo de cómo se encapsula para construir una puerta adicional llamada “circuito_1” con funcionalidad ampliada.

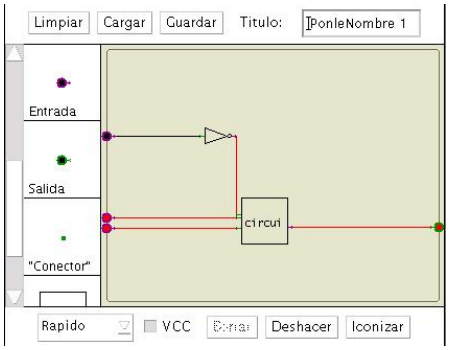


Figura 4. Encapsulación y reutilización.

En circuitos complejos donde el número de puertas sea elevado, existe la posibilidad de perder el diseño del circuito encapsulado que se está usando. La confusión es mayor si en lugar de un encapsulado se usan varios. Por tanto, Jlógica permite inspeccionar el contenido de los circuitos encapsulados sin más que seleccionar la pieza encapsulada y pulsar sobre el botón “Expandir[4] ”, tal y como se puede ver en la Fig. 5.

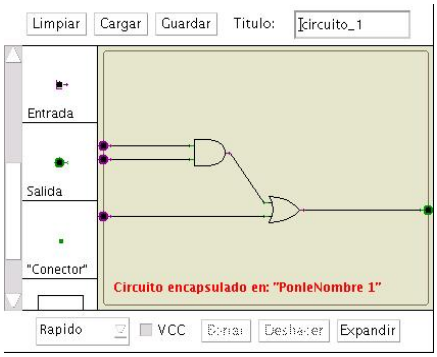


Figura 5. Contenido del encapsulado utilizado en el montaje.

3.2. Estructura de computadores.

El enfoque que se le otorga a esta asignatura en nuestro actual plan de estudios es el de iniciar al alumno en el conocimiento de los bloques fundamentales constitutivos de un computador.

[4] La opción expandir sólo se hace visible cuando existen componentes compuestos en el circuito.

El alumno al finalizar la asignatura ha debido adquirir el conocimiento de la metodología de diseño a nivel de registros así como la estructura y organización de una unidad central de proceso. Así pues, el aprendizaje realizado con módulos lógicos elementales en la asignatura Tecnología de Computadores son aplicados en esta asignatura para constituir bloques más complejos como ALU's o el diseño completo de una unidad central de proceso sencilla.

Las características de Jlógica explicadas anteriormente facilitan la consecución de estos objetivos sin más que sustituir el conjunto de piezas básicas usadas en la asignatura Tecnología de Computadores por un nuevo conjunto de piezas necesarias para esta asignatura como pueden ser una ALU, bancos de memoria, bancos de registros, multiplexores, etcétera.

4. Generación de código.

Uno de los aspectos prácticos tratados en Laboratorio de Arquitectura de Computadores es la simulación computacional de circuitos lógicos [3], mediante la que el alumno experimenta el testeo de bloques computacionales constitutivos de una arquitectura analizando el comportamiento ante distintos patrones de entrada y modificación de parámetros característicos.

Jlógica facilita el análisis anterior mediante la generación de código (Fig.6) a partir del diseño gráfico del bloque a estudiar. El modelo en esta versión se genera en código Java, si bien se pretende ampliar a otros lenguajes tal capacidad.

5. Conclusiones.

Jlógica goza de las capacidades propias del lenguaje Java como son la independencia de la plataforma lo que le confiere gran portabilidad y la hace una herramienta muy flexible y poco restrictiva, integrable en Internet. Jlógica se integra en páginas Web sin dificultad (applet). Otra ventaja de Jlógica es el carácter interdisciplinario que adquiere al poder utilizarse en diferentes asignaturas.

El desarrollo de Jlogica es el fundamento de una herramienta con la que se pretende en un futuro inmediato alcanzar niveles de complejidad de una arquitectura completa, abordando tanto el diseño del camino de datos como el de la unidad de control.

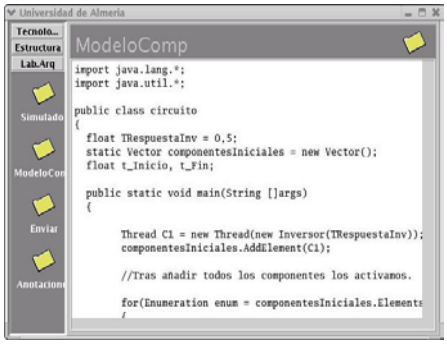


Figura 6. Ejemplo de generación de código para un inversor.

Referencias

[1] Duguay, Claude. Revista electrónica JavaPro. http://www.fawcette.com/archives/magazines/javapro

[2] Knudsen, Jonathan. Java 2D Graphics. First Edition. ISBN 1.56592-484-3. Ed. O'Reilly.

[3] Ruiz, V.G. *Diseño de Simuladores Digitales usando SDLC++*. **IV Congreso de Tecnologías Aplicadas a la Enseñanza de la Electrónica (TAEE)**. Barcelona, Spain. Septiembre. 2000.

[4] Walrath, Kathy y Campione, Mary. The JFC Swing Tutorial, a guide to construct GUIs. ISBN 0-201-43321-4. Ed. Addison-Wesley.

Estructura de computadores, simuladores e Internet

Javier García Zubía y José María Sáenz Ruiz de Velasco
Dpto. de Arquitectura de Computadores, Dpto. Ingeniería del Software
Facultad de Ingeniería
Universidad de Deusto
48007 Bilbao
e-mail: zubia@eside.deusto.es, jmsaenz@eside.deusto.es

Resumen

El trabajo presenta la aplicación Simul_Intenet que aborda la simulación de dos computadores básicos. Las ventajas de esta aplicación se centran en el grado de detalle al simular y en que está disponible en Internet. Simul_Internet es interesante para los docentes y alumnos de un curso de Estructura de Computadores.

1. Introducción

Lo tratado en los siguientes párrafos afecta en la Universidad de Deusto a las asignaturas de *Estructura de Computadores I* de Ingeniería Informática (primer semestre del segundo curso) y de *Tecnología de Computadores* de Ingeniería de Telecomunicaciones (segundo semestre del pirmer curso), ambas de seis créditos: 4,5 teóricos + 1,5 prácticos.

A nuestro modo de ver, el objetivo principal que han de tener estas asignaturas es que el alumno sea capaz de diseñar completamente un computador básico y programar en él. El primer paso es un trabajo en el aula, mientras que el segundo necesita de un simulador.

En general, los simuladores permiten cargar un programa, ejecutarlo y observar los resultados. En algunos, además, la ejecución puede ser instrucción a instrucción. Por último, los simuladores más completos son capaces de simular estado a estado, a nivel de microinstrucción. Se puede hablar por tanto de tres niveles de simulación.

La aplicación Simul_Internet contempla los tres niveles de simulación y permite la modificación del camino de datos durante la ejecución del programa. En cuanto a la implementación, Simul_Internet ha sido desarrollada en JAVA y esta accesible desde Internet, con las ventajas (y desventajas) que esto supone.

A continuación se presentan los computadores simulados, el entorno de ensamblado y el simulador propiamente dicho.

2. Computadores básicos: la Máquina Sencilla, la M+ y la M++

Antes de pasar a los simuladores es necesario describir, aunque sea brevemente, los computadores simulados: la MS y la M+.

2.1. La Máquina Sencilla

La Máquina Sencilla (MS) es ya es un clásico en la disciplina. Fue diseñado por Valero y Ayguadé en 1989 [1]. Brevemente, la MS es un computador con estructura de Memoria-Memoria, con cuatro instrucciones (ADD F, D, CMP F, D, MOV F, D y BEQ D) y direccionamiento absoluto. La memoria es de 128x16 bits y el camino de datos aparece en la figura 1.

El autómata simplificado para la unidad de control cableda cuenta con 8 estados, tres entradas y 10 salidas, las propias de la unidad de control.

La ventajas de la MS son bien claras: sencillez extrema y facilidad de diseño. Todo esto sin dejar de ser un computador. Su desventaja principal es que la MS es demasiado simple, demasiado sencilla.

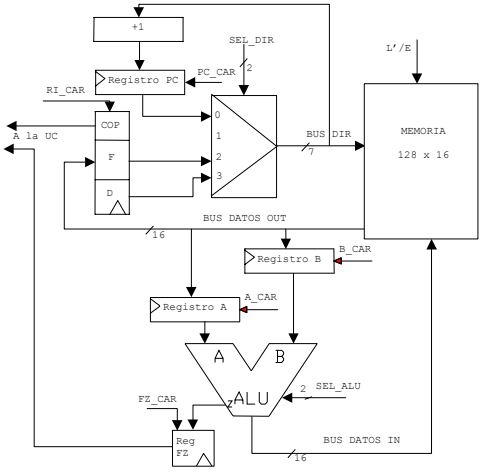


Figura 1. Camino de Datos de la Máquina Sencilla

2.2. La Máquina Plus (M+)

La M+ es un computador básico pero más evolucionado que la MS. Su primer diseño data del año 1997 [2].

La estructura de la M+ es Registro-Registro, tiene un único bus de datos bidireccional de 8 bits, 26 instrucciones (ADD, SUB, MOVE, AND, OR, XOR, NOT, CMP, INR, BEQ, BC y JMP, LDA, STA, LDAX, STAX, LFA y SFA) y un bus de direcciones de 16 bits potente y variado: inmediato, por registro, absoluto e indirecto. La memoria es de 64 Kbytes y el camino de datos puede verse en la figura 2.

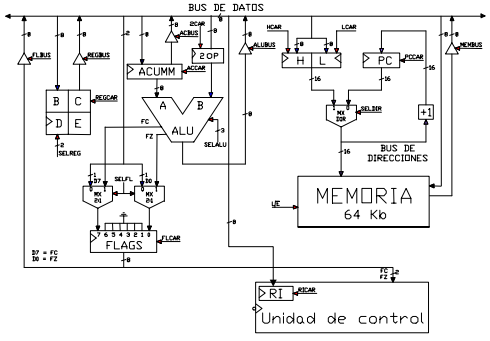


Figura 2. Camino de datos de la M+

El autómata de la unidad de control cableada de la M+ tiene 25 estados, 10 entradas (8 bits del código de operación y los flags FZ y FC) y 21 líneas de salida, las propias de la unidad de control.

2.3. El simulador de la Universidad de Deusto

El simulador desarrollado en la Universidad de Deusto, Simul_Internet, tiene como características principales:

- es accesible desde Internet,
- contempla conjuntamente a la MS y la M+,
- las unidades de control están implementadas cableadas mediante autómatas,
- niveles de ejecución: programa, instrucción y estado,
- visualiza dinámicamente el autómata de la unidad de control,
- permite la modificación interactiva de la máquina,
- carga y ensambla los programas con control del usuario y
- es potente, cómodo, visual y gratuito.
- Está accesible en: http://paginaspersonales.deusto.es/zubia

En la Figura 3 se ve el aspecto de la pantalla principal de Simul_Internet. En los siguientes párrafos describiremos brevemente la aplicación.

Figura 3. Pantalla principla de Simul_Internet

2.4. El ensamblador de Simul_Internet

En primer lugar se muestra el funcionamiento y aspecto del ensamblador de Simul_Internet (Traduct 2000). En la Figura 4 se puede ver como el proceso de ensamblado es una pantalla aparte. En ella el alumno escribe el programa en

ensamblador, y al ensamblarlo obtiene el código hexadecimal que deberá cargar en la memoria del computador correspondiente.

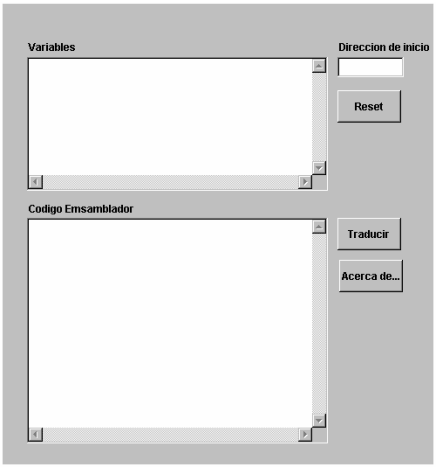


Figura 4. Pantalla principal del ensamblador

En la Figuras 5 y 6 se pueden ver, respectivamente, un programa en ensamblador de la M+ y el correspondiente código hexadecimal.

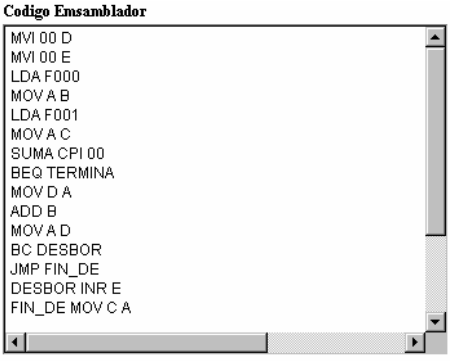


Figura 5. Programa en ensamblador de la M+

Una vez obtenido el código hexadecimal, el usuario dispone del mecanismo para cargar este programa en cualquier posición de la memoria.

2.5. La simulación de la MS

Si en la Figura 3 se hubiera pulsado la opción Máquina Sencilla, el alumno se habría encontrado con la Figura 7.

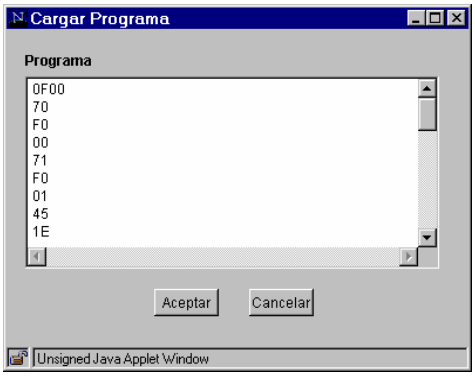


Figura 6. Programa en código hexadecimal

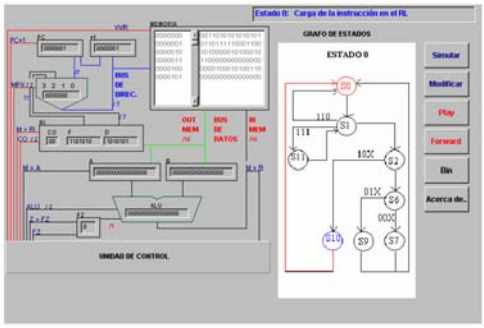


Figura 7. Pantalla del simulador de la MS

En la Figura 7 se pueden ver en la parte izquierda el camino de datos y las señales de control de la MS. Todos los registros, campos, etc. son accesibles por el alumno, que podrá modificarlos.

En la parte de la derecha se pueden ver los controles del nivel de simulación, la opción de modificación de campos y la elección binario/hexadecimal.

En la parte central se puede ver el diagrama de transición de estados del autómata de la unidad de control cableada. Cuando el alumno haya elegido la opción de simulación por estados podrá ver pintado en rojo el estado actual, y en azul el anterior. La visualización dinámica del autómata nos parece muy importante desde el punto de vista didáctico.

2.6. La simulación de la M+

Si en la Figura 3 se elige la opción Máquina Plus lo que nos encontramos es la Figura 8.

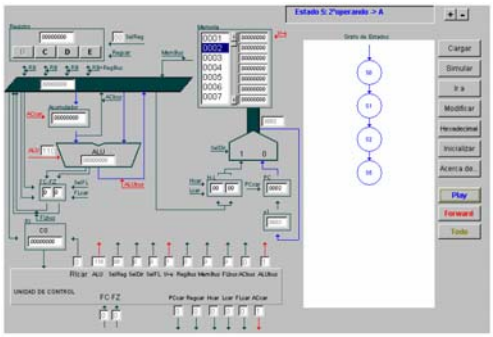


Figura 8. Pantalla del simulador de la M+

La pantalla es algo más compleja, pero igualmente manejable. Podemos ver en la parte izquierda el camino de datos y las señales de control, en la derecha, el control del nivel de simulación y en el centro, el diagrama de estados del autómata de la unidad de control cableada.

En cuanto a la simulación propiamente dicha, el alumno podrá ver cómo van cambiando el estado y contenido de la máquina con cada flanco de reloj. El Simul_Internet resalta visualmente, cambiando de color, aquellas líneas que modifican su valor.

La idea aplicada en Simul_Internet es que ha de verse todo, y todo debe ser accesible, siendo el alumno el que tiene el control de la máquina.

3. Experiencia docente

Desde el curso 1999-2000 el alumno dispone de una primera versión de Simul_Internet. Bastará una clase de 2 horas para que el alumno se haga con la aplicación.

La aplicación es utilizada por más de 300 alumnos al año. Las asignaturas implicadas son, como ya se ha dicho, *Estructura de Computadores I* en el primer semestre del segundo curso de Ingeniería Informática y *Tecnología de Computadores* del segundo semestre del primer curso de Ingeniería en Telecomunicaciones.

Aun a falta de estudios formales, la experiencia es positiva, y así nos lo hacen saber los alumnos. Ellos destacan de Simul_Internet su sencillez, potencia, adecuación al aula y disponibilidad en la red, y señalan como desventaja los problemas derivados de "la lentitud Internet".

4. El simulador e Internet

Un aspecto novedoso de Simul_Internet es que utiliza a Internet como plataforma. La aplicación se ha desarrollado en JAVA. La experiencia ha sido positiva en general, pero también ha habido problemas de lentitud. Por un lado la red es lenta y por otro, toda aplicación JAVA necesita ser interpretada por una máquina virtual de alto nivel, lo que no hace sino retardar todavía más la ejecución. Como opción a esta lentitud está que el alumno descargue la aplicación y la ejecute en su propio computador.

La ventaja de usar Internet reside en que la aplicación está disponible de continuo y en que la aplicación que se distribuye siempre está actualizada. Además poco a poco todos los centros de enseñanza están haciendo esfuerzos por trasladar parte de sus clases a la red.

Los requisitos para ejecutar la aplicación son un computador cualquiera tipo Pentium, un navegador actualizado (Internet-Explorer, Netscape, Mozilla, etc.) y el runtime para ejecutar programas en JAVA 1.3 o posterior.

5. Conclusiones

En el trabajo se ha presentado el simulador Simul_Internet. Su interés y originalidad se centran especialmente en su disponibilidad vía WEB, en el abanico de posibles niveles de simulación y en el control total que el usuario ejerce. Simul_Internet permite al profesor enseñar computadores básicos con el nivel y enfoque que el profesor desee, convirtiéndose en una herramienta ideal para enseñar en el aula.

Referencias

[1] Valero, M. y Ayguadé, E. *La Máquina Sencilla: Introducción a la estructura básica de un computador,* Facultad de Informática de la Universidad Politécnica de Cataluña, 1989.

[2] Angulo Usategui, J.M. García Zubía, J. y Angulo Martínez, I. *Fundamentos y estructura de computadores,* Ed. paraninfo, 2003.

CGRAPHIC: una herramienta gráfica para la enseñanza de los fundamentos de la programación (usando C)

Antonio J. Fernández, Jesús Millán Sánchez
Dept. de Lenguajes y Ciencias de la Computación, ETSII,
Campus de Teatinos, Universidad de Málaga
29071 Málaga
e-mail: afdez@lcc.uma.es

Resumen

Se describe una herramienta multimedia de apoyo docente para la enseñanza de los conceptos básicos de la programación. Esta herramienta consiste en un tutorial-simulador gráfico accesible en la Web que permite al alumno ahondar, gráfica y gradualmente, en los conceptos fundamentales de la programación imperativa.

1. Motivación y trabajo relacionado

En las ingenierías universitarias, la programación es una de las asignaturas básicas que más tiempo necesita para su comprensión y además, es un hecho reconocido que el lenguaje C es elegido en muchas universidades españolas como primer lenguaje de programación. Particularmente, [3] discute las adecuación de utilizar C como primer lenguaje de programación en las ingenierías. Entre sus ventajas citamos su practicidad, la generalidad de uso, motivación para el alumno, existencia de numerosas aplicaciones industriales escritas en C así como la disponibilidad gratuita de múltiples entornos para C. La principal desventaja consiste en que "es un lenguaje difícil tanto de enseñar como de aprender en un primer curso" universitario.

Un aspecto que contribuye a una mejor comprensión de la programación es la visualización dinámica de sus conceptos coordinada con la ejecución gradual de las sentencias fuente. En realidad existen muchos trabajos con fines educativos para facilitar el aprendizaje, aunque la mayoría se enfocan en el nivel físico sobre el funcionamiento interno de la máquina [2,5] y otros incluyen algún tipo de animación visual [4,6,9]. Sin embargo, se aprecia una carencia de simuladores orientados a lenguajes de alto nivel (e.g., [1] simula ejecuciones de código en PASCAL).

Para reducir en alguna medida la dificultad de enseñar y aprender los conceptos básicos de la programación usando el lenguaje C, hemos elaborado una herramienta que permite ahondar, de forma gradual y gráfica, en los fundamentos de la programación imperativa. Esta herramienta, que llamamos *CGRAPHIC*, es original (i.e., no está basada en otras) y complementa, de forma gráfica e interactiva, los conocimientos adquiridos en clase, los cuales se explican de forma intuitiva.

2. CGRAPHIC como software educativo

CGRAPHIC puede ser fácilmente catalogado como software educativo [8] puesto que cumple muchas de sus características tales como

- facilidad de uso,
- capacidad de motivar al alumno,
- tratamiento de temas relevantes,
- versatilidad y accesibilidad,
- interactividad con el alumno y
- originalidad.

Además, CGRAPHIC cumple las principales funciones del software educativo [8], tales como:

- *una función instructiva* pues orienta el aprendizaje de los estudiantes;
- *una función motivadora* ya que incluye elementos gráficos que captan la atención de los alumnos, mantienen su interés y los guía hacia los aspectos más importantes de los conceptos;
- *una función evaluadora* porque da una respuesta inmediata a las acciones de los alumnos según éstos las demanden.

3. Teoría + práctica

CGRAPHIC integra un tutorial on-line, de los conceptos básicos del lenguaje C, y un simulador visual de la ejecución de programas escritos en C, combinando pues una parte teórica con otra eminentemente práctica.

En la parte teórica, cada concepto básico de programación viene acompañado primeramente de una explicación teórica similar a la que pueda aparecer en cualquier libro básico de programación. Primero se define un concepto y luego se muestran, a nivel teórico, ejercicios explicativos del mismo. Además, se proponen una lista de conceptos afines que el alumno puede consultar de forma inmediata on-line.

En la parte práctica, asociado a cada concepto existe un conjunto de ejercicios interactivos cuya ejecución es visualizada gráficamente. En este contexto práctico, se combinan características de un intérprete de C con las de un depurador del mismo. Básicamente CGRAPHIC visualiza la ejecución de un programa diseñado mediante el lenguaje de programación C. La versión actual incluye un conjunto amplio de programas -de dificultad diversa- que se asocian a conceptos fundamentales de la programación imperativa, tales como: tipos básicos, tipos compuestos (registros-estructuras, arrays, etc), constantes, variables locales y globales, ámbitos de visibilidad, estructuras de control, bucles, direcciones de memoria, punteros, ficheros, funciones, procedimientos, paso de parámetros y tipos abstractos de datos (TADs).

4. El entorno de visualización

El entorno gráfico de ejecución de la parte práctica consta de lo siguientes elementos, los cuales son fácilmente identificables en la Figura 1: (1) *Una pantalla gráfica* para mostrar una representación gráfica de los elementos del programa C a medida que éstos son creados, modificados y/o eliminados; (2) *Una pantalla de código* que muestra el código del ejercicio actual que se está representando; además en cada uno de los estados intermedios, se resalta cuál es la siguiente línea de código que va a ser ejecutada; (3) *una pantalla de salida,* que indica tanto la salida estándar (es decir el monitor) del programa como instrucciones al usuario para un correcto funcionamiento; (4) *un control del modo de ejecución* (ver explicación posterior)*;* (5) *un cuadro de entrada,* que proporciona interactividad pues permite introducir datos como si del teclado se tratase; (6) un botón para iniciar la visualización *(ejecución) y* (7) *un botón de acción,* para pasar de un estado a otro en la ejecución.

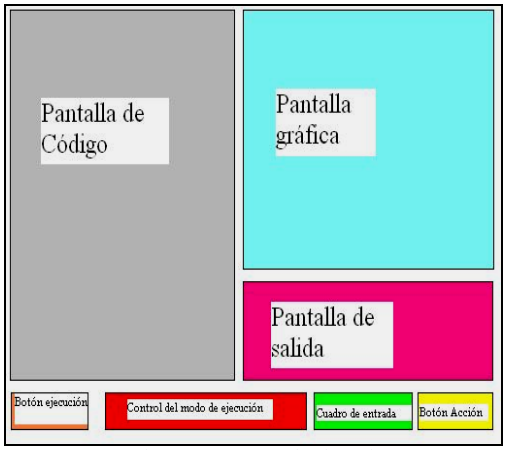


Figura 1. Entorno de ejecución.

La implementación: Debido a las limitaciones de espacio, comentaremos brevemente la implementación de CGRAPHIC. La parte teórica ha sido diseñada básicamente mediante páginas HTML; el entorno gráfico ha sido desarrollado en Java y sólo requiere un programa navegador para ser ejecutado. La Figura 2 muestra el esquema de todas las clases empleadas: se han implementado una serie de objetos genéricos que son usados para visualizar diferentes conceptos de programación. Entre esos objetos tenemos: variables, arrays, funciones, punteros, mapas de memoria, ficheros, estructuras y buffer de teclado. Cada objeto se implementa mediante una clase Java y cada ejercicio (e.g., *Ex*), implementado en otra clase Java (e.g., *Ex.java*), contiene un programa C que se carga durante la ejecución en la ventana de código del entorno (ver Figura 1). La ventana de código es definida en el fichero *Code.java*. El resto del entorno gráfico se implementa en el fichero *GraphicBox.java*, en cual es compartido por todos los ejercicios (los cuales también comparten la misma plantilla, implementada en el fichero *Template.java*). Más información puede ser encontrada en la siguiente dirección:

http://campusvirtual.uma.es/fundinfo/online/English_CGRAPHIC/default.htm.

Modos de ejecución: Se permiten dos modos posibles de ejecución a elegir por el usuario: un modo *directo*, en el cual el programa se ejecuta tal y como lo haría en un entorno clásico de programación en C, y un modo *pausado o paso a paso*, en el cual los programas son ejecutados instrucción a instrucción y la resolución (trazas) de cada instrucción es mostrada gráficamente por pantalla. En este modo, el usuario indica cuándo se quiere pasar al siguiente estado mientras se observan cómo varían los elementos del programa (variables, punteros, memoria,...) en los estados intermedios; consiste pues en una interpretación gráfica del ejercicio.

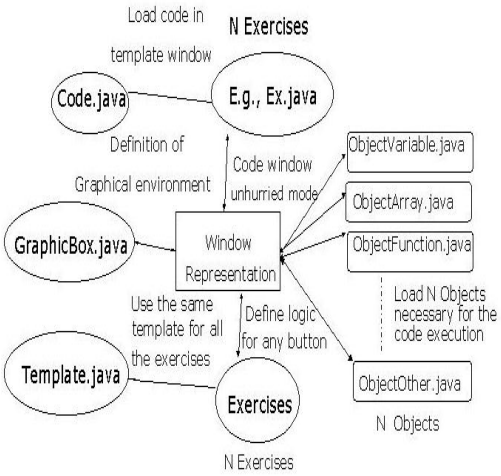


Figura 2. Esquema de Clases.

Ejemplos: La Figura 3 muestra un estado intermedio derivado de la ejecución paso a paso de un ejercicio (concretamente, el de intercambio de enteros usando una función *swap*). En esta figura se observan algunos de los *objetos* tratados gráficamente en la herramienta tales como las variables globales (en verde), las variables locales (en azul), el mapa de memoria, la línea de código a ejecutarse (resaltada en rojo), etc.

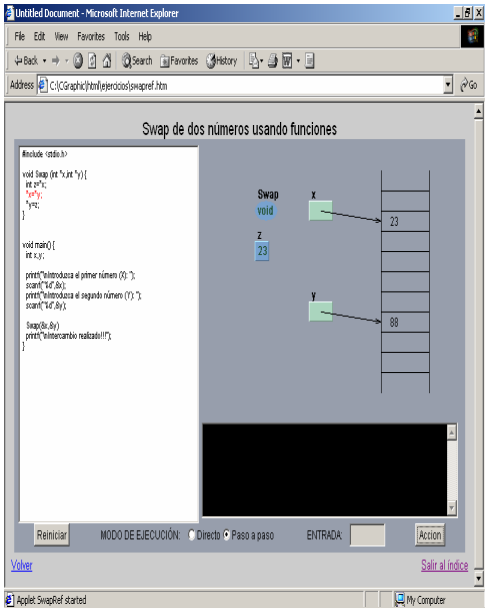


Figura 3. Estado intermedio de ejecución *paso a paso*.

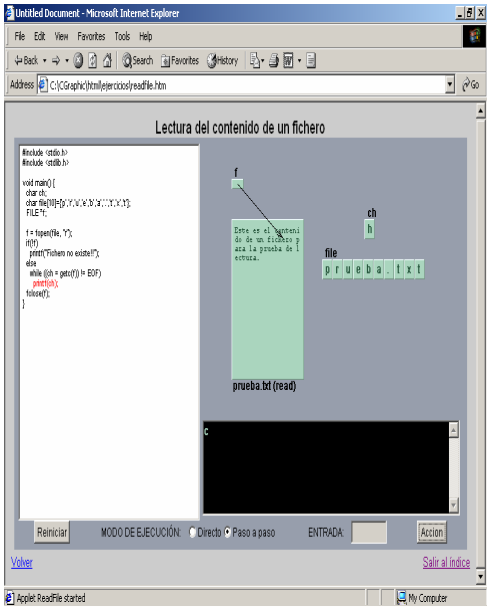


Figura 4. Otros estados intermedios. Ficheros-Arrays.

Las Figuras 4 y 5 muestran el estado intermedio de la ejecución pausada de otros

ejercicios relacionados con otros conceptos (en la Figura 4 la lectura de ficheros y las cadenas de caracteres; en la Figura 5, las direcciones de memorias, punteros, funciones y el TAD pila).

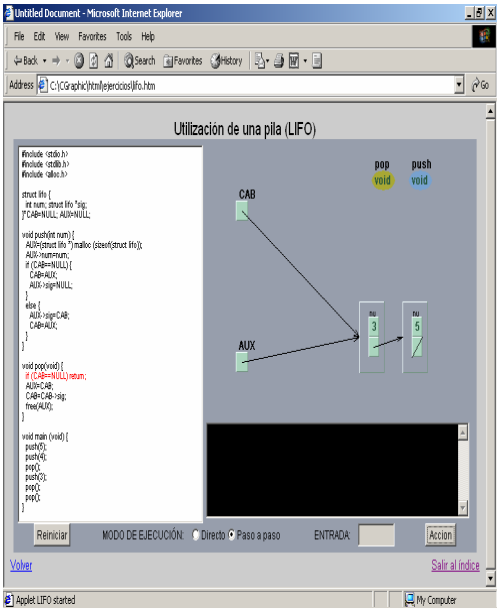


Figura 5. Otros estados intermedios. TADs-punteros

5. Conclusión

Hemos presentado CGRAPHIC una herramienta original diseñada exclusivamente para la enseñanza, usando C, de los conceptos fundamentales de la programación imperativa. Esta herramienta reduce el enorme hueco existente entre la asimilación de la teoría y su utilización práctica cuando se utiliza el C como primer lenguaje de programación. CGRAPHIC está totalmente operativa y accesible en la siguiente dirección web: *http://campusvirtual.uma.es/fundinfo/ (enlace On-line).*

Evaluación Con respecto a la evaluación de la herramienta, todavía es pronto para analizar su uso por parte de los alumnos pues ha comenzado a ser operativa en este segundo cuatrimestre. Sin embargo diremos que, desde el punto de vista del profesor, nos está resultando muy útil a la hora de explicar conceptos difíciles tales como la "entrada de datos bufferizada". En este sentido estamos usando el típico cañón para proyectar en las clases prácticas la visualización paso a paso de ejercicios, la cual es complementada con la explicación teórica del profesor. Esto facilita la comprensión del concepto por parte del alumno.

Al final del curso, proyectamos realizar una evaluación plena, siguiendo criterios ya establecidos [7].

Referencias

[1] Brette J-F. Transparent running and contextual help to learn and to teach an imperative language. *SIGCSE Bulletin*, 27(2):7-12, 1995.

[2] Campbell R. Introducing computer concepts by simulating a simple computer. *SIGCSE Bulletin*, 28(3):9-11, 1996.

[3] Fernández A.J., Trella M., Aranda M.C. y Galindo P. *Nuevas tecnologías y metodologías para la enseñanza de una introducción a la programación de ordenadores en ingenierías.* III Congreso Chileno de Educación Superior en Computación, Chile, 2001.

[4] Garralón J., Contreras D. y Núñez P. *PERICO: Programa Educativo de pRogramación Imperativa con visualización de elementos fisiCOs*. II Jornadas Andalusíes de Informática Gráfica, pp:63-72, 2000.

[5] Hassapis G. An interactive electronic book approach for teaching computer implementation of industrial control systems. *IEEE Transactions on Education*, 46(1):177-184, 2003.

[6] Henderson W. Animated models for teaching aspects of computer systems organizations. *IEEE Transactions on Education*, 37 (3):247-256, 1994.

[7] Iglesias O., Paniagua C. y Pessacq R. Evaluation of University Educational Software. *Computer Applications in Engineering Education*, 5(3):181-188,1997.

[8] Marques P. *Software educativo. Guía de uso y metodología de diseño*. Estes, 1995.

[9] Robbins S. y Robbins K. A microprogramming animation. *IEEE Transactions on Education*, 41(4):293-300, 1998.

Una Aplicación Interactiva para Visualizar las Hipótesis Generadas por Algoritmos de Aprendizaje Computacional

Santiago David Villalba Bartolomé, Juan José Rodríguez Díez*
*Área de Lenguajes y Sistemas Informáticos
Universidad de Burgos
e-mail: sdvb@wanadoo.es, jjrodriguez@ubu.es

Resumen

Se presenta una herramienta que puede ser empleada en la docencia de asignaturas de Minería de Datos, Aprendizaje Computacional, o de Inteligencia Artificial, que incluyan estas cuestiones dentro de su temario. Inspirada en diversas aplicaciones que visualizan las superficies de decisión generadas por algún tipo de clasificador, la presente herramienta es capaz de mostrar gráficamente la hipótesis lanzada por una vasta gama de clasificadores, desde árboles de decisión hasta redes neuronales, en el marco de un problema bidimensional (sencillo aunque visualizable) y multiclase.

1. Introducción

OAIDTB (Otra Aplicación Interactiva Demostrando Técnicas de Boosting) es una suite de programas para la aplicación y evaluación de técnicas de clasificación en general, y de boosting en particular, a problemas de aprendizaje computacional. Los programas que incluye la suite son los siguientes:

- Una biblioteca de clases java que implementan diversos algoritmos de boosting, extendiendo a la biblioteca WEKA [11]; a este conjunto de clases las llamaremos a partir de ahora boosters. Pueden ser utilizados desde la línea de comandos o embebidos en otro código java.
- Una aplicación gráfica para la selección, configuración, aplicación y evaluación de los métodos de aprendizaje tanto de WEKA como de OAIDTB.
- Una aplicación gráfica docente, *Bidimensional Generalization*, que permite visualizar, de una manera general, las hipótesis lanzadas por los clasificadores de WEKA y OAIDTB en el marco de un problema bidimensional y multiclase.

Existen diversas herramientas capaces de visualizar las superficies de decisión generadas por algún sistema de clasificación, como redes neuronales [13], máquinas de vectores soporte (SVM) [12], boosting [9][10], etc. La aportación principal de la herramienta que se presenta en este trabajo es que es capaz de visualizar una amplia gama de clasificadores, todos los incluidos en la biblioteca WEKA [11], así como aquellos que se implementen dentro del marco definido por esta biblioteca. En particular, si en la asignatura se exige la implementación de algún algoritmo, los alumnos son capaces de visualizar las hipótesis generadas por el clasificador que han desarrollado sin necesidad de programar la parte gráfica.

OAIDTB es software de código abierto y se distribuye bajo la licencia GNU General Public License. Se puede encontrar la última versión en http://pisuerga.inf.ubu.es/lsi/Asignaturas/MD/.

2. Un poco de teoría

A pesar de que se presupone en el lector un cierto conocimiento de los fundamentos necesarios para comprender el presente texto, se dan a continuación unas breves definiciones, más o menos precisas, de varios conceptos fundamentales.

- *Minería de datos:* Aplicación de algoritmos específicos para la extracción de patrones desde los datos.

- *Aprendizaje computacional:* En el contexto en que nos encontramos, este término se refiere al conjunto de teorías y algoritmos que forman la base técnica para la minería de datos.
- *Instancia:* La entrada para un esquema de aprendizaje computacional es un conjunto de instancias. Estas instancias son las "cosas" que deben ser clasificadas, agrupadas o asociadas. Cada instancia es un ejemplo individual e independiente del concepto que debe ser aprendido y cada instancia está caracterizada por los valores que toman un conjunto predeterminado de atributos.
- *Clasificador:* un algoritmo de aprendizaje clasificador trata de extraer información de un conjunto de instancias de entrenamiento, de manera que es capaz de predecir la clase a la que pertenecen ejemplares desconocidos (distintos de los utilizados para entrenarlo).
- *Boosting:* Los algoritmos de aprendizaje computacional basados en boosting (potenciación) buscan iterativamente una combinación lineal ponderada de clasificadores base que haga predicciones correctas en datos desconocidos.

Las clases que implementan a los algoritmos de boosting son el principal componente de la suite. Los algoritmos implementados son AdaBoostM1 [4], AdaBoostM1W [2], AdaBoostOC [6], AdaBoostECC [8], AdaBoostMH [5], RealAdaBoost [3], GentleAdaBoost [3], AdaCost [1] y CSBx [7].

3. Las aplicaciones gráficas

En el terreno de la didáctica son las aplicaciones gráficas de la suite OAIDTB las que tienen algo o mucho que decir.

La interfaz gráfica ha sido diseñada con el afán de ser cómoda e intuitiva para el usuario, proporcionando ayuda sobre qué es y para que sirve cada componente mediante "tooltips". El funcionamiento de las aplicaciones se basa, fundamentalmente, en el empleo del ratón, aunque se proporcionan teclas de acceso rápido a la mayoría de las funciones. Además, las aplicaciones no pierden en ningún momento la interactividad con el usuario, pudiendo siempre ser cancelados los procesos lanzados a pesar de que puedan, eventualmente, tener una importante carga computacional.

Sin más preámbulos, echemos un vistazo a lo que nos ofrecen estos programas.

3.1. El panel de clasificación

El panel de clasificación (Fig. 1) es una herramienta que permite seleccionar, o cargar un clasificador de OAIDTB o de WEKA, configurarlo, entrenarlo sobre un conjunto de instancias, guardarlo, evaluar su precisión proporcionando variadas estadísticas y una representación textual del clasificador y, en el caso de los boosters, mostrar detalles tales como la evolución del error, mediante gráficas cartesianas, o la precisión de los clasificadores base entrenados.

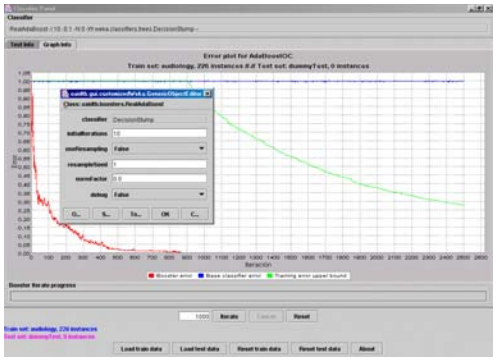


Figura 1. El panel de clasificación

La aplicación es flexible, ya que añadirle nuevos esquemas de clasificación es sencillo. Basta con implementar el algoritmo deseado en el marco proporcionado por la biblioteca WEKA y añadir el nombre de la clase a un fichero de texto que permite configurar la aplicación. La utilización de las capacidades de java para descubrir los miembros de las clases en tiempo de ejecución proporciona la posibilidad de incorporar estas clases, con sus características particulares, sin tener que recompilar el programa.

3.2. "Generalización Bidimensional"

Esta aplicación permite la visualización de las hipótesis lanzadas por diversos algoritmos de

aprendizaje computacional en el marco de un problema de datos bidimensional y multiclase.

En concreto, se trata de clasificar instancias representadas por puntos en un espacio bidimensional y cuyo atributo clase es el color de dicho punto. Con ella es posible enseñar, de una manera práctica y visual, cómo funcionan dichos algoritmos, mostrando propiedades teóricas de los mismos y conectando las hipótesis que lanzan con un problema que, aunque limitado, es fácil de entender y "entra por los ojos".

Una vez ejecutada, la ventana de la aplicación se divide en varias zonas (Fig. 2).

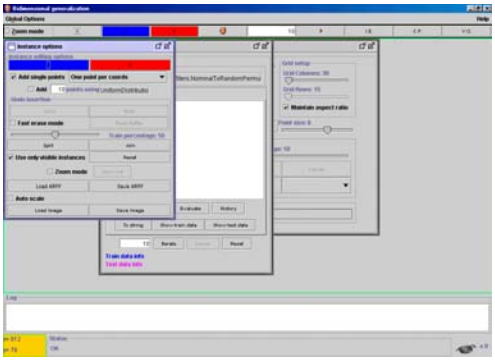


Figura 2. Pantalla principal de la aplicación

En la parte superior existe un pequeño menú con algunas opciones de configuración de la aplicación, y una barra de herramientas con botones para las acciones más frecuentes. En el centro se sitúa el área de dibujo, que es la región de la pantalla en que se dibujan las instancias (puntos de un determinado color) que introduce el usuario mediante pulsaciones del ratón. En la parte inferior existe un área de mensajes, en la que se le muestran al usuario las informaciones pertinentes, desde tiempos de ejecución hasta los posibles errores o excepciones producidas.

Flotando sobre todos estos componentes hay tres ventanas cuyo tamaño, para comodidad del usuario, puede ser alterado.

La primera ventana se corresponde con el panel de clasificación, ya comentado en la sección anterior.

La segunda ventana es el panel de edición de instancias. Permite introducir las instancias mediante pulsaciones de ratón en el área de dibujado, o cargarlas desde un archivo, bien sea en formato propio (ARFF – Formato de Archivo Atributo-Relación) o, y esta característica es experimental, desde un archivo de imagen gif o jpeg. Si se utiliza la inserción directa mediante pulsaciones del ratón, este panel permite la selección del color (clase) de los puntos (instancias) introducidos, así como permite la posibilidad de introducir un número arbitrario de instancias. Algunas funciones ofrecidas a través de los componentes de este panel son:

- Deshacer / rehacer la inserción de instancias.
- Configurar la función empleada en la inserción masiva de instancias, pudiendo elegir entre diversas funciones de distribución (normal, uniforme… o incluso las creadas por un usuario avanzado mediante del empleo de una simple API) para poder crear problemas interesantes o visualmente atractivos.
- Trasladar y escalar las coordenadas de las instancias para poder mostrarlas correctamente o más claramente en el área de dibujado.
- Dividir las instancias en conjuntos de entrenamiento y de prueba.
- Salvar las instancias en formato ARFF o una instantánea del contenido del área de dibujado.

La tercera ventana se corresponde con el panel de opciones visuales, que es el encargado de manejar todo lo referente a la visualización del problema, desde qué aspectos del mismo se muestran u ocultan hasta el tamaño de los puntos que representan las instancias, y de lanzar la creación de la imagen que represente la hipótesis del clasificador seleccionado y construido en el panel de clasificación, pudiendo configurar su nivel de precisión

Otro de los puntos fuertes del programa es la capacidad de hacer "zoom ilimitado" sobre cualquier región del universo del problema y

construir la imagen de hipótesis sólo sobre dicha región, de manera que es posible revelar las peculiaridades del clasificador en cuestión, como por ejemplo los límites de decisión de un árbol o de un clasificador basado en instancias.

.

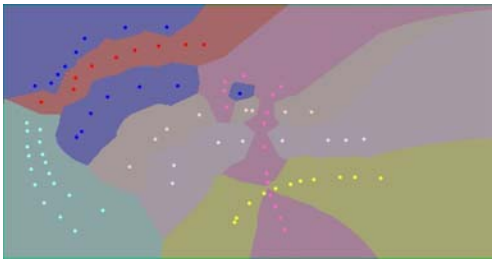


Figura 3. Hipótesis obtenida de un clasificador basado en instancias.

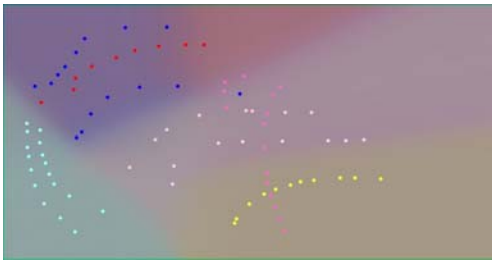


Figura 4. Hipótesis generada por una red neuronal

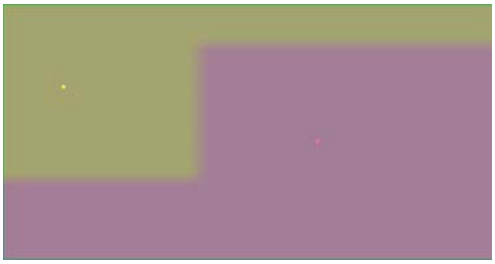


Figura 5. Zoom sobre una región de la figura 3

4. Conclusión

Se ha presentado una herramienta que permite representar, de manera gráfica y general, las hipótesis lanzadas por algoritmos de aprendizaje computacional basados en clasificación.

Referencias

[1] Fan, Stolfo, Zhang y Chan. *AdaCost: Misclassification cost-sensitive boosting*. Proceedings of The Sixteenth International Conference on Machine Learning. pp. 97-105. San Francisco: Morgan Kaufmann.

[2] Günther Eibl y Karl Peter Pfeiffer. *How to make AdaBoost.M1 work for weak base classifiers by changing only one line of the code.* ECML'02 - European Conference on Machine Learning

[3] Jerome Friedman, Trevor Hastie y Robert Tibshirani. *Additive logistic regression: a statistical view of boosting*. The Annals of Statistics, 38(2): 337-374, April 2000.

[4] Robert E. Schapire y Yoav Freund. *A decision-theoretic generalization of on-line learning and an application to boosting.* Journal of Computer and System Sciences, 55(1):119-139, 1997. van Leunen, M.C. *A handbook for scholars*. Oxford University Press, 1992

[5] Robert E. Schapire y Yoram Singer. *Improved boosting algorithms using confidence rated-predictions*. Machine Learning, 37(3):297-336, 1999

[6] Robert E. Schapire. *Using output codes to boost multiclass learning problems*. In Machine Learning: Proceedings of the Fourteenth Internatinal Conferences, pages 313-321, 1997.

[7] Ting, K.M.. *Cost Sensitive Classification Using Decision Trees, Boosting and MetaCost*. Book chapter in Heuristic and Optimization for Knowledge Discovery. Edited by Sarker, R., Abbass, H. & Newton, C. Idea Group Publishing. 2002.

[8] Venkatesan Guruswami y Amit Sahai. *Multiclass Learning, Boosting and Error Correcting Codes*. Proceedings of COLT'99

[9] [http://www.cs.huji.ac.il/~yoavf/adaboost/index.html], visitada el 23-1-03.

[10] [http://www.cs.technion.ac.il/~rani/LocBoost/], visitada el 23-1-03.

[11] [http://www.cs.waikato.ac.nz/ml/weka], visitada el 23-1-03.

[12] [http://svm.dcs.rhbnc.ac.uk/pagesnew/GPat.shtml], visitada el 23-1-03.

[13] [http://neuron.eng.wayne.edu/java/AHK/EPM_pps.html], visitada el 23-1-03

La herramienta ArtEM: aritmética entera y modular

Alfonso Gutiérrez, Violeta Migallón, José Penadés
Dpto. de Ciencia de la Computación e Inteligencia Artificial
Universidad de Alicante
03080 Alicante
e-mail: algutlan@hotmail.com
violeta@dccia.ua.es
jpenades@dccia.ua.es

Héctor Migallón
Dpto. de Física y Arquitectura de Computadores
Universidad Miguel Hernández
03202 Elche (Alicante)
e-mail: hmigallon@umh.es

Resumen

Los contenidos de matemática discreta en las titulaciones de informática incluyen una parte relativa a aritmética entera y modular. En este artículo presentamos una herramienta que ha sido diseñada para la realización de las prácticas de dicha parte y que actualmente se está utilizando en la asignatura Matemática Discreta de la Universidad de Alicante.

1. Introducción

La herramienta ArtEM (*Arit*mética *E*ntera y *M*odular) [3], es una aplicación informática programada en Visual Basic [5] y desarrollada con el fin de ser utilizada en las prácticas de cualquier asignatura que incluya como tópicos los relacionados con la aritmética entera y modular [1], [2], [4]. Está estructurada en 5 menús básicos:

- Euclides.
- Ecuaciones diofánticas.
- Números primos.
- Aritmética modular.
- Aplicación a la criptografía.

Los tres primeros menús están dedicados a la aritmética entera, el cuarto menú proporciona cálculos básicos en la aritmética modular como los cálculos del representante de clase, inverso de un elemento, función de Euler y potencias. El quinto menú constituye una aplicación a la criptografía centrándose en dos criptosistemas, uno de clave privada y otro de clave pública.

Todos los algoritmos disponibles en ArtEM se desarrollan de tal forma que el usuario es capaz de reconocer los pasos que se han seguido para su ejecución, de manera que se obtiene un importante valor pedagógico.

En las siguientes secciones describiremos el contenido de ArtEM, estudiando cada uno de sus menús por separado.

2. Menú Euclides

En este menú se desarrolla el algoritmo de Euclides para el cálculo del máximo común divisor de dos enteros. Además de describir el algoritmo de forma genérica se tiene la opción de mostrar todos los cálculos del propio algoritmo, tal y como se muestra en la Figura 1.

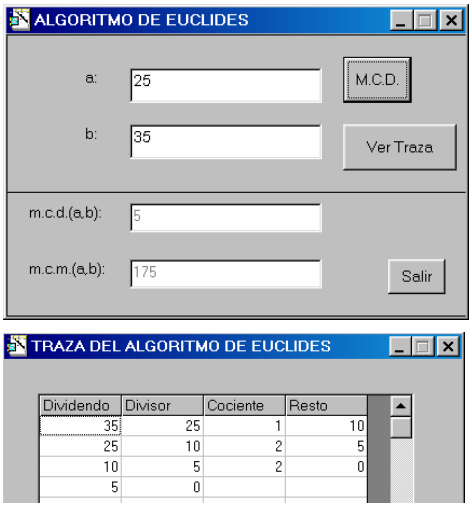


Figura 1. Algoritmo de Euclides

3. Menú ecuaciones diofánticas

En este menú se ofrece la posibilidad de resolver ecuaciones diofánticas, es decir, ecuaciones de la forma $ax+by=c$, donde a, b, c son enteros y x, y son las incógnitas que también son números enteros. Además de mostrar una descripción de los resultados teóricos necesarios para la correcta resolución de estas ecuaciones, se muestra el algoritmo necesario para el cálculo de una solución particular de una ecuación diofántica. En la ejecución del algoritmo, el usuario debe introducir los valores de a, b y c, obteniendo una solución particular de la ecuación diofántica correspondiente -cuya traza puede ser consultada- y la solución general. Como muestra presentamos la solución de la ecuación $2700x + 1500y = 234000$ en la Figura 2.

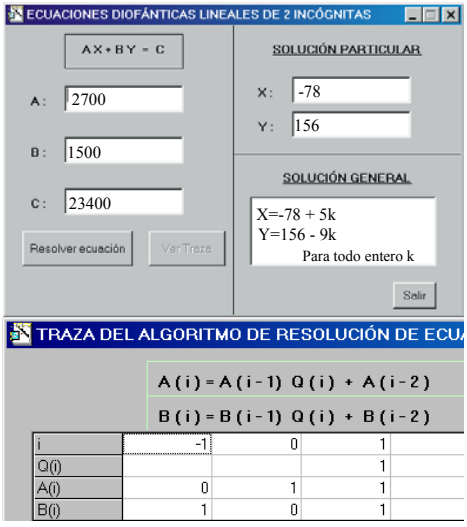


Figura 2. Ecuaciones diofánticas

4. Menú números primos

Se desarrollan en este menú procedimientos para crear una lista de números primos, averiguar si un número entero es primo y factorizar un entero en producto de sus primos. Estos algoritmos vienen acompañados de su descripción formal. La complejidad de estos algoritmos limita su uso a enteros pequeños. Las opciones que presenta este menú vienen indicadas en la Figura 3.

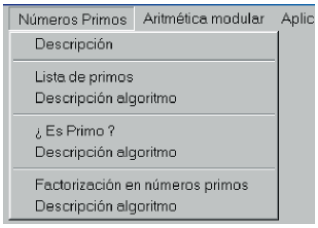


Figura 3. Opciones del menú números primos

5. Menú aritmética modular

Presentamos en este menú diversos cálculos básicos relacionados con la aritmética modular. Éstos son el cálculo del representante de clase en el conjunto de los enteros congruentes módulo n, que representamos por Z_n, el cálculo del inverso en Z_n, el cálculo de la función de Euler y el cálculo de potencias en Z_n. Como muestra presentamos el cálculo de la potencia $[5]^{75}$ en Z_{23}. El programa identifica que el $mcd(5,23)=1$ y por tanto, como el valor de la función de Euler en 23 es 22, se tiene que $[5]^{22}=[1]$. Así, como $[5]^{75}=([5]^{22})^3\ [5]^9$ sólo será necesario calcular $[5]^9 = [5]^8\ [5]$, que en este caso es $[11]$. Mostramos, en la Figura 4, la salida que se obtiene de la ejecución correspondiente a la traza del algoritmo.

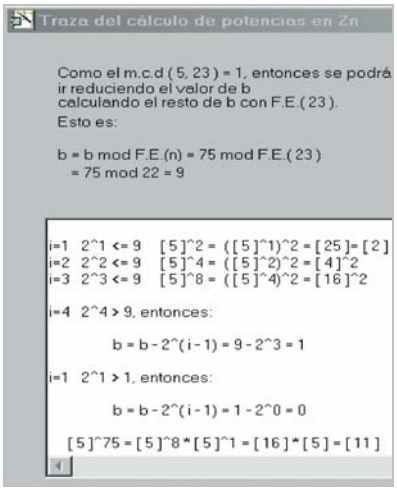


Figura 4. Cálculo de potencias en Z_n

6. Menú aplicación a la criptografía

En este menú pretendemos familiarizarnos con ciertas aplicaciones de la aritmética modular a la criptografía. Tiene dos partes claramente diferenciadas: la elección del alfabeto a utilizar y la elección del sistema criptográfico. En lo que se refiere a la elección del alfabeto, la aplicación tiene preestablecidos una serie de alfabetos que pueden ser seleccionados con el correspondiente menú, como muestra la Figura 5.

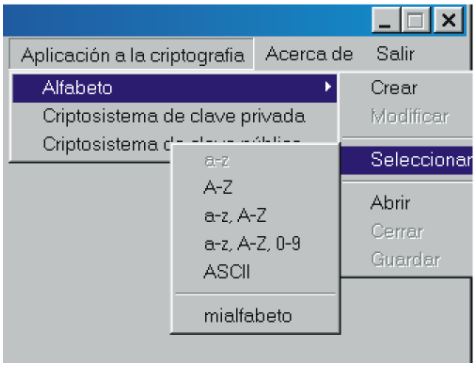


Figura 5. Elección del alfabeto

También se permite crear un alfabeto propio e incluso leerlo de disco si previamente se había creado. Para crear un alfabeto lo único que se debe hacer es ir asignando valores numéricos a cada uno de los caracteres que queremos que formen parte de nuestro alfabeto. El módulo con el que se trabajará en la codificación y descodificación vendrá dado en función del valor numérico asignado mayor. Como ejemplo, en la Figura 6, mostramos el alfabeto *{A,B,C,D,E,F,G}* al que se le han asociado las equivalencias numéricas *{11,16,1,23,20,17,24}* respectivamente y que en la Figura 5 viene definido con el nombre de *mialfabeto*.

Ya sea con un alfabeto creado por el usuario o con un alfabeto predefinido por la aplicación se dispone de dos tipos de criptosistemas: uno de clave privada y otro de clave pública. El criptosistema de clave privada corresponde con un criptosistema clásico cuyas funciones de cifrado y descifrado calculadas sobre Z_n son respectivamente:

$C_{r,s}([m]) = [r][m] + [s], \ / \ mcd(r,n)=1.$

$D_{r,s}([m^*]) = [r]^{-1}([m^*]-[s]).$

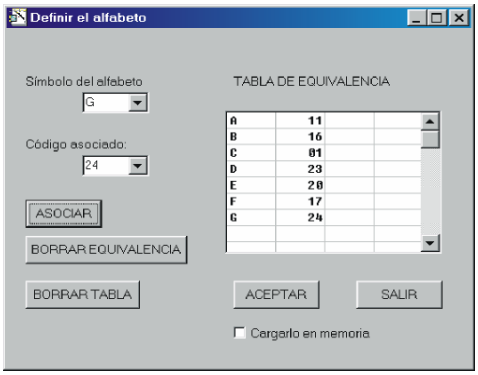


Figura 6. Definición de un nuevo alfabeto

Por su parte, el criptosistema de clave pública corresponde con el código RSA.

Como ejemplo de utilización de la aplicación ArtEM para este tipo de problemas, vamos a suponer que se ha seleccionado el alfabeto predefinido formado por los caracteres de la *A* a la *Z*, de la *a* a la *z*, y el espacio en blanco. Esto hace un total de *55* caracteres por lo que trabajaremos en Z_{55}. Vamos a realizar una codificación utilizando el criptosistema de clave privada. En primer lugar el programa nos pedirá *r* y *s*. Como *mcd(r,55)* debe ser *1*, el programa nos indica posibles valores de *r* a partir de un valor mínimo que el usuario introduce.

Si por ejemplo seleccionamos *s=8* y *r=6*, podremos, a través del botón *continuar*, iniciar una codificación con estas claves. La Figura 7 muestra la codificación de la frase "Esto es una prueba" usando este sistema criptográfico de clave privada y las claves anteriores. La primera ventana contiene la frase en cuestión que queremos codificar, la segunda ventana contiene la transcripción inmediata según el alfabeto que hayamos elegido y que se encuentra en la tabla de conversión, la tercera ventana contiene los valores numéricos de la codificación y la última ventana ya reproduce los caracteres codificados.

Así, con este sistema criptográfico la frase "Esto es una prueba" ha quedado codificada como "fJOñCcJCUhFCsDUcLF". La aplicación también permite invertir el proceso para descodificar un texto determinado. El proceso se realiza paso por paso pinchando en la correspondiente pestaña y en cada paso la aplicación nos da información de qué es lo que está haciendo.

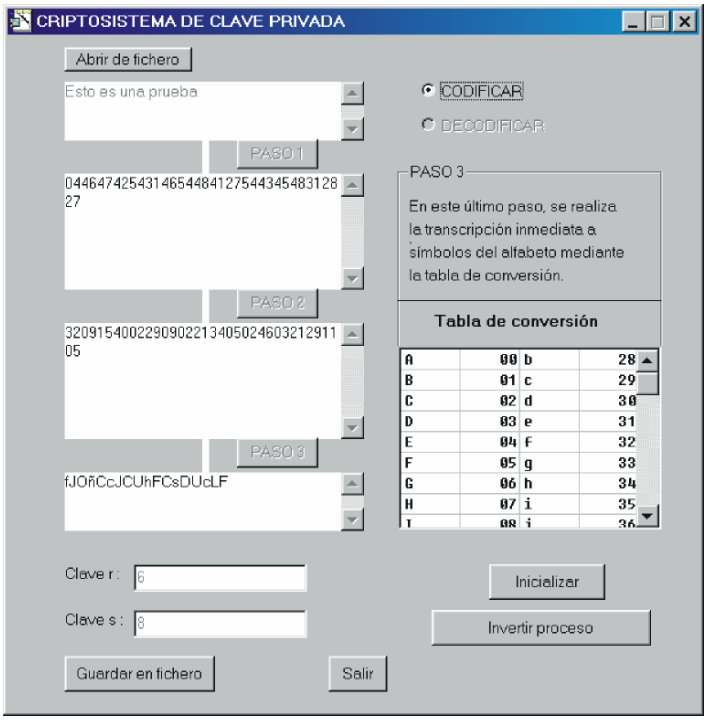


Figura 7. Ejemplo de codificación

7. Conclusión

El objetivo que nos marcamos con el diseño de la herramienta ArtEM fue el intentar impulsar el aprendizaje, experimentación, asimilación y ampliación de algunos de los contenidos de la matemática discreta, por parte del alumnado, con el uso del ordenador. No se trata de aprender a programar, pues para ello ya existen otras asignaturas, sino de aprovechar las capacidades pedagógicas del ordenador en beneficio de la calidad de nuestra docencia. La experiencia ha mostrado que el interés por parte del alumnado es muy aceptable y que además dichas prácticas facilitan la asimilación y comprensión de los contenidos de la aritmética entera y modular.

Tengamos en cuenta que, para el alumnado de informática en particular, esta materia tiene un grado de dificultad bastante considerable.

Referencias

[1] Biggs, N.L. *Matemática discreta.* Vicens Vives, 1994.

[2] Dierker, P.F., Voxman, W.L. *Discrete mathematics.* HBJ, 1986.

[3] Gutiérrez, A., Migallón, H., Migallón V., Penadés, J. *ArtEM.* Disponible en http://www.dccia.ua.es/~jpenades/ArtEM.html.

[4] Grimaldi, R.P. *Matemáticas discretas y combinatoria. Una introducción con aplicaciones.* Addison-Wesley, 1998.

[5] Petroutsos E., *Visual Basic 6.* Ediciones Anaya Multimedia S. A., 1999.

Uso de las referencias bibliográficas en la Ingeniería Informática[♣]

Francisco J. García Peñalvo
Dpto. de Informática y Automática
Universidad de Salamanca
37008 Salamanca
e-mail: fgarcia@usal.es

Roberto Therón Sánchez
Dpto. de Informática y Automática
Universidad de Salamanca
37008 Salamanca
e-mail: theron@usal.es

Ana B. Gil González
Dpto. de Informática y Automática
Universidad de Salamanca
37008 Salamanca
e-mail: abg@usal.es

Resumen
Las referencias bibliográficas que se incluyen en cualquier trabajo académico constituyen un importante valor a la hora de evaluar o clarificar los antecedes y/o las fuentes del trabajo en cuestión. No se suele considerar un factor crítico el correcto manejo de las citas bibliográficas hasta que los alumnos no están en el Tercer Ciclo de sus estudios, sin embargo, sería deseable que los alumnos de Primer y Segundo Ciclo tuvieran una mínima conciencia de la importancia que tiene una correcta lista de referencias, y fueran creando un hábito de organización de las fuentes que consultan. En este artículo se presenta una herramienta de soporte a la gestión de las fuentes bibliográfica, BiblioRef, desarrollada en el Departamento de Informática y Automática de la Universidad de Salamanca.

1. Introducción

Cualquiera que se haya enfrentado a la tarea de escribir un texto científico, habrá tenido que mediar con la labor de manejar las referencias bibliográficas que se pretenden incluir en el cuerpo del trabajo.

A la necesidad de obtener, seleccionar, consultar, comprender y sintetizar los contenidos de aquellas referencias que se consultan, hay que añadir la tarea de registrar las fuentes de nuestros conocimientos para poder referenciarlas en nuestras producciones científicas, siendo muy conscientes de que las listas de referencias bibliográficas que se incluyen en éstas son un criterio ampliamente utilizado para medir la calidad y la utilidad nuestras publicaciones.

La tarea de adaptar las referencias a un estilo concreto de citado se convierte en una labor tediosa, que se acrecienta cuando el número de referencias aumenta notablemente, especialmente para aquellos usuarios que trabajan en un entorno de procesamiento como puede ser MS Word.

Si bien es cierto que el correcto manejo de las fuentes bibliográficas no se les inculca a los alumnos hasta que llegan a su período doctoral, una lista de referencias escasa o inexistente empobrece enormemente los trabajos realizados durante sus estudios y especialmente los Proyectos Fin de Carrera (PFC) con los que culminan su carrera.

Los principales problemas que, en relación con la bibliografía, se han detectado en los PFC son:

- Carencia de cualquier tipo de información bibliográfica.
- Inclusión de una sección de bibliografía al final del documento, pero carencia de citas en el texto.
- Inclusión de una sección de bibliografía al final del documento en la que se mezclan tanto ciertas citas que se han incluido en el texto como otras obras consultadas pero no citadas.
- Aparición de citas en el cuerpo del documento, que luego no aparecen en la sección de referencias.
- Errores o carencia de un estilo de citado concreto, cuando no mezcla de varios estilos.

Nuestro Departamento está preocupado por facilitar la labor de gestión de las fuentes bibliográficas, así se ha desarrollado BiblioRef, herramienta que permite la gestión de los datos

[♣] Este trabajo ha sido parcialmente subvencionado por la Junta de Castilla y León y la Unión Europea a través del Fondo Social Europeo mediante el proyecto de investigación SA017/02.

asociados de las fuentes bibliográficas y su integración con MS Word, facilitando la inclusión de citas en un documento para, automáticamente, generar la sección de referencias de acuerdo a un estilo de citado.

Aunque BiblioRef fue desarrollada para los miembros del Departamento, algunos profesores estamos facilitándola a los alumnos para que gestionen las fuentes bibliográficas en sus PFC, obteniéndose una mejora en la calidad de la documentación de los mismos, a la vez que se les inculca un método de trabajo científico.

2. BiblioRef

Cuando el número de referencias a manejar crece, un procedimiento manual para la gestión de las fuentes bibliográficas se vuelve ineficiente y propenso a los errores. Este crecimiento plantea dos tipos de problemas: la gestión de los datos de las fuentes bibliográficas y el esfuerzo de ajustar la bibliografía a un formato concreto de citado.

Para solucionar el primer problema se puede recurrir a crear una base de datos personal, que facilite esa gestión de las fuentes bibliográficas. Pero para el segundo problema no existe una solución tan sencilla. Quizás el paradigma de actuación para este caso es el que se propone en LaTeX [2] donde, gracias a su interacción con BibTeX [3], se facilita enormemente la labor de adaptar las fuentes a un estilo de citado. Sin embargo, la utilización de LaTeX encuentra muchas reticencias en un amplio sector de usuarios, que se decantan por el otro estándar de facto a la hora de escribir un texto, MS Word. No obstante, MS Word no incorpora por sí mismo herramientas adecuadas para facilitar la gestión automática de las referencias bibliográficas.

BiblioRef [4] es una aplicación que cumple esta doble función de gestión de bibliografía y de inserción automática de las referencias en la sección de bibliografía conforme a un estilo de citado en un documento MS Word.

La interacción de BiblioRef con MS Word, viene a paliar una importante carencia de este último, ya que MS Word no aporta herramientas para la gestión bibliográfica, recayendo en el autor toda la responsabilidad del mantenimiento, normalmente realizado de forma manual, de la lista de referencias bibliográficas a incluir en un determinado documento.

La relación entre BiblioRef y MS Word es tal que cuando el autor requiere la inserción de una nueva cita, utiliza BiblioRef para seleccionarla de la colección existente (y ampliable en cualquier momento) y copiarla, pegándola en el punto de inserción dentro del documento Word. Una vez terminado el documento, se podrá generar automáticamente la sección de bibliografía del mismo, utilizando el estilo de citado elegido entre los soportados por BiblioRef. Además, no se olvida a BibTeX, ya que se siguen patrones de uso similares a los de este entorno, y se incorporan facilidades de exportación/importación de ficheros BibTeX, facilitando a los usuarios de BiblioRef no limitarse a un entorno concreto como es Word.

La Figura 1 ilustra cuál es el ciclo de vida de una fuente desde el momento en que se capturan sus datos hasta el instante en que se desea incluir en un documento.

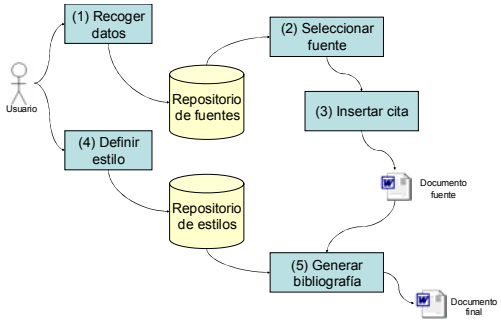


Figura 1. Ciclo de vida de una fuente bibliográfica

2.1. Recogida de datos

La primera fase en el ciclo de vida de una fuente bibliográfica es la extracción de los datos de la publicación, esto es, la información relevante que la define y que va a servir para catalogarla, buscarla o simplemente citarla.

Esta actividad que puede parecer trivial, pero nos puede obligar a invertir mucho tiempo, especialmente cuando no somos metódicos al extraer esos datos al obtener la publicación (ya sea físicamente, digitalmente o a través de referencias que aparecen citadas en otras publicaciones).

Para la correcta recolección de los datos de una fuente, con objeto de almacenarla en el repositorio de fuentes de la herramienta BiblioRef, debe conocerse cuál es el tipo de la publicación que está tratando. Los tipos de fuente bibliográfica que soporta BiblioRef, están muy influenciados

por los tipos soportado en BibTeX, aunque no existe una equivalencia completa. Concretamente se soportan: Artículos de revista, Libros, Capítulos de libros, Actas de congresos, Artículos en actas de congresos, Tesis, Documentos técnicos, Documentos electrónicos y Otras.

Para insertar los datos de una publicación (así como para editarlos) existe un asistente compuesto por cinco pasos, donde en el primer paso se introducen los datos básicos (entre ellos el tipo de la fuente, que incidirá en los datos concretos solicitados posteriormente), en el siguiente formulario se puede indicar si se tiene una versión digital de la publicación, en el tercero se introducen los autores por orden de firma (pudiéndose indicar si son o no editores de la publicación), en el cuarto paso es posible asociarle unos datos de clasificación a la fuente en base a áreas temáticas y materias concretas dentro de dichas áreas, y en el quinto y último paso se introducen los datos que completan la información de la fuente y que son dependientes del tipo de publicación de que se trate.

2.2. Selección de una fuente

A la hora de seleccionar una fuente para proceder a su inserción como cita en un documento se debe localizar ésta.

Cuando el número de citas que se manejan es pequeño, esa localización puede hacer a través de la ventana principal de la herramienta donde aparecen las fuentes en una rejilla, mostrándose los valores de algunos de sus atributos (título, tipo, año…) y con la característica de ordenar ascendente o descendentemente todas las fuentes por un campo, sin más que hacer clic con el puntero del ratón sobre el nombre de la columna que representa al atributo (Figura 2). Sin embargo, esta herramienta incluye una completa interfaz de búsqueda que permitirá obtener un subconjunto de las fuentes que se tienen almacenadas.

Figura 2. Presentación de las fuentes en la pantalla principal de BiblioRef

2.3. Inserción de una cita en un documento

Una vez que se tiene localizada la fuente que se desea citar, este paso es tan sencillo como dejar el punto de inserción del documento Word en la posición donde se quiere que se inserte la cita, una vez hecho esto se conmuta a la aplicación BiblioRef y se ejecuta la acción "Insertar Referencia" sobre la fuente seleccionada, y automáticamente se inserta un objeto BiblioRef en el documento Word (ver Figura 3).

2.4. Definición del estilo

A la hora de construir una lista de referencias en un documento, nosotros debemos indicar a BiblioRef cuál va a ser el estilo de citado que queremos emplear. Esta decisión implicará tanto el formato de las citas o llamadas que aparecerán en el cuerpo del documento, como el formato de los datos con que se muestra cada cita en la lista de referencias del documento.

Por defecto BiblioRef incluye tres estilos de citado: ACM, APA y un estilo de citado propio. Si estos estilos satisfacen al usuario podría generar de forma automática una sección de referencias en un documento. En caso contrario, el usuario cuenta con una interfaz avanzada donde puede definir de forma gráfica el estilo de citado que necesite.

2.5. Generación de la lista de bibliografía

El último paso sería la generación automática de la lista de referencias de un documento, en este caso BiblioRef sólo necesita que se le diga cuál es

el fichero que contiene los objetos referencia, cuál es fichero que contiene la definición del estilo y cuál es fichero destino que contendrá la sección de referencias generadas.

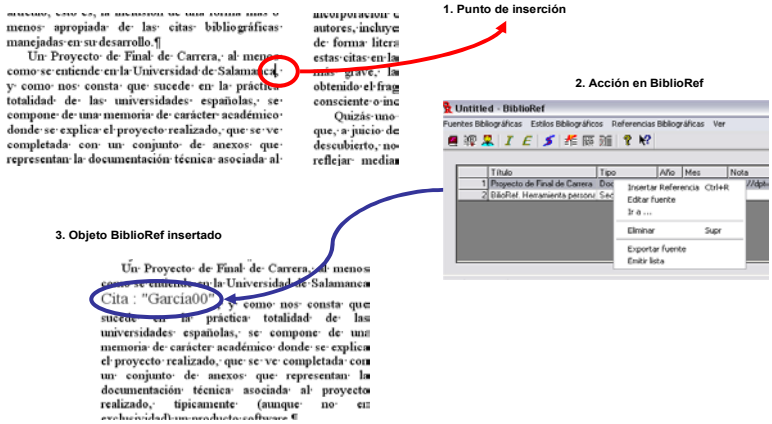


Figura 3. Inserción de una cita en MS Word

3. Trabajos relacionados

Existen numerosas herramientas de gestión de fuentes bibliográficas tanto de libre distribución, BibEdit (http://www.iui.se/staff/jonasb/bibedit/) o BibDB (http://www.mackichan.com), como de carácter comercial, ProCite (http://www.procite.com) o EndNote (http://www.endnote.com), pero estas herramientas se quedan en el ámbito de la gestión con la capacidad de exportar a formato BibTeX, pero no interaccionan con Word, uno de nuestros objetivos fundamentales.

Una herramienta que cuenta con características similares a BiblioRef, por cumplir el doble objetivo de la gestión y la integración con Word, es BibWord (http://mariachi.dsic.upv.es/bibword) [1]. En este caso la integración con Word se hace a través de macros programadas en WordBasic y no mediante una aplicación externa, y BibWord carece de soporte para la incorporación de nuevos estilos de citado.

4. Conclusiones

En este artículo se han realizado una serie de reflexiones sobre el escaso manejo que, en general, los alumnos de la Ingeniería Informática tienen sobre la gestión de la bibliografía y cómo esta circunstancia se refleja de una manera significativa en los PFC, que suelen quedar empobrecidos por la poca información bibliográfica que contienen.

Tratando de paliar este problema, se ha presentado BiblioRef, una herramienta que posibilita la gestión del conocimiento derivado de las fuentes bibliográficas, y que a su vez facilita el citado en entornos como MS Word y LaTeX. Esta herramienta, inicialmente desarrollada para facilitar la labor de los investigadores, está siendo puesta al alcance de los alumnos, obteniéndose prometedores resultados, al mejorar la calidad de los documentos en cuanto a las referencias y por crear en ellos un hábito de trabajo al organizar el conocimiento que adquieren de las fuentes bibliográficas que consultan.

Por último, decir que esta herramienta se puede conseguir para su libre utilización en http://tejo.usal.es/~fgarcia/docencia/isoftware/case/biblio.html.

Referencias

[1] Canós, J. H. A Bibliography Manager for Microsoft Word. *ACM Crossroads*, Vol. 6, Nº 4, 2000.

[2] Lamport, L. *A Document Preparation System*. Addison-Wesley, 1986.

[3] Patashnik, O. BibTeXing. BibTeX Distribution, 1988.

[4] Sánchez, J. M. García, F. J., Hernández, J. A. BilioRef. Herramienta personal para la gestión de citas bibliográficas. *Actas de ISKO 2003*.

SldDraw: Un trazador de árboles SLD

Francisco Gutiérrez
Dpto. de Lenguajes y Ciencias de la Computación
Universidad de Málaga
e-mail: pacog@lcc.uma.es

Mª del Carmen de Castro
Dpto. de Lenguajes y Sistemas Informáticos
Universidad de Cádiz
e-mail: maricarmen.decastro@uca.es

Resumen

Se presenta una aplicación software, en el ámbito de la Programación Lógica, que dibuja árboles de ejecución del método de resolución SLD. La aplicación está diseñada tanto como herramienta didáctica para el aprendizaje del método de resolución SLD con todas sus variantes como herramienta para el tutor que le permite encontrar árboles SLD con las características deseadas. Está destinada a alumnos y profesores que imparten docencia en niveles universitarios en asignaturas relacionadas con la Programación Lógica.

1. Introducción

En un principio, se pensó diseñar una herramienta de ayuda al profesor a la hora de proponer ejercicios que tracen árboles de ejecución por el método SLD [1] (árboles SLD). Antes de pasar a la realización de la aplicación se buscaron utilidades que ya resolvieran ese problema. La más parecida es la que se ofrece en un paquete junto el texto "Computational Intelligence. A Logical Approach" [2]. De todas formas, aunque esta herramienta genera árboles SLD, su propósito no es el mismo que el que guía a la aquí presentada.

SldDraw[3] pretendía cubrir una necesidad docente. Normalmente, a la hora de proponer ejercicios de traza de árboles SLD, el profesor se encuentra con la dificultad de encontrar árboles adecuados, en espacio y complejidad. El trazado a mano de los árboles es una tarea de cierta complejidad y encontrar esos árboles adecuados requiere mucha experiencia y sobre todo tiempo. La herramienta conseguida permite generar árboles SLD de forma automática a partir de un programa y un objetivo, e ir modificando tanto el programa como el objetivo hasta conseguir el árbol de tamaño adecuado.

Posteriormente se pensó que la aplicación también podía servir como herramienta didáctica en manos de los alumnos y profesores por lo que se le incorporaron más facilidades como cambios de reglas de búsqueda y selección, descripción detallada de las unificaciones que aparecen en cada rama del árbol, ejecuciones paso a paso, de éxito en éxito, puntos de ruptura, etc.

Con objeto de poder hacer trazas de ejecución de programas escritos en Prolog, se incorporan también características propias del lenguaje como aritmética extralógica, listas, predicados sobre metaprogramación entre los que cabe destacar el predicado =.., corte, etc.

Con SldDraw, el profesor puede seleccionar ejercicios en el que se signifiquen las diferencias que se producen al cambiar de regla de búsqueda, de regla de selección, al reordenar los objetivos, al realizar distintos usos de un predicado, etc. Por otro lado, al alumno le vale como test para comprobar que entiende los mecanismos que comporta la realización de árboles SLD y en particular, el modelo de ejecución de Prolog, así como el uso de ciertos predicados específicos de este lenguaje de programación o el efecto del corte.

SldDraw no incluye un intérprete de Prolog eficiente sino simplemente un simulador. Por tanto, no es posible realizar trazas de programas

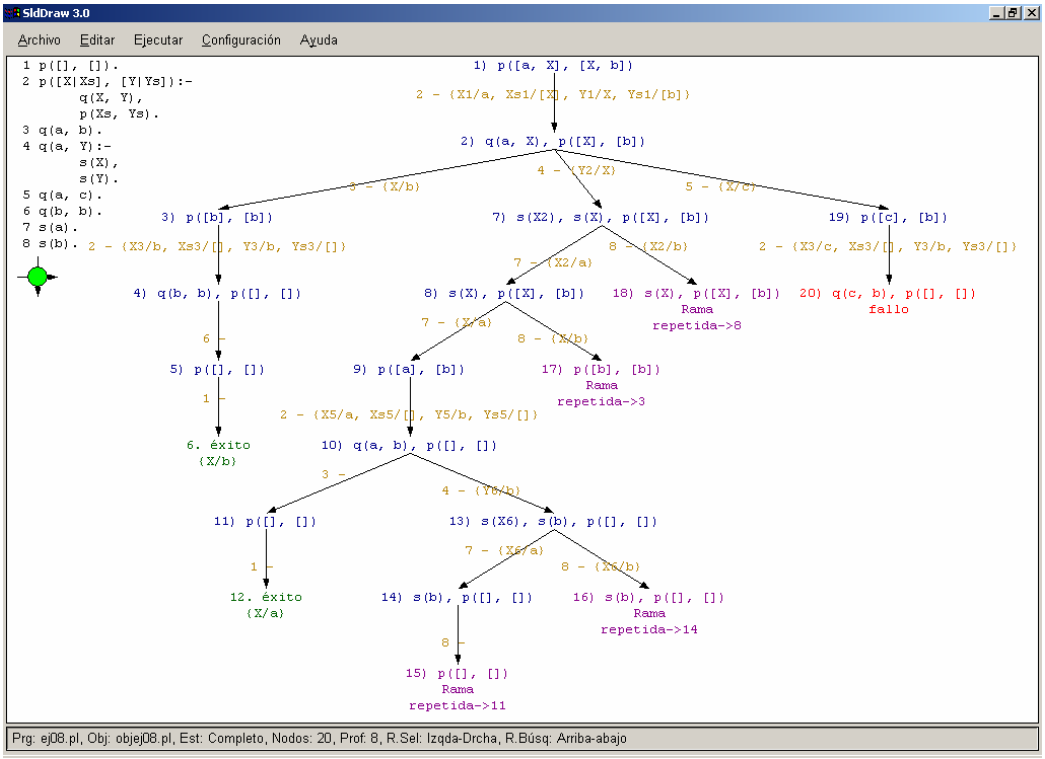


Figura 1. Ejemplo de ejecución de SldDraw

Prolog que generen una gran cantidad de nodos. Como ejemplo puede decirse que un árbol con 600 nodos y profundidad 40 se consigue visualizar en un tiempo razonable.

2. Descripción

Se trata de un trazador de árboles de ejecución SLD que muestra el árbol SLD y permite manipularlo hasta conseguir una representación adecuada. Al contrario del modelo de ejecución de Prolog, el árbol se genera por niveles y, dado que un árbol puede tener ramas infinitas, se fija inicialmente un nivel máximo de desarrollo pudiendo en cada momento modificarse. Un círculo verde o rojo nos indica claramente si el árbol está completo o no.

La aplicación admite como programas y objetivos a un subconjunto de los programas Prolog lo suficientemente amplio para que sea de utilidad. En particular, no incorpora características de entrada de datos y sólo incorpora los predicados básicos de salida. Está desarrollada en SWI-Prolog con XPCE [4] y este hecho resulta ser un aliciente para los alumnos que observan como el lenguaje que están aprendiendo es válido para la realización de aplicaciones de cierta complejidad. Existen versiones para Windows y para Linux: en realidad, esta es una facilidad de SWI-Prolog. Tiene un entorno similar al de cualquier aplicación actual basada en ventanas. Por otro lado, también se ha desarrollado una utilidad Java, llamada JsldDraw[5] bastante parecida a SldDraw que permite ser ejecutada como Applet y como aplicación.

Cada elemento del árbol es mostrado en un color diferente permitiendo distinguirlos y detectar fácilmente los distintos tipos de nodos; por ejemplo, un nodo fallo aparece en rojo y un nodo éxito en verde.

Se permite también modificar el programa o el objetivo mediante un editor que puede ser seleccionado por el usuario.

La aplicación dispone de un fichero de configuración en el que entre otras cosas, se permite modificar los colores de los nodos y unificaciones, el idioma de la aplicación, el editor utilizado, etc.

El algoritmo para la representación gráfica ha sido adaptado del propuesto para la representación de árboles en ML [6].

3. Ejecución

Para comenzar debemos disponer de un programa en Prolog que esté libre de errores y almacenado en un fichero de extensión *.pl.* Si el programa contiene errores, y se intenta abrir desde la aplicación se muestra una ventana de aviso indicando el error producido pero en la que la descripción del error no está suficientemente clara. A continuación se debe disponer de otro fichero con la misma extensión en el que estará escrita la consulta que queremos realizar.

Ya desde SLDdraw abrimos ambos ficheros, el que contiene el programa y el de la consulta y pedimos a la aplicación que nos represente el árbol. Otras opciones nos permiten iniciar el proceso de representación paso a paso.

Tanto el programa como el objetivo pueden modificarse desde la aplicación generando árboles para cada caso. Podemos imprimir el árbol resultante. En este caso podemos ajustar el número de páginas horizontales y verticales que ocupará la representación. Esto permite imprimir árboles cuya representación exceda del tamaño de una página. Además, podemos obtener un fichero encapsulado postscript con la representación.

Ya se ha mencionado que la herramienta no es un intérprete de Prolog. A diferencia de Prolog la aplicación trabaja con chequeo de ocurrencias (occur check) de manera que no produce unificación cuando esta situación se detecta. Por otro lado, detecta ramas infinitas y ramas cíclicas siendo la detección de estas últimas configurable.

4. Ámbito de utilización

Es una herramienta muy útil en las clases de prácticas de las asignaturas de programación lógica para el aprendizaje del lenguaje Prolog y en general de la resolución SLD. Desde el programa más simple se puede comprobar el árbol que genera mostrando unificaciones y respuestas computadas.

Dado que el aprendizaje de un lenguaje de programación declarativo como es Prolog suele resultar complicado, los alumnos agradecen disponer de una utilidad de estas características.

Es una aplicación relativamente pequeña y sencilla de aprender y utilizar. La distinción de los elementos que participan en la ejecución mediante colores permite desde el primer momento al alumno identificar cada parte y cada paso.

El hecho de que las unificaciones que se realizan aparezcan escritas en las ramas también hace más rápida su comprensión y aprendizaje. Así mismo, al desplegarse, aparece una ventana con la descripción paso a paso del proceso de obtención del unificador. Esto también ayuda a la comprensión del algoritmo de unificación de Robinson.

En la actualidad se está utilizando con éxito en dos universidades, la de Málaga y la de Cádiz. Así mismo, sabemos que se está utilizando también en algunas universidades de Centroamérica.

En particular, en Málaga se utiliza desde hace dos años, tanto en la Ingeniería Informática como en las Ingenierías Técnicas y aunque no se ha realizado ningún test para comprobar la ventaja que ha supuesto el uso de la herramienta, los alumnos, y sobre todo los repetidores que no la habían usado en años anteriores, han manifestado su reconocimiento por poder contar con una herramienta de estas características. Según sus comentarios, y en los laboratorios se pone de manifiesto, usan la aplicación a menudo para aclarar significados, variaciones según las formas de uso, etc. Con respecto al profesorado, ha aumentado la variedad y diversidad de ejercicios relativos a la

creación de árboles SLD tanto en las prácticas como en los exámenes.

En Cádiz, es el primer año que los alumnos de Ingeniería Técnica en Informática de Gestión están utilizando la herramienta. Se les ha pasado una encuesta al final del cuatrimestre y, en su mayoría, les parece muy positivo contar con una aplicación complementaria al intérprete que les facilite el aprendizaje. Les resulta fácil de manejar, y la consideran práctica para detectar errores y depurar los programas. También están de acuerdo, en su mayoría, en que comprenden mejor el método de Resolución SLD y la forma en que se ejecutan los programas en Prolog gracias a la aplicación.

En realidad el ámbito de utilización se podría extender hacia cualquier persona que necesite programar en Prolog .

5. Problemas o dificultades que resuelve

A continuación enumeramos las dificultades que a nuestro entender resuelve la herramienta:

- Dificultad inicial para comprender este tipo de programación por parte de personas que siempre (o en la mayoría de los casos) han programado y aprendido a programar en lenguajes basados en otro paradigma (imperativo, etc.).
- Comprensión de que la ejecución de programas en Prolog se realiza mediante el método de resolución SLD. Es decir, la búsqueda de soluciones se realiza en *búsqueda primero en profundidad con retroceso* sobre el árbol, una vez fijadas las reglas de selección y de búsqueda. Esto aclara de inmediato el porqué de la no completitud de Prolog.
- La depuración de programas mediante esta traza gráfica es mucho más clara que la depuración tradicional sobre programas lógicos.
- Aprendizaje del efecto de los predicados de control, en especial del corte.

6. Ventajas

Comentamos a continuación, algunos elementos de la aplicación que hacen atractivo su uso y extensible a otros ámbitos:

- Facilidad de instalación y uso, así como su reducido espacio.
- Posibilidad de configurar el idioma.
- Existencia de versiones para varios sistemas operativos.
- Inclusión de numerosos ejemplos.
- Posibilidad de imprimir o guardar el árbol.
- La modificación de la visualización permite obtener el mismo árbol de diferentes formas.

7. Conclusiones y posibles mejoras

En definitiva es una herramienta útil y didáctica dentro del ámbito de la Programación Lógica, que sirve tanto de apoyo al docente en su trabajo como al alumno en la tarea de aprender y asimilar.

Para el futuro se pretende dotar a la aplicación de un analizador sintáctico que resuelva errores en la construcción de programas y objetivos, la posibilidad de no desarrollar trozos del árbol que no sean significativos y la posibilidad de abrir ventanas con subramas (contemplada en JsldDraw).

8. Referencias

[1]J.W. Lloyd. Foundations of Logic Programming. Second, Extended Edition. Springer-Verlag, 1987.

[2] D. Poole, A. Mackworth y R. Goebel. Computational Intelligence. A Logical Approach. Oxford University Press, 1998.

[3]Antonio Barranquero Campos. Visualización de árboles SLD. Proyecto Fin de Carrera. Dpto. de Lenguajes y Ciencias de la Computación. Universidad de Málaga. 2002.

[4] Jan Wielemaker. SWI-Prolog/XPCE. University of Amsterdam.

[5]Antonio Jesús Paredes García. JApplet para visualizar árboles de refutación SLD. Proyecto Fin de Carrera. Dpto. de Lenguajes y Ciencias de la Computación. Universidad de Málaga. 2002.

[6] Andrew J. Kennedy. Drawing Trees. Journal of Functional Programming, Cambridge University Press, Mayo 1996.

Sigraf: SImulador de GRAFos.

Judith Antolín Sendino
María Ruiz Ruiz
Escuela Politécnica Superior,
Universidad de Burgos
{jas00, mrr0012}@alu.ubu.es

Carlos Pardo Aguilar
Juan J. Rodríguez Díez
Dpto. de I. Civil, Lenguajes y Sistemas Informáticos
Universidad de Burgos
{cpardo, jjrodriguez}@ ubu.es

Resumen

En el presente artículo se presenta una herramienta de apoyo al aprendizaje en un campo tan importante de la informática como las Estructuras de Datos, centrándose en el diseño, desarrollo y aplicación del Tipo Abstracto de Datos (TAD) grafo para la resolución de diversos tipos de problemas.

Aunque se encuentran disponibles diversas animaciones de algoritmos sobre grafos, normalmente éstas se limitan a un solo algoritmo (e.g., Dijkstra), o a un solo problema (e.g., árboles expandidos mínimos). La posibilidad de introducir un grafo y poder ejecutar interactivamente toda una gama de algoritmos sobre el mismo, es claramente ventajosa con respecto a tener que introducir el mismo grafo para diversos algoritmos, utilizando una interfaz diferente para cada uno de ellos.

1. Introducción

La disponibilidad de herramientas que permitan al alumno trabajar con conceptos adquiridos en las clases teóricas, así como resolver dudas, es escasa, además las horas de prácticas de las que se dispone en la universidad son limitadas. Por ello, es importante la disponibilidad de herramientas que proporcionen un entorno de pruebas y desarrollo de las diferentes cuestiones que van aprendiendo, que redunde en un mayor aprovechamiento de las horas dedicadas a clases prácticas.

Se considera de especial interés el estudio de los grafos que se usan para modelar cualquier situación en la que existan elementos unidos con otros tales como redes de alcantarillado, redes de comunicación, circuitos eléctricos, entre otros. Una vez modelado el problema mediante un grafo se pueden hacer estudios sobre diversas propiedades, para ello se utilizan algoritmos concretos que resuelvan ciertos problemas. La teoría de grafos ha sido aplicada en el estudio de problemas que surgen en áreas diversas de las ciencias, como la química, la ingeniería eléctrica o la investigación operativa. El primer paso siempre será representar el problema como un grafo. En esta representación cada elemento, cada objeto del problema se representa mediante un nodo. La relación, comunicación o conexión entre los nodos da lugar a una arista, que puede ser dirigida o bidireccional (no dirigida) [1].

En el ámbito de asignaturas de Estructuras de Datos, que tienen gran carga teórica y aplicación práctica, se pueden encontrar demostraciones de algunos algoritmos sueltos [3], [6]. Sin embargo, no se ha encontrado disponible ninguna herramienta que abarque aquéllos que normalmente se incluyen dentro de estas asignaturas. La utilización de una herramienta distinta para cada algoritmo es incómoda, puesto que cada una de ellas tiene su peculiar manera de editar un grafo. Por ello se planteó el desarrollo de una aplicación para el estudio de este TAD y de varios de sus algoritmos. Aunque sería deseable disponer de un entorno de visualización de todas las estructuras de datos habituales, entendemos que la cantidad de algoritmos que se pueden realizar sobre grafos justifica una herramienta para su estudio. Por otro lado, dentro de las estructuras de propósito general, la de grafo es la más compleja, así que se considera que es el primer paso más adecuado hacia un sistema de animación de cualquier estructura de datos.

El resto del artículo se organiza del siguiente modo. La sección 2 presenta los objetivos planteados. La herramienta se presenta en la

sección 3. Finalmente se exponen las conclusiones en la sección 4.

2. Objetivos

El objetivo era crear una aplicación que permitiera diseñar grafos, guardarlos, modificar otros ya existentes, imprimirlos y ver de forma didáctica y atractiva los distintos algoritmos que se estudian sobre ellos. Todo ello de forma gráfica y fácil de utilizar para ayudar en su comprensión a los alumnos.

La herramienta se ha desarrollado utilizando el paradigma Orientado a Objetos, ya que facilita la reutilización de las aplicaciones y es probable que esta herramienta en un futuro sea ampliada en posteriores proyectos finales de carrera u otros trabajos de investigación

Uno de los aspectos que se tuvo en cuenta fue la necesidad de diferenciar entre dos modos de funcionamiento del grafo, por un lado el grafo abstracto que es el que se encarga de modelar la realidad y sobre el que se ejecutan los algoritmos, y por otro lado las propiedades gráficas para poder representarlos en la pantalla (posición, color, etc.).

Como consecuencia se generaron dos clases para cada modo de funcionamiento, con las relaciones mostradas en la Figura 1.

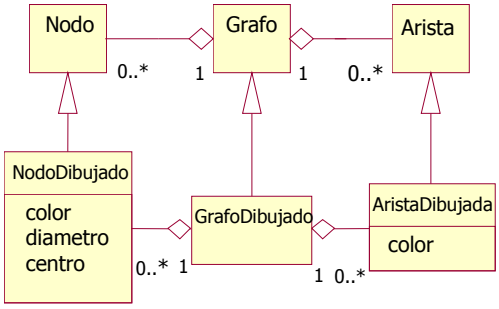


Figura 1: Diagrama de clases

3. La herramienta

Para la realización de esta aplicación se ha estudiado el comportamiento de otros programas en situaciones específicas, intentando obtener un producto con el que el usuario estuviera familiarizado y su uso fuese lo más intuitivo posible, e intentando adaptar su modo de trabajo y sus funcionalidades a un entorno como el de la enseñanza, así como mejorar los aspectos más importantes. Se consideró relevante incrementar al máximo la disponibilidad y portabilidad de la aplicación, estando disponible ésta en la página web de la asignatura para que los alumnos puedan descargarla cuando deseen, y pudiendo ser ejecutada en distintos sistemas operativos, ya que el hecho de estar implementada en Java, lenguaje robusto, seguro y de gran portabilidad ayuda a la obtención de estas características.

La aplicación consta de una interfaz gráfica sencilla y fácil de manejar, que permite al usuario trabajar con esta estructura de datos de forma rápida y eficaz (ver Figura 2). Sobre este aspecto se ha de comentar uno de los requisitos más relevantes, la facilidad y comodidad de manejo que incluye que todas las operaciones de la aplicación se podrán realizar tanto por ratón como por teclado. Se debe tener en cuenta que si además del temario de la asignatura y de una herramienta específica, el alumno debe aprender a trabajar con otra adicional, ésta debe intentar facilitar su trabajo sin añadir complejidad.

Desde el menú principal, se puede acceder en cualquier momento a la ayuda de la aplicación para obtener información sobre cualquier aspecto relacionado, tanto con la forma de diseñar un grafo como con la aplicación de los algoritmos con la herramienta, además de las explicaciones teóricas sobre la estructura para facilitar su aprendizaje. Para todas estas funcionalidades se proporciona una barra de herramientas con botones significativos para cada opción (ver Figura 3).

Durante todo el proceso de desarrollo y modificación del grafo se realizan diversas comprobaciones, evitando por ejemplo que se dibujen nodos solapados, que antes de poder dibujar una arista existan los nodos que esta desea unir, que cuando se borra un nodo se borren todas sus aristas asociadas. En la barra de estado de la aplicación se indican mensajes de error sobre las comprobaciones anteriores, se visualizan las coordenadas del puntero de ratón en todo momento y si se selecciona un elemento del grafo pueden obtenerse sus propiedades.

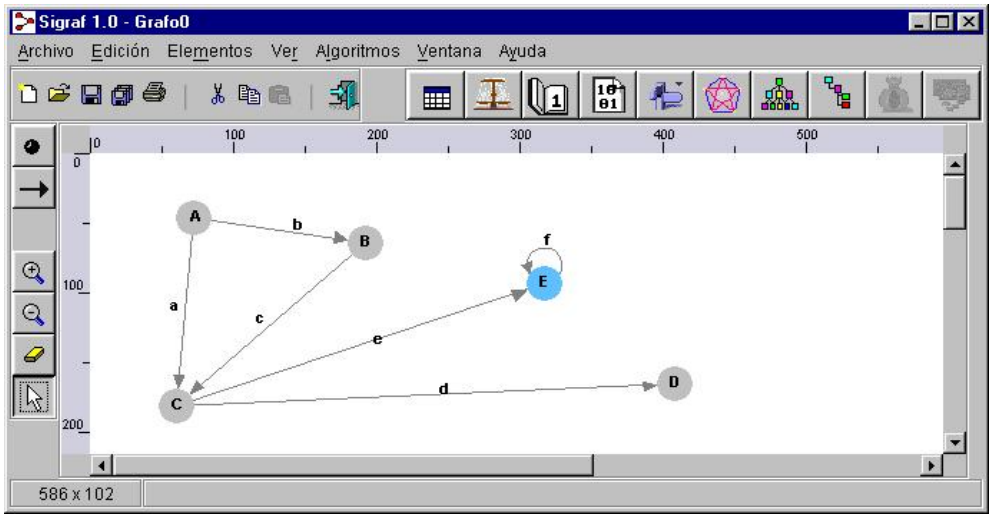


Figura 2: Interfaz de la aplicación.

Figura 3 Barra de herramientas de Dibujo

3.1. Representaciones de los grafos

Dentro de las diferentes visualizaciones de los grafos se encuentran la matriz de adyacencias, la matriz de costes y la lista de adyacencias. Cada una de estas puede obtenerse a través del menú *'Ver'* de la aplicación, desde el cual también podemos configurar qué propiedades de las aristas del grafo se desean mostrar y obtener más información sobre cada uno de los nodos, del grafo.

3.2. Ejecución de los algoritmos sobre grafos

La ejecución y visualización de los algoritmos, igual que el resto de opciones de la herramienta, se puede realizar a través del menú o desde la barra de herramientas específica de estos. Dichas visualizaciones se realizan en una ventana auxiliar, bien de forma gráfica mostrando los cambios mediante colores sobre el propio grafo, bien con los datos obtenidos(matrices de caminos, caminos mínimos…) o con ambas cosas según el algoritmo.

En la mayoría de ellos puede realizarse su ejecución paso a paso para seguir la evolución del algoritmo de forma más didáctica para el alumno.

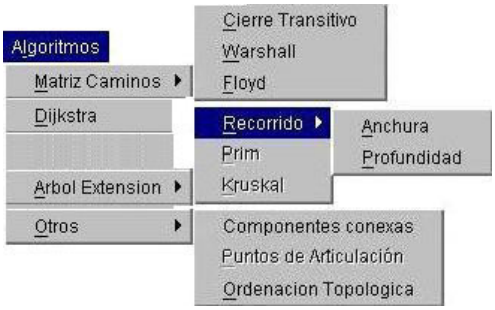


Figura 4 Menú de algoritmos.

3.3. Tratamiento de ficheros

Para el almacenamiento de los grafos se generan ficheros de texto con un formato específico y extensión ".grf" para identificarlos. Las operaciones sobre éstos (crear, abrir, cerrar,

guardar...), se realizan a través del menú *'Archivo'* o desde la barra de herramientas estándar. Al permitir tener varios ficheros abiertos a la vez, se lleva un control de los que han sido modificados y se proporciona la posibilidad de cambiar de uno a otro desde el menú *'Ventana'*.

3.4. Ejemplo de ejecución de un algoritmo.

El *cálculo de las componentes conexas* de un grafo consiste en obtener aquellos subconjuntos de nodos entre los cuales la comunicación es total. En este caso no se proporciona la opción de realizarlo paso a paso, ya que no se cree necesario para que el alumno comprenda el desarrollo del mismo. Un grafo no dirigido G es conexo si existe un camino entre cualquier par de nodos que forman el grafo. En el caso de que el grafo no sea conexo se pueden determinar todas las componentes conexas del mismo

En un grafo dirigido podemos distinguir entre grafo dirigido conexo y grafo fuertemente conexo. Un grafo dirigido es conexo si para cada par de vértices existe una cadena que los une. Y un grafo dirigido es fuertemente conexo si para cada par de vértices existe un camino que los une. Por tanto, para saber si un grafo es o no conexo se calculan sus componentes, si tras visitar todos sus nodos se obtienen una única componente se dice que el grafo es conexo.

La visualización realizada para este algoritmo consta de una pantalla dividida en dos paneles. En el de la izquierda se visualiza el grafo formado por las componentes conexas obtenidas, mientras que en la parte derecha se muestran los nodos que forman la componente elegida o todo el grafo. Para facilitar la comprensión al alumno, se realiza una asociación de colores usando el mismo para una componente en el panel de la izquierda, que para los nodos por los que está formado en la parte derecha.

4. Conclusiones y trabajos futuros

Debido a que la herramienta se ha desarrollado recientemente y dada la situación temporal del tema dedicado a grafos al final de la asignatura de Estructuras de Datos, es demasiado pronto como para establecer conclusiones respecto a su utilidad. No obstante, se considera que aunque sólo sea como "transparencias interactivas" la herramienta es útil. Queda por determinar hasta que punto los alumnos utilizan la herramienta por su cuenta, y su valoración de la misma.

Dentro de las posibles ampliaciones, algunas de las cuales ya están en marcha, se pueden comentar las siguientes. Por un lado, las mejoras en la capacidad de edición y visualización, como mostrar la traza del algoritmo además del grafo, permitir arcos curvados o colocar automáticamente los nodos y arcos. Para este último aspecto se planea utilizar alguna herramienta como Graphviz [4]. El formato de fichero utilizado para almacenar los grafos es actualmente un formato propio, por lo que parece interesante incluir la posibilidad de importar y exportar a otros formatos, por ejemplo GraphML [2]. Aunque en la versión actual están disponibles bastantes algoritmos, este apartado se puede completar, con por ejemplo, cuestiones relativas a flujo en redes o problemas NP.

Referencias

[1] Mark Allen Weiss "Estructuras de datos en Java". Adisson-Wesley.

[2] U. Brandes, M. Eiglsperger, I. Herman, M. Himsolt, and M.S. Marshall: GraphML Progress Report: Structural Layer Proposal. Proc. 9th Intl. Symp. Graph Drawing (GD'01), LNCS 2265, pp. 501-512. Springer-Verlag, 2002.

[3] Peter Brummund, Ngozi V. Uti. "The Complete Collection of Algorithm Animations". http://www.cs.hope.edu/~alganim/ccaa/

[4] Emden R. Gansner "Drawing graphs with GraphViz" 2003. http://www.research.att.com/sw/tools/graphviz/libguide.pdf

[5] Luis Joyanes Aguilar, Ignacio Zahonero Martinez "Estructuras de datos. Algoritmos, abstracción y objetos". Mac Graw Hill 1999.

[6] M. Gloria Sánchez Torrubia, Víctor M. Lozano Terrazas. "Algoritmo de Dijkstra. Un tutorial interactivo". VII Jornadas de Enseñanza Universitaria de la Informática (JENUI 2001)

Autoevaluación a través de Internet por medio de test

Pedro A. Castillo Alberto Prieto Antonio Cañas Beatriz Prieto

Dpto. de Arquitectura y Tecnología de Computadores
Universidad de Granada
18071 Granada
e-mail: pedro@atc.ugr.es

Resumen

En este trabajo se presenta un sistema automático de *generación* y *corrección* de exámenes tipo test (SAGET). El sistema selecciona aleatoriamente de una base de datos un conjunto de preguntas tipo test, con varias opciones de respuesta; tanto el número como su índice de dificultad son elegidos por el usuario. Una vez que el alumno contesta el test, éste es corregido automáticamente, y se muestra nuevamente en pantalla el cuestionario destacando tanto las respuestas del usuario como las respuestas correctas. El sistema es útil en una doble vertiente: para que el alumno se autoevalúe desde cualquier lugar donde tenga acceso a Internet, y para que el profesor genere automáticamente cuestionarios de test para examinar a sus alumnos.

1. Introducción

Los tests se presentan como una herramienta de gran utilidad para facilitar el aprendizaje, y de hecho han sido utilizados ampliamente y desde sus orígenes en numerosos sistemas CAI (*Computer Assisted Instruction*) o CAL (*Computer Aided Learning*).

Por otra parte, una de las tareas más tediosas de la enseñanza es la evaluación de los alumnos. Por lo general, la evaluación de las asignaturas de materias científico-técnicas se efectúa por medio de ejercicios de *problemas* y por *prácticas* de laboratorio, complementados con ejercicios de test. Los exámenes tipo test presentan una gran dificultad en su diseño [1] (elección de las preguntas y las opciones de respuestas posibles) ya que implican seleccionar las cuestiones más adecuadas a los objetivos de medición establecidos; sin embargo, se adaptan muy bien a una corrección completamente objetiva y automática, ventajas que no se obtienen con otras modalidades de exámenes.

En la bibliografía podemos encontrar trabajos en los que se proponen sistemas para corrección automática de exámenes tipo test [2], sin embargo, este trabajo se centra en conseguir una mayor versatilidad y comodidad de uso del sistema, realizándose todas las fases de forma automática, y en la introducción de un índice de dificultad asociado a cada cuestión.

1.1. Objetivos del trabajo

Los principales objetivos y características de SAGET son:

- Generación automática de exámenes de tipo test para cualquier asignatura.
- Selección por parte del usuario del número de preguntas que componen el test.
- Selección de un nivel de dificultad por parte del alumno o profesor.
- Corrección y evaluación automáticas *on-line*, bien para realizar exámenes reales, o como apoyo al estudio realizando el alumno ensayos a través de Internet (autoevaluación).
- Generación de la plantilla de corrección que ayude al profesor en una corrección manual.
- Adaptación de la dificultad de cada pregunta de acuerdo al número de veces que se haya contestado correcta, incorrectamente, o se haya dejado sin contestar.

Todo el sistema se ha desarrollado de forma que sólo hace falta una conexión a Internet para acceder al servidor donde se encuentra SAGET y tener un navegador en el que visualizar los documentos HTML.

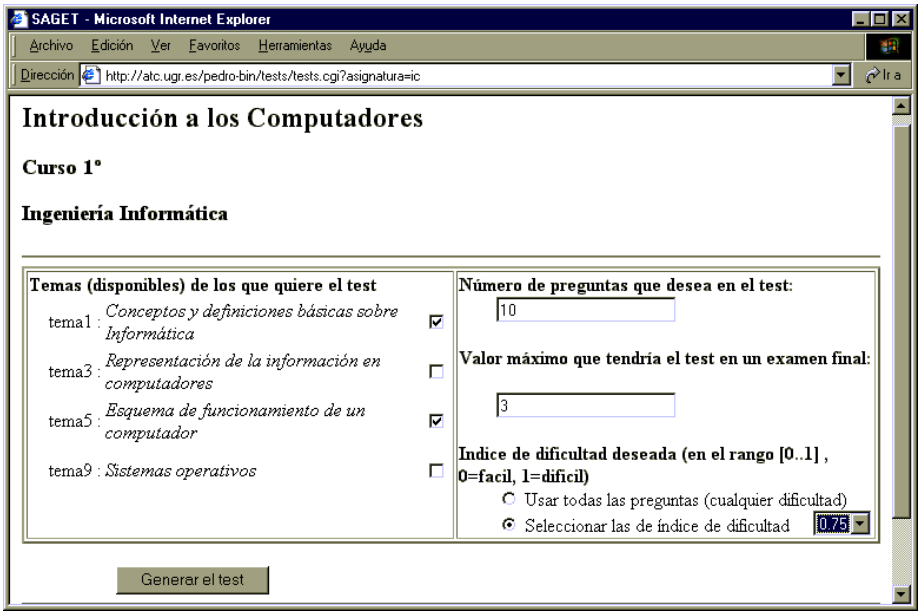


Figura 1. Formulario en el que seleccionaremos los temas de los que queremos realizar el test, el número de preguntas y la dificultad de éstas.

A continuación describiremos en detalle el funcionamiento del sistema y la implementación llevada a cabo.

2. Descripción del funcionamiento del sistema

El funcionamiento esquemático del sistema queda descrito en los siguientes puntos:

1. El estudiante, a través de la página principal del sistema selecciona la asignatura sobre la que desea el test.
2. El sistema obtiene ciertos datos sobre esa asignatura para mostrar un primer formulario HTML (Figura 1) en el que se solicitan los temas a incluir en la evaluación, el número de preguntas en el test, y el nivel de dificultad (de 0 a 1) deseados. A continuación, el estudiante podrá pedir la generación del test y pasar a resolverlo.
3. El sistema recibe el número de preguntas y la dificultad de éstas, y extrae de los temas especificados, de forma aleatoria, las preguntas que se ajusten a dicha dificultad. Devuelve al navegador cliente un formulario HTML con las preguntas y botones asociados a cada una de las cuatro respuestas posibles (Figura 2).
4. Una vez contestadas, el servidor recibe las respuestas y comprueba la corrección o incorrección de cada una de ellas.
5. El servidor vuelve a presentar el test, pero ahora destacando en color rojo las respuestas incorrectas dadas por el usuario y en verde las respuestas correctas (Figura 3). Con esta revisión personalizada el estudiante tiene la oportunidad de conocer qué preguntas son correctas y aprender de sus errores.

3. Implementación del sistema

SAGET se ha concebido como una herramienta que pueda ser utilizada por profesores de distintas asignaturas y titulaciones (no sólo de Informática).

A cada titulación se le asocia una base de datos distinta, dentro de cada una de las cuales se incluyen carpetas para las diferentes asignaturas. Cada carpeta de una asignatura se compone de diferentes archivos, uno de ellos describe la

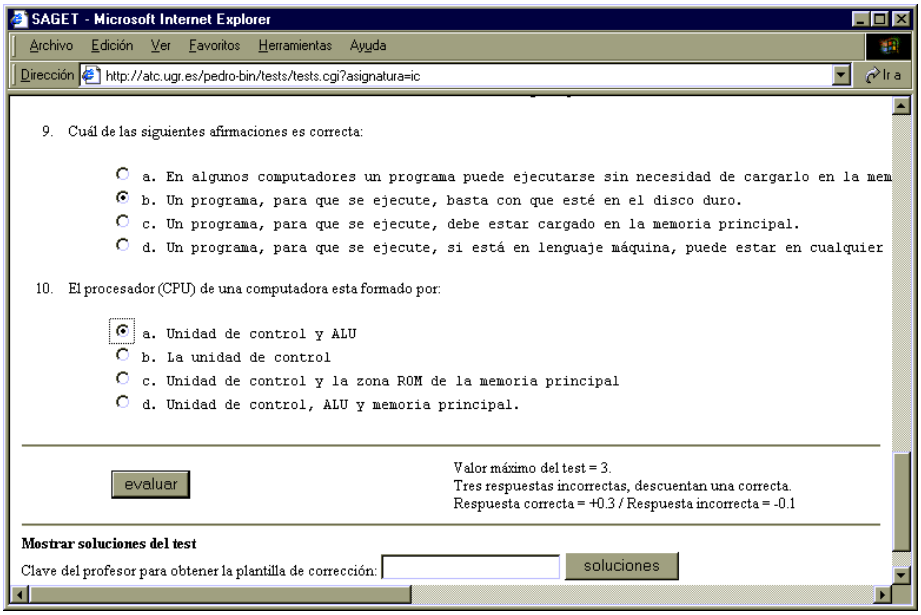


Figura 2. Formulario que muestra las preguntas del test (cuatro opciones por pregunta) y las opciones para evaluar las contestaciones o para generar la plantilla de corrección (opción para el profesor).

asignatura (curso, titulación, número de temas disponible) y los otros corresponden a cada uno de los temas de las asignaturas incluyendo las distintas cuestiones de cada uno de ellos (enunciado, opciones, contestación correcta, e índice de dificultad). Para añadir un tema a una asignatura, simplemente se añade un nuevo archivo, con las cuestiones de dicho tema, a la carpeta correspondiente a dicha asignatura.

Otra característica destacable de SAGET es poder especificar el nivel de dificultad deseado para el test. Así, según la preparación del estudiante (o del nivel que se requiera en el examen) se buscarán preguntas más o menos difíciles (que hayan sido contestadas más veces incorrectamente o se hayan dejado sin contestar).

Para establecer el índice de dificultad de un test, el sistema asocia a cada cuestión un índice de dificultad, que varia entre 0 y 1, y que es establecido y adaptado continua y automáticamente por el sistema, en función de las contestaciones recibidas previamente por el sistema. El índice de dificultad de una cuestión tiene el valor 0 (fácil) al principio, y se actualiza cada vez que es planteada, de acuerdo con la siguiente expresión:

$$i = \frac{n_{erroneas}}{n_{planteadas}}$$

Donde $n_{erroneas}$, y $n_{planteadas}$ son, respectivamente, el número de veces que dicha cuestión ha sido contestada erróneamente y el número de veces que el sistema la ha planteado a los alumnos. El sistema elige las cuestiones aleatoriamente pero tratando de que el índice de dificultad media de las cuestiones propuestas se aproxime al máximo al deseado por el usuario.

La puntuación del test se lleva a cabo con una corrección estadística, tratando de que si se contestase aleatoriamente a las cuestiones, la puntuación media fuese 0. Así, la ecuación que calcula la puntuación final, p, del test es:

$$p = \left(N_{correctas} - \frac{N_{incorrectas}}{3} \right) \times \frac{V_{\max}}{N_{cuestiones}}$$

donde V_{max} es la puntuación máxima del test, $N_{cuestiones}$ es el número total de cuestiones incluidas en el mismo, y $N_{correctas}$ y $N_{incorrectas}$ son el número total de cuestiones contestadas correcta e incorrectamente.

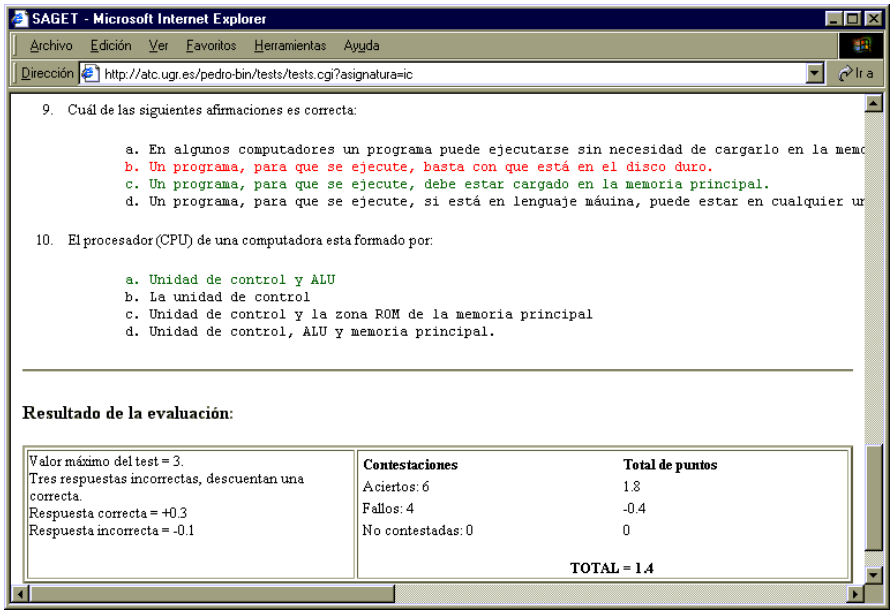


Figura 3. Página de evaluación en la que se muestran las correcciones a las contestaciones incorrectas. Las estadísticas muestran el total de puntos por los aciertos, fallos y no contestadas.

El sistema ha sido implementado teniendo en cuenta su portabilidad y facilidad de ampliación con nuevas asignaturas. Por ello se usa un programa CGI escrito en Perl. En la actualidad se utiliza un conjunto de archivos de texto para almacenar las preguntas, aunque está desarrollándose una base de datos relacional y la interfaz necesaria para mantenerla más fácilmente.

El sistema se está aplicando a la asignatura de Introducción a los Computadores, impartida en diversas titulaciones de la Universidad de Granada, y que sigue el texto indicado en la referencia [3]. Las preguntas que conforman la base de datos actual han sido diseñadas por los profesores que imparten dicha asignatura.

4. Conclusiones y trabajo futuro

Se ha descrito un nuevo sistema totalmente automático de generación de test, con preguntas y respuestas seleccionadas aleatoriamente de una base de datos, pudiendo el usuario seleccionar los temas a incluir, el número de preguntas, el índice de dificultad y la puntuación máxima dada al test. SAGET se ha proyectado como ayuda para el estudio, de forma que los estudiantes puedan autoevaluarse desde cualquier ordenador con acceso a Internet, y como ayuda al profesor para el diseño de exámenes de test. El sistema ha tenido una buena acogida entre los alumnos que lo han utilizado ampliamente para preparar la asignatura en el primer cuatrimestre.

Como mejoras al sistema, está previsto añadir nuevas asignaturas y, algo que ya está siendo implementado, añadir el módulo de identificación de los alumnos para examinarlos en aulas de PC conectados a Internet.

Referencias

[1] J. Muñiz, *Teoría clásica de los test,* Pirámide, 1998.

[2] N. Pavón, José R. Cano, F. Márquez, A. Sainz. SCRAE'Web: *Sistema de Corrección y Revisión Automática de Exámenes a través de la WEB.* JENUI 2002, pp.231-235. 2002.

[3] A. Prieto, A. Lloris, J. C. Torres, *Introducción a la Informática,* 3ª Ed., McGraw-Hill, 2002.

Una Componente "e-Learning" de Aprendizaje Colaborativo para el Proyecto IDEFIX

T. Hernán Sagástegui Ch., José E. Labra G., Juan M. Cueva L.
José M. Morales G., María E. Alva O., Eduardo Valdés, Cecilia García

Departamento de Informática
Universidad de Oviedo
C/ Calvo Sotelo S/N,
33007 Oviedo
e-mail: thsagas@correo.uniovi.es, labra@lsi.uniovi.es, cueva@lsi.uniovi.es, jmmoral@correo.uniovi.es, malva360@correo.uniovi.es, eduardovr@telecable.es, cgpelayo27@hotmail.com

Resumen

Este artículo presenta el diseño y la arquitectura de implementación de un modelo de aprendizaje colaborativo a través de Internet. El sistema dará soporte a una experiencia piloto que se está desarrollando en la asignatura de lógica de la EUITIO de la Universidad de Oviedo, permitiendo el desarrollo colaborativo de ejercicios en "e-reuniones" a grupos de alumnos y el entrenamiento en la resolución de exámenes mediante un juego de realidad virtual. Se basa en la utilización de servicios Web como una componente del proyecto IDEFIX.

1. Motivación

El proyecto IDEFIX (*Integrated Development Environment Frameworks based on Internet and eXtensible technologies*) se centra en el desarrollo de entornos integrados de desarrollo a través de Internet [1]. IDEFIX puede extender su dominio a la enseñanza por Internet de cualquier asignatura de la Escuela Universitaria de Ingeniería Técnica de la Universidad de Oviedo EUITIO, así como a la enseñanza compartida de asignaturas de libre elección a través de Internet del proyecto AulaNet [2]. En este contexto, en este artículo se presenta una componente del proyecto IDEFIX, referida al diseño y arquitectura de un modelo de aprendizaje colaborativo, cuyo objetivo es apoyar la imparticion de la asignatura de lógica de la EUITIO a través de la Web.

Actualmente la asignatura de lógica se imparte en el primer año de la carrera de ingeniería informática de la EUITIO, las clases se dan presencialmente y como material pedagógico de apoyo se dispone de una guía didáctica y de un cuaderno que recoge los exámenes realizados en la asignatura los años anteriores que sirven como ejercicios.

Se desea que los alumnos de la asignatura de lógica puedan acceder de forma remota al cuaderno que recoge los exámenes /ejercicios y puedan resolverlos en grupo a través de la Web, bien desde los laboratorios disponibles en la facultad, o desde sus casas, con la ventaja de la libertad total de horario. Se tienen los siguientes requisitos:

a) Implementar el desarrollo colaborativo de los ejercicios del cuaderno de exámenes a través de la Web.

b) Motivar a los alumnos a realizar un entrenamiento para los exámenes. Para ello se ha propuesto la creación de una especie de juego mediante realidad virtual en Internet.

2. Antecedentes

El aprendizaje colaborativo asistido por ordenador (CSCL) apoya el trabajo en grupo de una tarea común, proporciona una interfaz compartida para

que el grupo trabaje en ella [3,4,12] y usa la tecnología de ordenadores como herramienta que ayuda a los aprendices a comunicarse y colaborar en actividades conjuntas, mediante una red de ordenadores, apoyando la coordinación, y la aplicación del conocimiento en cierto dominio [5,6]. CSCL se usa en el ambiente educativo y sirve de soporte a los estudiantes en el aprendizaje, facilitando el proceso de trabajo en grupo y la dinámica de grupos de una manera que no se lograría cara a cara [3].

Existen numerosas aplicaciones colaborativas según la respectiva taxonomía de aplicaciones y el tipo de actividad de aprendizaje que apoyan: tutoriales, resolución de problemas, simulaciones [9], debates, modelación y CourseWare (Blackboard: Virginia Commonwealth University, LearningSpace: Lotus & IBM [11], WebCT: University British Columbia, TopClass: WBT Systems [10], etc). Para el aprendizaje de la lógica existen proyectos como JAPE [7,8].

Una aplicación colaborativa dentro de un esquema común de trabajo en grupo posee las siguientes características: memoria grupal, roles, protocolos de colaboración y percepción [3,5]. La memoria grupal es el espacio común donde los miembros de un grupo almacenan información en forma ordenada referente al desarrollo de la actividad [3,5]. Este espacio es creado con la finalidad de proveer al grupo de un dispositivo efectivo de comunicación y es el resultado tanto del proceso de trabajo como del producto final conseguido por el grupo [3,5]. Un rol es un conjunto de privilegios y responsabilidades atribuidas a una persona o a un módulo del sistema (agente). Los roles entre los miembros del grupo pueden ser implícitos o explícitos, para hacer más eficiente y coordinado el logro de los objetivos [3,5]. Los protocolos de colaboración son las distintas maneras de interactuar de las personas consensuadas por el grupo. Son reglas que permiten a los individuos comunicarse entre sí de tal forma que cada uno pueda enviar y recibir señales comprensibles para los demás [3,5]. La percepción es toda información que provee una conciencia grupal al individuo que forma parte de un grupo. En el contexto del objeto de la información tenemos "percepción de usuarios" y "percepción de datos". La percepción de usuarios provee información sobre los miembros del grupo, informa de quiénes están conectados y lo que éstos hacen. La percepción de datos suministra información referente a los cambios efectuados sobre los datos [3,5].

3. Modelo de Aprendizaje Colaborativo

El modelo de aprendizaje colaborativo para el aprendizaje de la lógica está basado en el grupo y en las e-reuniones del grupo, en un entorno distribuido, pudiéndose diseñar un ciclo de vida de las e-reuniones [3,4,5,6]. Las componentes elementales del modelo de aprendizaje colaborativo de la lógica a través de la Web, son:

* **Memoria Grupal:** conformada por la base de datos de alumnos de la universidad (enlaces a asignaturas y profesores), la base de ejercicios / exámenes y la base de diseño de juegos.
* **Roles:** rol del profesor, rol del alumno que desarrolla ejercicios, rol del alumno que da examen, rol del usuario invitado.
* **Protocolos de Colaboración:** son las Reglas establecidas por el profesor, reglas del desarrollo de los ejercicios, reglas para rendir un examen, reglas para los usuarios invitados, reglas del administrador.
* **Percepción:** es la información generada en el desarrollo de las prácticas y de los exámenes; que los miembros del grupo conectados desarrollan, los resultados obtenidos, la generación de ideas, la toma de decisiones. Una posible herramienta para la percepción es el chat.
* **Interfaz:** el usuario interactúa con la memoria grupal y con el conocimiento generado a través de la interfaz, permitiendo la selección del trabajo siguiente; a) el desarrollo colaborativo de los ejercicios del cuaderno de exámenes a través de la Web o, b) la realización de los exámenes de entrenamiento a través de la Web.

4. Arquitectura del Modelo

La arquitectura para la implementación del modelo de aprendizaje colaborativo de la lógica a través de la Web está basado en el esquema cliente/servidor de la Figura 1. Se está implementando como una componente del proyecto IDEFIX sobre la plataforma .NET de Microsoft.

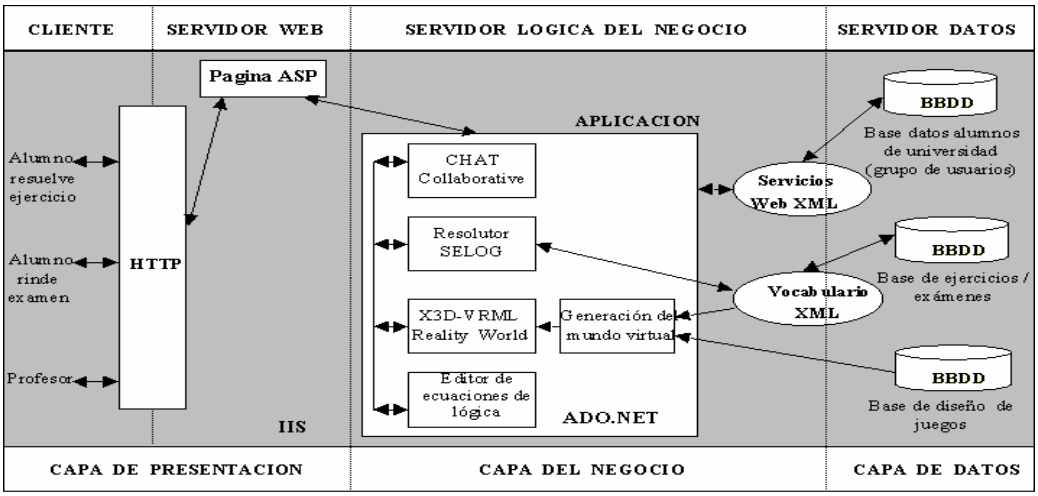


Figura 1. Arquitectura de implementación

4.1. Capa de presentación

Esta capa presenta la interfaz de usuario y se encarga de la tarea de visualización de los clientes y de la entrada de datos. La característica básica de esta capa es que el usuario (alumno, profesor, invitado) es un cliente navegador que utiliza el protocolo HTTP para acceder a la aplicación.

4.2. Capa del negocio

Encargada de la lógica de la aplicación y de la lógica de acceso a los datos y constituida por:

* **Editor de ecuaciones de lógica:** similar al que incorpora el procesador de textos Microsoft Word, y que facilita la creación de las ecuaciones de lógica para almacenar en un vocabulario XML.
* **CHAT colaborativo:** chat de consulta y retroalimentación.
* **Resolutor SELOG:** permite el desarrollo colaborativo de los ejercicios de lógica del cuaderno de exámenes a través de la Web. Presenta una interfaz y realiza funciones como; selección del tipo de ejercicios, obtención de pistas, consulta y evaluación de la dificultad de las preguntas, solución y comentarios. Al finalizar la sesión el usuario obtiene el número de preguntas resueltas correctamente y erróneamente. SELOG enlaza el CHAT colaborativo para la comunicación y retroalimentación del desarrollo de los ejercicios en grupo.
* **Reality World:** está diseñado para dar soporte en el entrenamiento de la realización de exámenes. Reality World es un producto de realidad virtual en X3D y presenta un mundo de objetos. El micro-mundo de la lógica es un mundo formado por esferas que describen los niveles del juego. Para pasar de una esfera a otra hay que aprobar todas las fases del nivel correspondiente. Cada nivel corresponde a un capítulo de la asignatura y en este nivel las fases se corresponden con los contenidos del capítulo. En un nivel determinado se incluyen todos los ejercicios hechos en los anteriores niveles; si no se han hecho todos los ejercicios de los distintos niveles no se podrá entrar en este último nivel. Respecto al resto de niveles, el jugador puede ir cambiando de nivel en nivel, siempre y cuando haya finalizado una fase o no haya comenzado la siguiente.

El documento XML es un documento en un vocabulario XML que define la configuración de la asignatura y modela los ejercicios. A partir del documento XML y los datos del mundo virtual leídos de las bases de datos, un programa

generador del mundo virtual, escrito en C#, permite pasar de XML a X3D generando el mundo en X3D.

4.3 Capa de Datos

En esta capa se encuentran los datos del alumno y sus relaciones con los datos de la asignatura y del profesor, almacenados en una base de datos. Asimismo está creada una base de datos que almacena todos los datos de los exámenes, preguntas y conceptos. También está creada una base de datos de diseño que almacena los juegos y los datos que el generador del mundo virtual necesite.

5. Conclusiones y Trabajos Futuros

La componente de aprendizaje colaborativo del Framework IDEFIX aun esta en desarrollo y actualmente se tiene un prototipo de prueba (http://idefix.sourceforge.net) que se utilizará como apoyo a la enseñanza de la lógica. Para la observación de los beneficios se clasificará a los estudiantes en grupos de usuarios y en grupos de no usuarios y se comparará su rendimiento.

Este trabajo busca contribuir al desarrollo de aplicaciones colaborativas para la educación virtual y "e-learning" en la educación secundaria y superior, satisfaciendo necesidades edu-comunicativas y pedagógicas propias [13]. Asimismo busca acuñar y apoyar el desarrollo de aplicaciones colaborativas de educación virtual de uso libre. Las investigaciones futuras estarán dirigidas a implementar componentes CSCL específicas en la imparticion virtual de las asignaturas de la Escuela Universitaria de Ingeniería Técnica de la Universidad de Oviedo (EUTIO), como apoyo a los métodos presenciales.

Referencias

[1] Labra, J., Morales J., Fernández A., Sagastegui H.: *A Generic e-Learning Multiparadigm Programming Language System: IDEFIX Project*. SIGCSE'03, USA (2003)

[2] Pérez, R., López, A.: *Aulanet, una Experiencia de Aula Virtual*. Spain (2000)

[3] Ellis, C., Gibbs, S., Rein, G.: *Groupware some issues y experiences*, Comm. of the ACM, Vol. 34 No. 1 (1991) 38-58.

[4] Conklin, J.: *Capturing Organizational Memory. Readings in Groupware and Computer-Supported Cooperative Work*. Morgan Kaufmann Publishers, CA (1993) 561-565

[5] Guerrero, L. Fuller, D.: *CLASS: A Computer Platform for the Develpment of Education´s Collaborative Applications*. Proceedings of CRIWG´97, 3rd International Workshop on Groupware, Spaín (1997) 1-3.

[6] Gokhale, A.: *Collaborative Learning Enhances Critical Thinking*. Journal of Techonology Education, Vol. 7, Nº 1, Fall 1995, University Libraries. Virginia (1996)

[7] Aczel, J.: *The Evaluation of a Computer Program for Learning Logic: The Role of Students' Formal Reasoning Strategies in Visualising Proofs*. CALRG Technical report (2000) 192

[8] The Jape Visualisation Project http://iet.open.ac.uk/pp/j.c.aczel/Jape/index.html

[9] LEGO Mindstorms http://mindstorms.lego.com/eng/default.asp

[10] TopClass e-Learning Suite™ http://www.wbtsystems.com/products

[11] IBM/Lotus Software http://www.lotus.com/home.nsf/tabs/learnspace

[12] Lin, W.: CSCL Theories. Texas University. USA (1996) http://www.edb.utexas.edu/csclstudent/Dhsiao/theories.html

[13]Computer-Supported Intentional Learning Environments www.ed.gov/pubs/EdReformStudies/EdTech/csile.html