VIII Jornadas de Enseñanza Universitaria de la Informática

Cáceres, del 10 al 12 de julio de 2002

VIII Jornadas de Enseñanza Universitaria de la Informática

JENUI 2002

Departamento de Informática
Universidad de Extremadura

Cáceres, del 10 al 12 de julio de 2002

ACTAS DE LAS VIII JORNADAS DE ENSEÑANZA UNIVERSITARIA DE LA INFORMÁTICA (JENUI 2002)

ORGANIZADAS POR:

Departamento de Informática
Universidad de Extremadura

ENTIDADES COLABORADORAS:

Asociación de Enseñantes Universitarios de Informática (AENUI)
Asociación de Ingenieros Informáticos de Extremadura (AIIEx)
Ayuntamiento de Cáceres
Diputación de Cáceres. Institución Cultural EL BROCENSE
Escuela Politécnica de Cáceres (Universidad de Extremadura)
Junta de Extremadura
La Caixa
Ministerio de Ciencia y Tecnología

Primera edición: julio de 2002

COMITÉ DE PROGRAMA

PRESIDENTA

Rosalía Ros Peña (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 EXTREMADURA

Pilar Bachiller Burgos
Pablo Carmona del Barco
Andrés Caro Lindo
Antonio Castillo Martínez
Pedro J. Clemente Martín
Mª . Luisa Durán Martín-M.
Pablo García Rodríguez
Alfonso Gazo Cervero
Alberto Gómez Mancha
Julia González Rodríguez
José Luis González Sánchez
José Luis Herrero Agustín
Adolfo Lozano Tello
Mercedes Macías García
Juan Carlos Manzano Pérez
Joe Miró Julià (U. Illes Balears)
Juan Manuel Murillo Rodríguez
Amparo Navasa Martínez
Mª . Antonia Nieto Santisteban
Miguel Ángel Pérez Toledano
Antonio Polo Márquez
Juan Carlos Preciado Rodríguez
Félix Rodríguez Rodríguez
Miryam Salas Sánchez
Marisol Sánchez Alonso
Fernando Sánchez Figueroa

COLABORADORES EN EL PROCESO DE REVISIÓN

José J. Aguilera (UJAEN)
Pedro J. Álvarez (UNIZAR)
Fernando Álvarez (UNIOVI)
Joan Aranda (UPC)
Sandra Baldassarri (UNIZAR)
José R. Balsas (UJAEN)
Pedro Blesa (UPV)
José M. Burgos (UPM)
Mateo Camps (UPCO)
Juan C. Cano (UPV)
Patricia Compañ (UA)
Regino Criado (URJC)
Antonio J. de Vicente (UAH)
Adelaida Delgado (UIB)
Juan J. Escribano (UEM)
José Fortes (ULPGC)
Isabel Gallego (UPC)
Javier Galve (UPM)
José A. Gámez (UCLM)
Mª . José García (UEM)
Mª . José Gil (DEUSTO)
Domingo Giménez (UM)
Estrella Gómez (UEM)
Mª . Engracia Gómez (UPV)
Antoni Grau (UPC)
Ángel Grediaga (UA)
Carlos Gregorio (UCM)
Juan A. Guerrero (UCLM)
Inés Jacob (DEUSTO)
Mª . Carmen Juan (UPV)
Pedro J. Lara (UEM)
Lenin G. Lemus (UPV)
Fernando Llopis (UA)
Antonio Maña (UMA)
Esperanza Marcos (URJC)
Antonio Martí (UPV)
Juan C. Martínez (UPV)
Manuel Mejías (US)
Joe Miró (UIB)
Andrés Molina (UJAEN)
Antonio Moreno (URV)
Susana Muñoz (UPM)
Raúl Murciano (UEM)
Iñaki I. Ochoa (UNIZAR)
Santiago Ortego (EUPMT)
Ángel F. Perles (UPV)
José Poveda (UV)
Alberto Prieto (UGR)
Jorge Ramió (UPM)
Isabel Ramos (US)
Miguel Rebollo (UPV)
Jairo Rocha (UIB)
Francisco Ruiz (UCLM)
Jesús Salido (UCLM)
Fermín Sánchez (UPC)
Lourdes Tajes (UNIOVI)
Maite Urretavizcaya (EHU)
Alberto Verdejo (UCM)
Ferran Virgós (UPC)
F. Javier Zarazaga (UNIZAR)

PRÓLOGO

El presente volumen contiene los trabajos presentados en las VIII Jornadas de Enseñanza Universitaria de la Informática (JENUI 2002), celebradas en Cáceres los días 10, 11 y 12 de julio de 2002.

Esta octava edición ha afianzado la importancia de las Jornadas, a las que se han presentado más de 140 colaboraciones. Este aumento de ponencias enviadas, además de suponer un trabajo enorme para el cuerpo de revisores, ha contribuido al incremento de la calidad global y supone una garantía de continuidad de estas Jornadas.

Finalmente se aceptaron 80 trabajos (lo que determina un 57 % de índice de aceptación), después de una ardua labor de selección del Comité de Programa, dado el número y calidad de las contribuciones enviadas desde 35 universidades españolas. Aunque no se han recibido trabajos de todas las universidades, sí debe decirse que hay contribuciones de muchas comunidades autónomas, acercándonos al objetivo de llegar a ser el foro de la enseñanza de la Informática universitaria en España.

Estas Jornadas no hubieran sido posibles sin el trabajo voluntario y desinteresado de muchas personas. Quisiéramos agradecer a todos los miembros del Comité de Programa su asesoramiento en muchas decisiones sobre la organización de las Jornadas. También queremos agradecer su ayuda a las entidades colaboradoras, cuyo patrocinio ha resultado fundamental para alcanzar nuestro objetivo.

Finalmente, debemos expresar nuestra gratitud a todos los autores que enviaron sus artículos y a los participantes en estas Jornadas, que son los que garantizan el éxito de JENUI.

Cáceres, julio de 2002

El Comité Organizador de JENUI2002

CONTENIDOS

Ponencias

Demostraciones

ÍNDICE DE AUTORES

Ponencias

Tema estratégico

Formación a distancia y entornos virtuales

Necesidades específicas para la docencia de programación en un entorno virtual

Ma. Jesús Marco Galindo
Josep Prieto Blázquez

Estudios de Informática y Multimedia
Universitat Oberta de Catalunya
e-mail: {mmarcog,jprieto}@uoc.edu

Resumen

Es comúnmente aceptado el hecho de que la enseñanza de programación debe ir siempre acompañada de la realización de actividades prácticas por parte de los estudiantes con la finalidad de consolidar su aprendizaje en esta materia. La realización por parte del estudiante de prácticas de programación a través de un entorno no-presencial de enseñanza no es tarea fácil.

En el presente artículo presentamos la experiencia de diseño y desarrollo de un curso inicial de programación en un entorno virtual, curso impartido desde hace cuatro años en la asignatura de *Fundamentos de programación I (FPI)* en las Ingenierías Técnicas de Informática (especialidades Gestión y Sistemas) de la *Universitat Oberta de Catalunya (UOC*).

A partir de esta experiencia, analizamos diferentes herramientas que consideramos indispensables para impartir docencia virtual en programación. Herramientas que han permitido mejorar el proceso de realización de prácticas virtuales de programación y que han contribuido por tanto a la mejora de la satisfacción y del rendimiento efectivo de los estudiantes. Concretamente analizamos la experiencia del uso en esta asignatura de:

- Los *laboratorios virtuales* de programación
- Herramientas para la *corrección automática* de programas y para la *simulación* de algoritmos.

1. Introducción

El aprendizaje de las asignaturas de programación en las Ingenierías Informáticas se basa fundamentalmente en la realización por parte de los estudiantes de múltiples ejercicios prácticos de programación de dificultad progresiva a través de los cuales el estudiante adquiere y consolida sus conocimientos de programación en diferentes lenguajes.

Concretamente en la asignatura de *Fundamentos de Programación I* para conseguir los objetivos propuestos durante el semestre, se combina el estudio de la teoría algorítmica – diseño de algoritmos – con la práctica continua de programación en C – codificación de programas -.

La pauta recomendada de estudio de esta asignatura consiste en el estudio de cinco módulos didácticos según un calendario propuesto inicialmente y guiado por las recomendaciones del profesor. Este estudio se combina con la realización continua de ejercicios consistentes en el diseño del algoritmo correspondiente y en su posterior codificación en el lenguaje de programación C.

La asimilación correcta de los conceptos depende crucialmente de la realización continua de los ejercicios propuestos.

Hasta aquí nada difiere del planteamiento clásicamente adoptado por la mayoría de asignaturas de introducción a la programación. No obstante, el hecho de que la docencia de esta asignatura se lleve a cabo en un entorno no presencial condiciona sustancialmente la manera en que se desarrolla el curso y es cuando aparece

la necesidad de incorporar herramientas específicas que permitan y faciliten el desarrollo virtual de este planteamiento eminentemente práctico.

semestre de las Ingenierías Técnicas en Informática. Actualmente tiene 1300 estudiantes repartidos en 15 aulas diferentes, cada una coordinada por un profesor. El objetivo básico de la asignatura es el aprendizaje de los

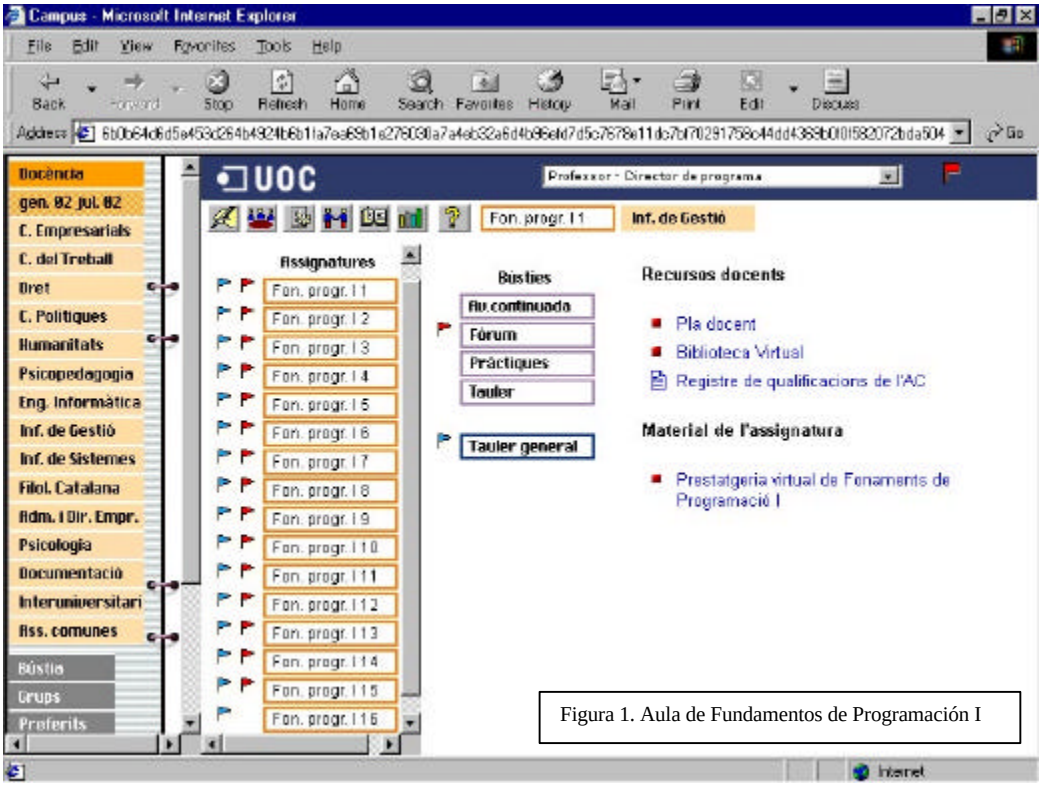


Figura 1. Aula de Fundamentos de Programación I

2. Entorno de la asignatura

La UOC fue creada en 1995 para facilitar el aprendizaje a distancia de forma virtual dentro de la educación universitaria. El uso de la tecnología, permite romper la barrera espacio-temporal y ofrecer un modelo de formación en Internet a través de un campus virtual, gracias al cual, el estudiante puede acceder al aprendizaje desde cualquier lugar y en cualquier momento. El estudiante pasa pues, a ser el centro de un proceso formativo personalizado, asistido por un equipo docente y por unos recursos didácticos y servicios de apoyo específicos[1].

La asignatura de Fundamentos de Programación I es una asignatura troncal de primer

conocimientos fundamentales de programación estructurada, a través de la algorítmica y de la realización de prácticas en lenguaje C.

El aula de teoría, que es dónde se desarrolla la asignatura sigue la estructura habitual de un aula dentro del *campus virtual* de la UOC, [2] que tal y como refleja la figura 1 consta básicamente de:

- Elementos de *comunicación*: tablón del profesor y foro.
- Elementos de *acceso a la información* asociada: plan docente, material complementario, biblioteca virtual, registro de calificaciones, …

La especial importancia que adquiere en esta asignatura la realización de ejercicios prácticos

comporta que los recursos habituales en una aula virtual resulten insuficientes.

Si además de esto, consideramos que el perfil de la mayoría de los estudiantes es peculiar – el estudiante medio tiene alrededor de 30 años trabaja y tiene hijos [3] -, la necesidad de incorporar nuevos recursos didácticos que faciliten la realización de las prácticas y por tanto, permitan sacar el máximo partido del tiempo de estudio del que dispone el estudiante, se hace del todo evidente.

corrección automática y de simulación de algoritmos que comentamos en los apartados siguientes.

2.1. El laboratorio virtual

Con el objetivo de dar soporte a la realización de ejercicios prácticos de programación aparece el *laboratorio de prácticas*. Cada estudiante, pues, está asignado además de a una aula específica de la asignatura, a un laboratorio virtual.

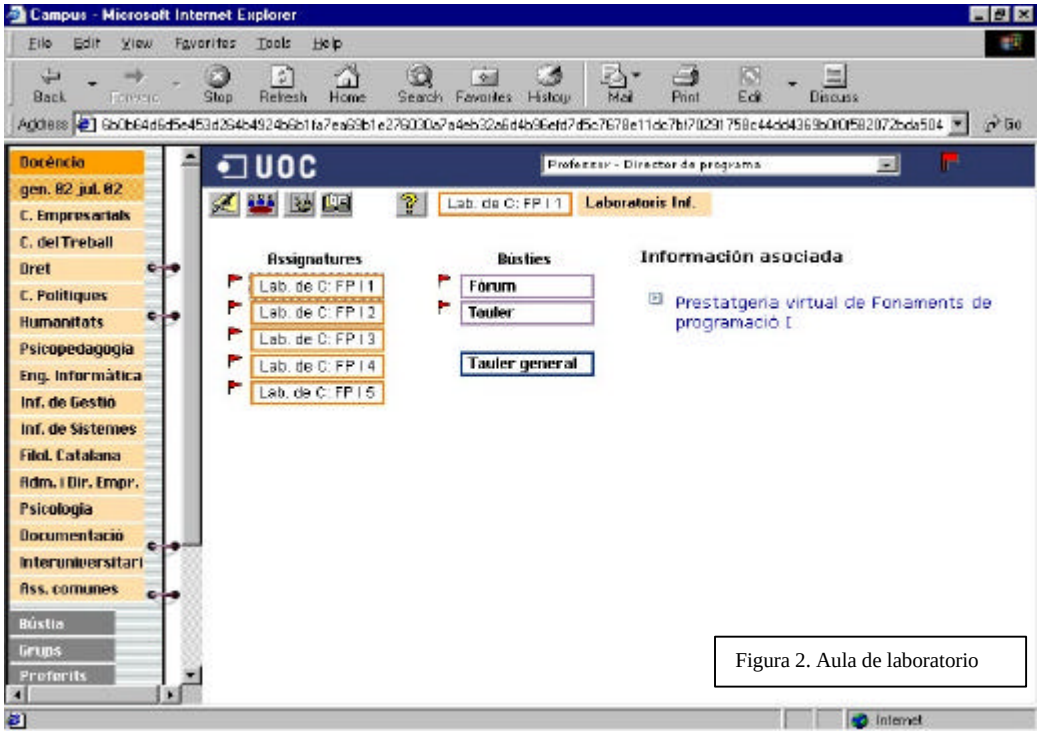


Figura 2. Aula de laboratorio

Por tanto, los tres elementos clave que determinan la necesidad del uso de nuevos recursos y herramientas son:

- El perfil específico de estudiante.
- La importancia de la realización de prácticas.
- El entorno virtual sobre el que se desarrolla la docencia.

Para dar respuesta a esta necesidad, surgen los espacios específicos para la realización de prácticas virtuales y las herramientas de

El espacio de laboratorio tiene la misma estructura y recursos que una aula virtual - tablero del profesor, foro de discusión y acceso a materiales del laboratorio específicos - como muestra la figura 2. Este espacio está coordinado por un profesor de laboratorio específico quien asume la tarea de dar soporte y guiar a los estudiantes en la puesta en práctica de los ejercicios de programación que van realizando a lo largo del curso.

De este modo, se separa la parte teórica de la asignatura - aprendizaje de la algorítmica - que se desarrolla en el aula habitual de teoría - de la parte práctica - realización de programas sencillos en lenguaje C- que se desarrolla de manera paralela en el aula de laboratorio asociada a la asignatura.

Esta separación, como veremos, presenta varias ventajas tanto docentes como de gestión.

Por un lado, desde el punto de vista puramente académico, refuerza conceptualmente la importancia de la algorítmica en el aprendizaje de la programación ya que ayuda al estudiante a diferenciar claramente entre diseño y codificación. Esto último le permite comprender que saber programar no es sinónimo de conocer un lenguaje de programación concreto, sino más bien de dominar las herramientas y estructuras algorítmicas, en definitiva, que para aprender programación es indispensable aprender a diseñar algoritmos que resuelvan problemas concretos. La posterior traducción del algoritmo diseñado a un lenguaje de programación para obtener el correspondiente programa ejecutable, es básicamente un simple paso de traducción del lenguaje algorítmico al lenguaje concreto de programación - en este caso C-.

Por otro lado, también facilita al estudiante una orientación y apoyo continuo en la instalación del software -editor y compilador - y en los procesos de codificación, compilación, depuración y pruebas de los programas que va implementando durante el semestre y que le permiten poner en práctica los conocimientos prácticos que va aprendiendo.

El uso del espacio de laboratorio virtual fomenta además la comunicación y colaboración entre los estudiantes. Éstos pueden comentar, sugerir y preguntar todo lo que necesiten relacionado con la realización de las prácticas con el resto de estudiantes de su aula. Además el espacio permite el uso de herramientas de trabajo cooperativo que posibilitan la realización de prácticas en grupo.

Y finalmente, la existencia de estos dos espacios virtuales diferenciados -aula y laboratorio- facilita también la gestión de la docencia dado que permite disponer de dos perfiles docentes distintos: un profesor responsable exclusivamente de la docencia de la algorítmica y otro responsable de la programación en el lenguaje C, profesores por tanto con objetivos docentes distintos y especializados cada uno en su ámbito determinado aunque, evidentemente, fuertemente coordinados entre sí.

La aparición de los laboratorios virtuales fue muy bien acogida tanto por profesores como por estudiantes, razón por la cual se ha extendido su uso a otras asignaturas donde también es necesario realizar prácticas de programación. Algunos ejemplos son las asignaturas de sistemas operativos que cuentan con un laboratorio de Linux, las de redes de ordenadores que cuentan con un Laboratorio de C avanzado o las de bases de datos y fundamentos de programación orientada a objetos que cuentan con laboratorio de Java. En total, este semestre hay activos 12 laboratorios virtuales que dan soporte a unos 2000 estudiantes de 22 diferentes asignaturas de las Ingenierías Informáticas.

Una vez presentado el funcionamiento de estos laboratorios, es importante resaltar que los profesores de laboratorio además de encargarse del soporte a la formación en programación en C son los encargados de la evaluación de los ejercicios de programación presentados por los estudiantes. Aunque esta función supone un volumen de trabajo muy importante para el profesor debido al número de estudiantes que tiene a su cargo (alrededor de 250 en cada laboratorio), es sumamente importante que el profesor muestre a los estudiantes los resultados de sus ejercicios con la mayor brevedad posible. De este modo no se interrumpe su proceso de aprendizaje continuo.

Para poder realizar este proceso de manera más eficiente, se incorpora al laboratorio de programación una herramienta de corrección automática de programas, herramienta que se ha convertido de uso indispensable en las prácticas de programación y analizamos en el próximo apartado.

2.2. El corrector automático de programas

Un parte del tiempo que se destina a la corrección de los ejercicios de programación consiste en la comprobación del correcto funcionamiento de estos programas, para lo cual hay que compilar y ejecutar cada uno de ellos sobre un determinado conjunto de juegos de pruebas. Es por tanto un proceso muy repetitivo y monótono que requiere

mucho tiempo de dedicación por parte del profesor.

Considerando además, el gran número de estudiantes que realizan ejercicios prácticos de programación de evaluación continua a través del laboratorio virtual, se hace evidente la necesidad de realizar esta tarea de corrección de manera más eficiente.

Es por ello que necesitamos herramientas para que este proceso de comprobación del correcto funcionamiento de los programas pueda ser automatizado. De manera, el profesor recibe directamente el resultado de esta comprobación sobre el ejercicio que ha presentado cada alumno, y así puede centrarse en aspectos que requieran un tratamiento más individualizado y que realmente aporten un valor añadido a la corrección del ejercicio como por ejemplo la corrección y evaluación de la calidad del diseño del programa.

En el mercado existen diversas herramientas, algunas de ellas creadas por universidades con necesidades parecidas. Durante cuatro semestres se ha estado utilizando en la asignatura de FPI el *Ceilidh*, desarrollado por el departamento de Computer Science de la Univesidad de Nottingham (UK). Se tuvo que adaptar el sistema Ceilidh al campus virtual de la UOC ya que estaba pensado inicialmente para funcionar sobre plataformas UNIX y conexión telnet al servidor. Se adaptó pues el sistema de tal manera que los estudiantes enviaban sus códigos fuente mediante una adjunción en un mensaje a través del sistema de mensajería del campus virtual y automáticamente recibía otro mensaje con la calificación de su ejercicio.

Inicialmente, el sistema se puso a disposición de 719 estudiantes de la asignatura que tuvieron a su disposición diversos ejercicios de programación en lenguaje Pascal para practicar.

En semestres posteriores, se ha ampliado su uso a otras asignaturas que realizan prácticas en lenguajes de programación distintos como Java y C hasta llegar a ser utilizado por más de 2000 estudiantes por semestre, alcanzando puntas de 300 correcciones diarias entre las 15 asignaturas implicadas.

A finales del curso 2000-2001, se decidió prescindir de esta herramienta y empezar el desarrollo de una herramienta de corrección automática propia. Esta decisión fue debida principalmente a dificultades en el mantenimiento y la implementación de modificaciones y al bajo rendimiento, que hacían insostenible su utilización a largo plazo.

Se inició entonces el proyecto SICAP - Sistema Interactivo de Corrección Automática de Programas - [4] cuyo objetivo principal es diseñar e implementar un sistema informático que permita la corrección automática de los ejercicios de programación.

En un sentido amplio, por corrección automática de programas se entiende desde la comprobación de su correcta ejecución ante un conjunto de pruebas predeterminado, hasta la validación de la complejidad, tipografía y estructura del código fuente, así como también la detección de posibles copias entre las soluciones aportadas por los diferentes estudiantes [5].

Por ello el sistema, además de automatizar la funcionalidad básica de corrección de programas, pretende también incorporar mecanismos de inteligencia artificial. De esta manera se posibilitaría la personalización de la respuesta facilitada al estudiante de tal modo que, en base al resultado obtenido en cada ejercicio, se le puedan hacer recomendaciones de estudio concretas y particulares. Por otro lado, ayudaría al profesor a detectar posibles errores o ambigüedades en los enunciados de los ejercicios planteados o en los juegos de prueba utilizados para su corrección, así como constatar posibles problemas en la metodología planteada para el estudio de contenidos concretos.

Para conseguir estos dos últimos objetivos será necesario que el sistema cumpla dos requerimientos básicos:

- Permitir la simulación de algoritmos que ayuden a la comprensión de los mecanismos de diseño y de ejecución de los ejercicios de programación realizados como comentaremos en el próximo apartado.
- Incluir el uso de herramientas estadísticas, de monitorización y de minería de datos que permitan al profesor explotar convenientemente la información y obtener el conocimiento relevante en base a los resultados obtenidos por sus estudiantes.

El sistema redundará por tanto, no únicamente en un ahorro de tiempo para el profesor sino que ofrecerá además al estudiante un seguimiento más personalizado, y por tanto de más calidad, de la evolución de su aprendizaje. El estudiante por otro

lado, podrá interactuar ininterrumpidamente con el sistema y con ello aumentará su autonomía de estudio y le permitirá seguir un ritmo individual y flexible de aprendizaje.

El resultado del proyecto permitirá disponer de un corrector automático diseñado específicamente para las necesidades actuales y futuras de las asignaturas que requieran la realización de ejercicios de programación, planteado desde el principio como un sistema multiplataforma, estable y adaptable.

estandarización a grupo del QTI (Question & Test Interoperability Specification) del IMS Global Learning Consortium [6] y SCORM (Sharable Content Object Reference Model) [7].

2.3. Simulador de algoritmos

Como hemos visto, para facilitar el aprendizaje de los conceptos más abstractos de la algorítmica, es importante disponer de herramientas de simulación que ayuden al estudiante a comprender por ejemplo las estructuras básicas –secuencial, condicional e iterativa -. Con este propósito se inició paralelamente al proyecto SICAP de corrección automática, el proyecto eADA – sistema electrónico para el aprendizaje de algorítmica -.

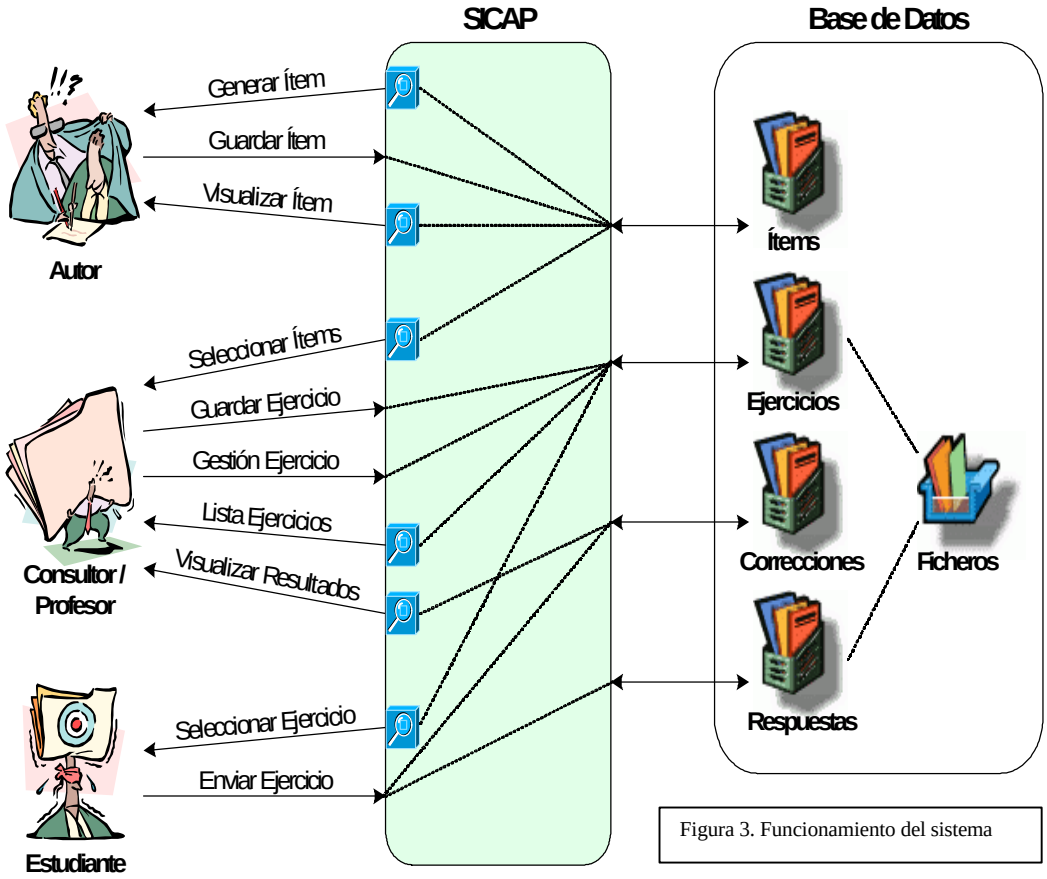


Figura 3. Funcionamiento del sistema

La figura 3 muestra el funcionamiento básico del sistema.

Un objetivo importante del proyecto es también la difusión y la relación con el resto de universidades y centros de formación para unificar criterios en los sistemas de corrección automática de programas, y realizar una propuesta de

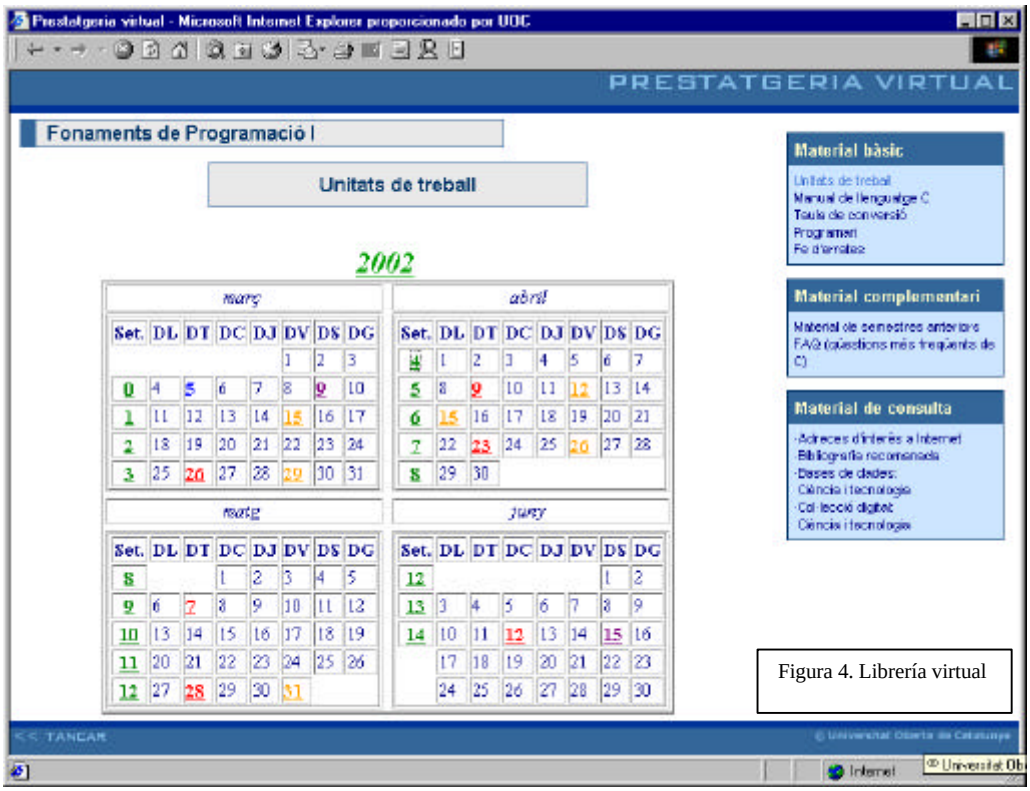


Figura 4. Librería virtual

El objetivo de este proyecto es proporcionar al estudiante una herramienta que posibilite el seguimiento del desarrollo que necesita un algoritmo desde su concepción hasta que está listo para ser traducido a un lenguaje de programación. Debe ayudarle a entender los principales conceptos y estructuras algorítmicas así como a desarrollar habilidades de diseño de algoritmos.

Para ello, debe dar respuesta a las siguientes funcionalidades:

- Actuar como material de síntesis y de glosario de los conceptos más importantes de algorítmica.
- Desempeñar el papel de pizarra virtual que permita mostrar el proceso a seguir para diseñar algoritmos.
- Actuar como depurador de algoritmos.
- Permitir la interacción del estudiante para que éste pueda experimentar cómo cambios en el diseño afectan al funcionamiento del programa.
- Actuar como herramienta de aprendizaje de disponibilidad continua.
- Incorporar el uso de recomendadores que personalicen el aprendizaje y ayuden al estudiante a resolver dificultades concretas.

La herramienta ha de permitir la adaptación continua de los contenidos a las necesidades específicas de cada semestre y por lo tanto modificar y ampliar los contenidos de manera cómoda y ágil para los profesores.

Actualmente, para dar respuesta a esta necesidad, se utiliza la biblioteca virtual de FPI – figura 4 - mientras en paralelo se emprende el desarrollo de la primera fase del proyecto eADA. Esta primera fase consiste en la implementación de un prototipo con funcionalidades básicas de simulación de algoritmos y de interacción con el estudiante.

Fases posteriores del proyecto contemplarán la creación de herramientas de autor para agilizar la gestión de los contenidos con los que interactúa el sistema. La evolución del prototipo base deberá incorporar el uso de agentes inteligentes que permitan una tutorización personalizada y adaptada a las necesidades concretas de cada estudiante.

Conclusiones

La docencia virtual de programación a lo largo de todos estos semestres ha puesto de manifiesto las dificultades más importantes con las que se encuentra el estudiante durante su aprendizaje de este tipo de contenidos en un entorno virtual.

Estas dificultades pueden sintetizarse en dos:

- La dificultad en la compresión de contenidos abstractos que son de difícil transmisión de manera escrita.
- La dificultad para la realización de prácticas no presenciales.

Afortunadamente, un entorno virtual no únicamente agudiza estas dificultades sino que a la vez posibilita, como hemos visto, el uso de sistemas que permiten paliarlas en gran medida:

- Herramientas dinámicas y visuales de simulación.
- Laboratorios virtuales de programación y herramientas de corrección automática de programas.

Estamos pues aprovechando la potencialidad que nos ofrecen las tecnologías de la información i comunicación para convertir, lo que se podría considerar como una desventaja del entorno virtual de aprendizaje, en una ventaja pues un entorno virtual permite incorporar más fácilmente este tipo de herramientas automáticas y de simulación que un entorno presencial.

De todos modos, habrá que evaluar qué beneficios reales aportan al estudiante el uso de las herramientas que actualmente están en desarrollo, para poder comprobar en qué medida éstas permiten paliar las dificultades de su aprendizaje no presencial. Un análisis de este tipo nos permitiría además, descubrir hacia dónde debemos enfocar el estudio de futuros sistemas para el soporte de la enseñanza virtual.

Referencias

[1] J. Ma. Duart, A. Sangrà. *Aprenendizaje y virtualidad*.UOC. Barcelona, 1999.

[2] Espasa Anna, Marco Ma. Jesús. *Estudi comparatiu de dues assignatures en un entorn virtual d'aprenentatge*. Comunicación del TIEC. Barcelona, 2002.

[3] http://www.uoc.edu/

[4] Joseph Prieto Blázquez. *Proyecto de Evaluación Automática SICAP*. UOC, 2002.

[5] J. Sohonen. *Model for a semi-Automatic Assesment Tool in a Web-Based Lewarning Environment.* International Conference on Computers Education, ICCE 2001.

[6] http://imsprojct.org/question/index.html

[7] http://www.adlmet.org/

Plataforma de enseñanza de lenguajes de programación a través de Internet: Proyecto IDEFIX

Jose E. Labra Gayo, José M. Morales Gil
Dpto. de Informática
Universidad de Oviedo, España
e-mail: labra@lsi.uniovi.es jmmoral@correo.uniovi.es

Roberto Turrado C.
Fachhochschule of Ulm
Alemania
e-mail: rturrado@yahoo.com

Resumen

En este artículo se describe la arquitectura del proyecto IDEFIX cuyo objetivo es desarrollar una plataforma que facilite la enseñanza de la programación en diferentes lenguajes mediante la utilización de Internet. El sistema permite la realización de prácticas de laboratorio mediante la creación de un entorno dinámico de desarrollo basado en Internet. En este entorno, los estudiantes tienen acceso a través de Internet a los enunciados de los ejercicios de programación escritos en un formato XML, que facilita la presentación en sistemas heterogéneos y que permite la posterior evaluación de forma automática. El sistema facilitará la realización interactiva de los ejercicios monitorizando los resultados parciales, fomentando el desarrollo colaborativo y facilitando la automatización del proceso de evaluación.

1. Introducción.

La Universidad de Oviedo, junto con otras seis universidades españolas del denominado G7 participa en la creación de un campus virtual que permita la enseñanza compartida de asignaturas de libre elección a través de Internet. Dentro de dicho campus, el proyecto AulaNet [17], propio de la Universidad de Oviedo, ha incorporado la asignatura "*Programación lógica y funcional*" desde el curso 2001/2002. Dicha asignatura es impartida por uno de los autores de este artículo y consiste principalmente en la presentación de los paradigmas de programación lógico y funcional con sus correspondientes prácticas en los lenguajes Prolog y Haskell.

Existen en el mercado diversas herramientas que facilitan la publicación de cursos generales en Internet. En particular, en el proyecto AulaNet se utiliza el sistema WebCT como herramienta común básica de desarrollo. Sin embargo, la enseñanza de lenguajes de programación tiene ciertas particularidades que suponen un mayor reto para su implementación en la red.

El proyecto IDEFIX (*Integrated Development Environment Frameworks based on Internet and eXtensible technologies*) [12] se centra en el desarrollo de entornos integrados de desarrollo a través de Internet. El proyecto está formado por investigadores del grupo Oviedo3 de la Universidad de Oviedo y de la Fachochschule of Ulm, para la cual se está desarrollando un sistema de realización de prácticas de laboratorio a través de Internet de la asignatura de Sistemas Distribuidos. En este artículo se describe la justificación y las intenciones que persigue dicho proyecto.

2. Antecedentes.

Dentro de las principales funciones que cumplen los profesores a cargo de los cursos de lenguajes de programación, está la supervisión de las prácticas de laboratorio, en donde debe encargarse de revisar y asesorar al alumno, sobre los problemas que se le plantean al llevar a la práctica los conocimientos adquiridos en las clases teóricas.

La realización tradicional de prácticas de laboratorio tiene una serie de inconvenientes, entre los que se pueden destacar [9].

- En la mayoría de los casos, el profesor no cuenta con un tiempo suficiente para atender a la totalidad de los alumnos de manera particular y resolver cada una de sus dudas.
- En muchos centros, no se cuentan con instalaciones suficientes para satisfacer la

demanda de equipos necesarios para llevar a cabo las prácticas.
- El ejercicio de las prácticas, se efectúa tiempo después de haber obtenido el conocimiento teórico, lo que presupone un esfuerzo mayor para refrescar los conocimientos provocando el surgimiento de nuevas dudas.
- La posibilidad de detectar plagios, se torna una tarea muy complicada que reclama mucho tiempo del maestro.
- Los maestros dedican mucho tiempo a tareas rutinarias, que les impiden desarrollar actividades más creativas.

La realización de estas prácticas de laboratorio a través de Internet podría subsanar algunos de los puntos anteriores.

3. Ventajas e inconvenientes de la educación a distancia.

Mucho se habla de los beneficios que las nuevas tecnologías están aportando en todos los campos, y si se considera específicamente la educación, se podría decir que, si bien resultan innegables los grandes logros y aportaciones que en este campo se están dando, también seria justo mencionar que se han creado los siguientes mitos, que amenazan con afianzarse sin su correspondiente proceso reflexivo ni empírico [6].

- La tecnología intensifica la enseñanza y el aprendizaje
- Los ordenadores han cambiado la forma de trabajar
- Los ordenadores han cambiado la forma de enseñar
- La introducción de tecnología en el proceso de enseñanza es innovadora
- Los alumnos aprenden más fácilmente con ordenadores
- La tecnología solucionará los conceptos erróneos de los estudiantes

Lo anterior nos obliga a realizar un análisis más cuidadoso acerca de las beneficios e inconvenientes que representa el hacer uso de Internet para propósitos educativos, evitando hacerlo de forma indiscriminada y atendiendo a sus repercusiones. Varios autores [2,4] dan cita a los aspectos favorables del empleo de la red en forma adecuada, destacando los siguientes:

- Evita los desplazamientos de los usuarios.
- Flexibiliza los horarios educativos.
- Promueve la formación de hábitos.
- Individualiza el ritmo de aprendizaje.
- Diversifica los métodos y las tecnologías educativas.
- Facilita la actualización de los contenidos.
- Permite la simulación de actividades.
- Facilita el acceso y aumenta las referencias bibliográficas.
- Promueve la participación activa y el trabajo colaborativo.
- Facilita el trabajo administrativo del maestro.
- Retroalimenta al maestro y le permite reorientar el curso oportunamente.

Obviamente, las mismas fuentes citan los siguientes inconvenientes:

- Escasez de relaciones humanas.
- Resistencia al cambio de parte del profesorado.
- Necesidad de capacitar a los maestros.
- La legitimidad de los procesos de evaluación no está lo suficientemente garantizada.
- El prestigio de los títulos obtenidos por este medio, no gozan de mucho reconocimiento.
- La libertad y flexibilidad pueden resultar contraproducentes para los alumnos inmaduros.
- Incertidumbre sobre la eficacia de los resultados y sus efectos psicológicos.
- Los problemas técnicos y los costes que trae consigo el uso de Internet como plataforma de trabajo.

Lo anterior, invita a reflexionar sobre el uso indiscriminado de las nuevas tecnologías, y a proceder con cautela en su aplicación como una panacea para la resolución de todos los problemas.

4. Propósitos y características del proyecto.

Como bien sabemos, en el proceso de enseñanza de lenguajes de programación, se pretende entrenar al alumno, para que pueda desarrollar un conocimiento creativo, que le permita modelar problemas del mundo real y trasladarlos al dominio del ordenador, para obtener soluciones de

una manera más rápida y eficiente. Esto constituye sin lugar a dudas, el objetivo primordial que debe cumplir nuestro proyecto, pero también se nos presenta la oportunidad de lograr otros beneficios que no están enfrentados con el anterior, y que de alguna manera, pueden contribuir al logro de esta y de otras metas.

Por lo anterior, se ha considerado, que el proyecto debe incorporar las siguientes características como cualidades principales de su diseño.

4.1. Asegurar un modelo eficiente de enseñanza-aprendizaje.

Se pretende llevar a la práctica el esquema representado en la figura 1.

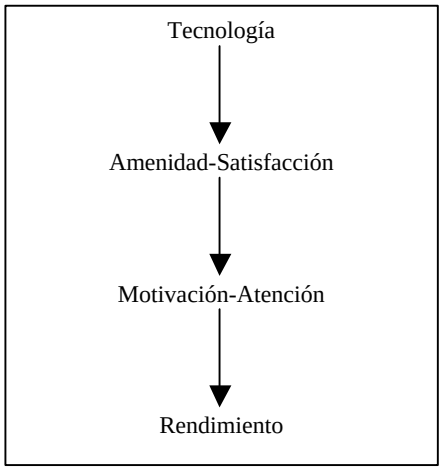


Figura 1. percepción del impacto de la tecnología en el proceso educativo.

Para lo cual se considera muy importante, basar la estrategia en un modelo que esté fundamentado en los siguientes puntos:

- Captar la atención del alumno, mediante una inducción que lo motive a aprender.
- Proponer actividades que varíen los estímulos y diversifiquen las formas de trasmisión y adquisición del conocimiento.
- Emplear una comunicación efectiva, donde se incluye la sencillez en los términos, y el uso apropiado de tecnicismos.
- El uso efectivo de apoyos visuales, con un propósito definido, y apareciendo en el momento apropiado.
- La organización lógica (que incluye los objetivos, metodología y control del tiempo), incluyendo el manejo de preguntas, tanto las enfocadas a evaluar el desempeño, así como las que deberán contestarse a los alumnos.
- Situar al alumno, haciéndole saber siempre, qué actividad está realizando y las diversas opciones con las que cuenta.

4.2. Facilitar la labor administrativa del profesor.

Existen numerosas herramientas para la administración y evaluación del trabajo del estudiante a distancia [3,11]. El sistema proporcionará al profesor diversas utilidades enfocadas a llevar a cabo de una manera sencilla y organizada, el control de los aspectos académicos y administrativos del curso.

Esto le permitirá, tanto a él como a la administración de la universidad, así como de manera restringida a los mismos alumnos, conocer información estadística diversa, sobre el desempeño académico individual o colectivo del curso, además de facilitar los procesos administrativos de inscripciones, listas, pagos etc.

4.3. Crear un herramienta flexible e independiente del lenguaje de programación.

La adaptabilidad del sistema, para servir a la enseñanza de varios lenguajes de programación sin importar su paradigma es un reto importante en una asignatura como "*Programación lógica y funcional*" en la que, actualmente, se imparten dos lenguajes de programación (Prolog y Haskell) de dos paradigmas diferentes. Esto implica un diseño flexible, que permita cambiar la configuración de las herramientas utilizadas por el sistema como, por ejemplo, los bancos de ejercicios y pruebas, el manejo de interpretes, y sus interfaces con las diversas herramientas que controlarán la trasmisión de la información a través de Internet.

4.4. Desarrollar interfaces adaptadas a las características individuales de cada alumno.

Todos los seres humanos, aunque poseemos cualidades más o menos homogéneas, contamos con ciertas cualidades o destrezas desarrolladas en mayor o menor mediada, lo cual implica que las interfaces de usuario provistas en el entorno del sistema, deben acoplarse a dichas diferencias, de modo que cualquier persona sea capaz de utilizar la herramienta sin problemas. Atendiendo a lo anterior, se debe procurar que tanto el diseño de las paginas web y el entorno iconográfico que en ellas se maneje, así como el ambiente grafico con que contará el alumno para desarrollar sus programas de práctica o de examen, deben adaptarse a sus propias habilidades o preferencias.

Ante todo, será importante ofrecer un entorno amigable y sencillo, que permita a los usuarios realizar las actividades que deseen sin contratiempos ni dificultades. El grupo Oviedo3 ha desarrollado diversas herramientas que permiten comprobar la usabilidad de sitios Web de forma remota [7]. La incorporación de dichas herramientas permitirá obtener datos sobre la interacción realizada por los usuarios y favorecerá la calidad del sistema.

4.5. Propiciar ambientes de trabajo en equipo.

Aunque la educación a distancia, se percibe como una herramienta para trabajar de manera aislada, las herramientas que nos proporciona para interactuar con otros usuarios, como el correo electrónico, *chat*, grupos de discusión, etc. ofrecen la posibilidad de realizar un sistemas de creación colaborativa de información [10].

En el caso de lenguajes de programación, las peculiaridades de los mensajes de error de cada sistema y las múltiples posibilidades que aparecen al realizar programas, hacen necesario el desarrollo de sistemas colaborativos de asistencia al programador. No es de extrañar, la proliferación de los denominados sitios Wiki dedicados a diferentes lenguajes de programación, en los que los propios usuarios pueden añadir información de forma interactiva. En el proyecto se pretende incorporar un sistema similar de asistencia a los estudiantes en el que ellos mismos puedan añadir información de ayuda.

4.6. Proporcionar ayuda en línea

La oportunidad al prestar la ayuda, se transforma en un elemento fundamental para el aprovechamiento del tiempo y la asimilación de conocimiento. Se contará con un sistema de ayuda automática, que podrá accederse para comparar los problemas que se presenten al alumno, contra los que aparecerán un banco histórico, que almacenará los problemas o preguntas más frecuentemente solicitadas, además de contar con una bibliografía de consulta, para extender los conocimientos en algunos temas en los que se desee profundizar.

También es posible contar con asesorias síncronas, mediante videoconferencia o *chat*, que pueden programarse en horarios preestablecidos y asesoría asíncrona por medio del correo electrónico.

4.7. Implementación mediante servicios web.

La implementación de una herramienta flexible, que pueda ser ejecutada desde cualquier ordenador y además transportada a diferentes plataformas, exige el uso de herramientas estándar, soportadas por los servicios que proporciona Internet. En la actualidad, el desarrollo de aplicaciones sobre Internet tiende a basarse en los denominados servicios Web, que permiten descentralizar el control de determinados componentes de las aplicaciones facilitando una mayor versatilidad.

4.8. Internacionalización de la información

Hasta hace unos años, gran cantidad de software se producía pensando en satisfacer las necesidades locales o de un mercado reducido, y por lo tanto en un solo lenguaje. En la actualidad, el uso potencial por usuarios que manejan otros idiomas, y que comparten sus intereses educativos en materias como los lenguajes de programación, representa un aspecto muy importante, que motiva a afrontar el reto de hacer los programas más útiles, confeccionándolos en la lengua y cultura de cada colectivo potencial de usuarios.

4.9. Compatibilidad con los sistemas existentes

Como se ha mencionado, el sistema desarrollado debe ceñirse a su incorporación en un sistema existente. En la actualidad, se utiliza el sistema WebCT, pero es necesario tener en cuenta la posible utilización de otros estándares de enseñanza a distancia. Existen numerosas propuestas de estandarización de material de enseñanza a distancia [1, 13, 14] que deben tenerse en cuenta en la elaboración de material docente.

4.10. Evaluación automática del trabajo del alumno.

Sin duda alguna, la evaluación representa un punto clave del proceso de enseñanza-aprendizaje; para los maestros, este proceso de calificar un trabajo o bien un examen, en donde no existe una respuesta concreta es una tarea ardua cuya complejidad aumenta en relación con el número de alumnos a los que se desea evaluar.

De forma particular, los profesores de lenguajes de programación, tienen un gran reto tratando de asignar una nota cuyo valor refleje realmente la cantidad y calidad de los conocimientos adquiridos por el alumno durante el curso.

Llevar a cabo lo anterior de manera automática, es quizás, la parte más difícil de implementar. Teniendo en cuenta que la mayoría de las pruebas que se realizarán en este curso serán del tipo de construir un programa, el dominio de las respuestas que puede considerarse correctas resulta infinito. Una posibilidad teórica sería la utilización de sistemas de interpretación abstracta [15] que analicen ciertas componentes de los programas elaborados por los estudiantes, sin necesidad de ejecutarlos. Sin embargo, este tipo de análisis requiere un estudio de la semántica dinámica de los lenguajes implicados y en la actualidad sólo podría aplicarse a subconjuntos de dichos lenguajes. Este estudio semántico ha sido realizado de una forma modular en [16] y se considera una futura línea de investigación.

De una forma más pragmática, la evaluación de ejercicios de programación de forma automática se ha realizado en [18]. En el proyecto IDEFIX, se desarrollará un sistema basado en la especificación, para cada ejercicio de programación, de un conjunto de pruebas que relacionan los datos de entrada con la respuesta a obtener. Algunas de estas pruebas serán visibles a los estudiantes, para que puedan validar ellos mismos sus programas. Otras pruebas serán ocultas, para evitar posibles trampas y permitir una evaluación automática. De la misma forma, se incorporará un sistema de chequeo automático de posibles copias o plagios.

5. Arquitectura del Sistema.

La arquitectura del sistema se presenta en la figura 2. Se mantienen los nombres en inglés de los diversos componentes para facilitar su reconocimiento en la implementación real. Los principales componentes son:

- **Problem Manager:** Forma el núcleo del sistema. Se comunica a través de Internet con los usuarios (alumnos, profesores y administradores) y gestiona la realización de los problemas de programación en los diferentes intérpretes y máquinas. Para ello, se crean diferentes procesos para la ejecución de cada problema utilizando un protocolo de comunicación con dichos procesos.
- **X Wrapper:** Encapsula la funcionalidad mínima de un intérprete genérico. Dicha funcionalidad viene dada por primitivas del tipo: *load* (carga un programa), *execute* (ejecuta un programa), *add* (añade un componente a un programa), *abort* (aborta la ejecución de un programa), etc. Se implementará un wrapper por cada uno de los lenguajes de programación que deseen incorporarse al sistema. En la actualidad se contempla la incorporación de los lenguajes *Haskell* y *Prolog*.
- **Machine (EXE):** El protocolo de comunicaciones con el *problem manager* también admite la ejecución de programas ejecutables distribuidos (componente *Machine*) que forman uno de los requisitos para impartir la asignatura de Sistemas Distribuidos.
- **Web based IDE:** Se encarga del interfaz de usuario y de la creación de un entorno de desarrollo amigable. El entorno se adaptará a las necesidades específicas de los usuarios.
- **Collaborative Programmer Assistant:** Se encarga de la creación de un sistema de

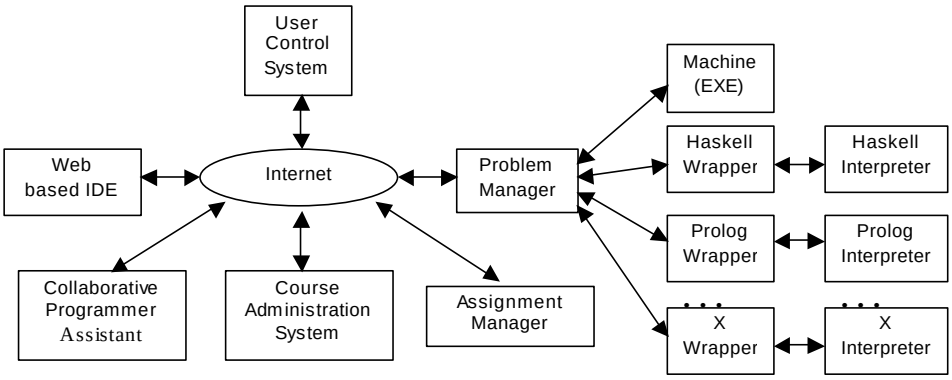


Figura 2. Arquitectura del sistema

ayuda a los programadores. Como se ha mencionado, se potenciará el desarrollo colaborativo por parte de los propios usuarios.

- **Course Administration System:** Este sistema realizará las tareas administrativas relacionadas con el curso. Estas tareas se comunican con el sistema de gestión académico general de la universidad. Por otro lado, este sistema llevará a cabo la monitorización de las diferentes peticiones realizadas por los estudiantes, facilitando la posterior evaluación. El sistema tendrá en cuenta, por ejemplo, el número de veces que un estudiante carga o ejecuta un programa, el número de programas erróneos y el tipo de errores, la participación de los estudiantes en los diversos foros, etc.
- **Assignment Manager:** Subsistema encargado de la gestión de las prácticas y problemas de programación. Aunque existen diversas herramientas para la publicación de ejercicios [8], en la actualidad se ha optado por la creación de un sistema propio basado en XML que permite la transformación automática de los enunciados a HTML y a LaTeX, permitiendo a la vez una visualización interactiva a través del Web y una impresión de calidad. El sistema admite el envío y presentación de enunciados de programación de forma temporal así como la recepción y evaluación de respuestas.
- **User Control System:** Sistema de control de usuarios. La principal novedad de este sistema es que no está empotrado en la aplicación, sino que se basa en la utilización de un servicio Web similar al servicio Passport de Microsoft. De esta forma, se facilita la compatibilidad con la posterior adopción de sistemas integrales de seguridad por parte de la Universidad.

Debido a la heterogeneidad del equipo IDEFIX en el proyecto se ha adoptado un sistema descentralizado de control de versiones basado en SourceForge y se utiliza el idioma inglés para la especificación y codificación. Por otro lado, el software y las especificaciones desarrolladas mantienen una licencia de software libre.

6. Conclusiones

En los últimos años, los sistemas de enseñanza a través de Internet han surgido de forma espectacular. En el caso de la enseñanza de lenguajes de programación, aparecen unas necesidades específicas que suponen un reto para este tipo de sistemas.

En este artículo se han justificado las principales características que debe tener un sistema de realización de prácticas de programación a través de Internet y se ha presentado la arquitectura del sistema que en estos momentos se está desarrollando en el proyecto IDEFIX.

Un objetivo principal de la arquitectura presentada es su aplicación genérica, tanto a múltiples lenguajes y paradigmas de programación, como a diferentes asignaturas. De hecho, este tipo de herramientas, también podrían ser utilizadas por empresas de desarrollo de

software, ya que facilitan la creación de sistemas colaborativos y permiten evaluar el rendimiento de los programadores.

Entre las futuras líneas de investigación se considerará: la utilización de técnicas semánticas que permitan evaluar de forma automática el trabajo desarrollado sin necesidad de ejecutar los programas (con los consiguientes riesgos de seguridad), y apoyándose en herramientas de usabilidad, se realizarán investigaciones que conduzcan a la implementación de un entorno de desarrollo amigable.

Finalmente, aunque en la actualidad el proyecto está todavía en una fase preliminar, será necesario evaluar y comparar el rendimiento académico entre estudiantes presénciales y estudiantes a distancia con el fin de establecer las pautas a seguir para mejorar el sistema desarrollado.

Referencias.

[1] Advanced Distributed Learning. *Sharable Content Object Reference Model* http://www.adlnet.org

[2] Arturo Azcorra Saloña, Carlos J. Bernardos, Óscar Gallego Gómez, e Ignacio Soto Campos, *"Informe Sobre el Estado de la Teleeducación en España"*, Departamento de Tecnologías de las Comunicaciones, Universidad Carlos III de Madrid, Enero 2001

[3] K. M. Dawson-Howe, "*Automatic Submission of Programming Assignments*", SIGCSE Bulletin 27(4) Dec. 95 pp. 51-53

[4] Centro Informático y de Comunicaciones del Edificio de Ingenierías, *"La Tecnología Educativa en el CICEI: una Perspectiva Histórica"*, septiembre de 2001. http://innova.cicei.com/historia/

[5] T. Clear, A. Haataja, J. Meyer, J. Suhonen, S. A. Varden, "*Dimensions of Distance Learning for Computer Education*", ITiCSE 2000 Working Group Reports, SIGCSE Bulletin, Vol. 33 (2). Junio 2001

[6] Hassan Fazel, Vicente Manzano, Fco. Javier Pérez, *"Variables Contextuales de las Aplicaciones Multimedia en la Universidad"*, Facultad de Sicología, Universidad de Sevilla, Revista Electrónica de Metodología Aplicada, 2000

[7] B. M. González R., Jose E. Labra G., Juan M. Cueva L. "*Web navigability testing with remote agents*". Second ICSE Workshop on Web Engineering. Lecture Notes in Computer Science, vol. 2016, pp. 311-323, Springer-Verlag, 2001.

[8] C. Gregorio Rodríguez, L. Llana Díaz, P. Palao Costanza, C. Pareja Flores, R. Martínez Unanue, J. A. Velázquez Iturbide, "*EXercita: Automatic Web Publishing of Programming Exercisesl*", 6th Annual SIGCSE/SIGCUE Conference on Innovation and Technology in Computer Science Education, SIGCSE Bulletin, Vol. 33(3), pp. 161 - 164, Sep. 2001

[9] Bruno Guardia Robles, *"Asesores Inteligentes Para Apoyar el Proceso de Enseñanza de Lenguajes de Programación",* Tesis para la Maestría en Ciencias Computacionales del I.T.E.S.M., diciembre 1997.

[10] M. Guzdial, "*Use of Collaborative Multimedia in Computer Science Classes*", 6th Annual SIGCSE/SIGCUE Conference on Innovation and Technology in Computer Science Education, SIGCSE Bulletin, Vol. 33(3), pp. 17 - 20, Sep. 2001

[11] Dorota M. Huizinga, "*Identifying topics for Instructional Improvement through On-line tracking of programming assignments*", 6th Annual SIGCSE/SIGCUE Conference on Innovation and Technology in Computer Science Education, SIGCSE Bulletin, Vol. 33(3), pp. 129 - 132, Sep. 2001

[12] Página del Proyecto IDEFIX. http://www.di.uniovi.es/aplt/idefix

[13] IEEE Learning Technology Standards Committee http://ltsc.ieee.org

[14] IMS Global Learning Consortium http://www.imsproject.org

[15] N. D. Jones, F. Nielson, "*Abstract Interpretation: a semantics-based tool for program analysis*". Handbook of Logic in Computer Science, vol. 4 Semantic Modelling. Oxford University Press, 1995.

[16] J. E. Labra, M. C. Luengo, J. M. Cueva, A. Cernuda, "*A Language Prototyping Tool based on Semantic Building Blocks*". Computer Aided Systems Theory, EUROCAST 2001, Lecture Notes in Computer Science, vol 2178, Springer-Verlag

[17] R. Pérez Suárez, A. J. López Menéndez, "*Aulanet, una Experiencia de Aula Virtual"*, Departamento de Economía Aplicada, Universidad de Oviedo, septiembre 2000.

[18] R. Saikkonen, L. Malmi, A. korhonen, "*Fully Automatic Assessment of Programming Exercises*", 6th Annual SIGCSE/SIGCUE Conference on Innovation and Technology in Computer Science Education, SIGCSE Bulletin, Vol. 33(3), pp. 133 - 136, Sep. 2001

Colecciones, correctores y generadores automáticos de ejercicios de programación

Alessandra Gallinari, Carlos A. Lázaro Carrascosa, J. Ángel Velázquez Iturbide
Escuela Superior de Ciencias Experimentales y Tecnología
Universidad Rey Juan Carlos
28933 Móstoles (Madrid)
e-mail: {a.gallinari, c.a.lazaro, a.velazquez}@escet.urjc.es.

Resumen

Este artículo pretende analizar algunos de los logros conseguidos en el ámbito de las colecciones, correctores y generadores automáticos de ejercicios. Se presta particular atención a problemas de programación, pero se consideran metodologías y temáticas comunes a otras disciplinas. Recientemente se han desarrollado múltiples herramientas pero falta una sistematización de las mismas. El objetivo del artículo es consolidar el conocimiento de estos sistemas, lo que puede ser la base para un uso más racional de ellos y para identificar mejoras.

Palabras Clave: Formación a distancia, WWW, Internet, programación, herramientas educativas, colecciones de problemas, sistemas automáticos de generación y corrección de problemas.

1. Introducción

La práctica es una parte fundamental de todo proceso educativo. Forma con la teoría un todo indisociable, ya que el preguntarse por el cómo reclama saber responder previamente el qué [31]. Además, ambas se alimentan mutuamente, como defiende McNiff cuando afirma que "la teoría y la práctica forman un ciclo, ésta pone en tela de juicio a aquélla, que revisa permanentemente sus contenidos para seguir siendo probados" [19].

Esta práctica, en el contexto de la programación, se concreta en la resolución de ejercicios, que se utilizan tanto en la etapa formativa de los alumnos como en su evaluación. A lo largo del proceso de aprendizaje, los ejercicios se suelen presentar primero en forma de ejemplos, a fin de ilustrar el contenido teórico de un tema particular o estimular al alumno a analizar y evaluar una solución propuesta a un problema dado. En una segunda fase los ejercicios se pueden proponer como problemas abiertos, de tal modo que el alumno tiene que hacer un mayor esfuerzo creativo para proponer posibles soluciones y escoger las correctas. En esta última fase el profesor puede evaluar formalmente los resultados obtenidos por parte del alumno.

La confección, clasificación, almacenamiento y corrección de estos ejercicios constituye una tarea importante en el quehacer de los profesores, y su automatización un reto para los profesionales del mundo de la informática.

Las ventajas que puede proporcionar esta automatización son conocidas:

- Acceso a grandes cantidades de información.
- Intercambio de material docente.
- Reducción de errores humanos en la corrección y mayor uniformidad en la evaluación de problemas.
- Detección de los plagios.
- Apoyo a la enseñanza individualizada de los alumnos. Una de las consecuencias de esta ventaja es la disminución de los problemas que derivan de las diferencias de nivel inicial que suelen presentar los estudiantes [9].
- Aumento de motivación de los alumnos.
- Apoyo a la autoevaluación del alumno.
- Mayor facilidad para poder trabajar en grupos.
- Posibilidad de crear para los alumnos un curriculum académico más orientado a la preparación al trabajo en empresas [11,38].
- Recopilación de estadísticas acerca los resultados obtenidos por los estudiantes, lo que favorece el proceso de evaluación de los alumnos y del sistema empleado.
- Potencial ahorro de tiempo por el profesor, especialmente en la corrección de ejercicios

Estas características son particularmente apropiadas para cursos ofrecidos enteramente a distancia. Por ejemplo, poder examinarse a distancia parece ser una importante motivación para los alumnos [28,46].

En este artículo se presentan algunas de las tendencias observadas en el ámbito de las colecciones, corrección y generación automática de ejercicios. Aunque algunos de los aspectos tratados están muy relacionados con las herramientas de visualización, no se dará una descripción detallada de éstas. Existen numerosas referencias bibliográficas dedicadas a este tema (por ej. [4,35]). Sí se tendrán especialmente en cuenta, en cambio, las herramientas de programación diseñadas para ser utilizadas en la *Web*, debido a que la utilización de las redes de ordenadores refuerza las ventajas antes mencionadas, dado el carácter de universalidad e independencia de las plataformas que éstas tienen.

La estructura del artículo es la siguiente: el segundo apartado define algunos términos importantes que familiarizarán al lector con el resto del artículo. En el tercero se presentan herramientas de apoyo a ejercicios. El cuarto apartado trata sobre la evaluación de los sistemas sometidos a este importante proceso. El quinto y último contiene unas breves conclusiones.

2. Definiciones

Las nuevas tecnologías están cambiando una de las características más propias del ser humano: la comunicación. A esto no es ajena la educación: la enseñanza, para que se convierta en aprendizaje, debe tener una fluida comunicación entre el educador y los participantes [19].

Además, estas nuevas tecnologías son necesarias para la *educación a distancia*, entendiendo como tal aquella modalidad educativa que no está regida por el espacio, constituyéndose como fundamento de su estudio una serie de materiales especialmente diseñados para guiar el autoaprendizaje. Las herramientas actualmente empleadas en la educación a distancia se pueden dividir en dos grupos: *sincrónicas* y *diacrónicas* [41]. Se diferencian en que las primeras permiten compartir material en tiempo real y requieren que los usuarios trabajen *online* simultáneamente, mientras que las segundas se pueden utilizar sin restricciones temporales. Son tecnologías complementarias, no antagonistas.

La educación a distancia no es nueva: las primeras experiencias datan de 1728, fecha en la que apareció un anuncio en la gaceta de Boston ofreciendo material de enseñanza y tutorías por correspondencia. Su uso se consolidó en el siglo XX con una mayor organización y el empleo de medios audiovisuales. No obstante, es la revolución tecnológica de los años noventa basada en la explosión de Internet la que abre nuevos cauces a este modelo educativo, reforzando todas las ventajas con las que cuenta [1,31]:

- Apertura: Internet proporciona un acceso universal a la enseñanza.
- Flexibilidad: permite al alumno escoger sus propios espacios, tiempos y ritmos de estudio.
- Economía: puede disminuir gastos asociados con desplazamientos, material bibliográfico, etc., si bien precisa de una inversión inicial y de costes de mantenimiento (ordenadores, impresoras, acceso a Internet).
- Interactividad: aunque pueda parecer sorprendente, en muchas ocasiones la relación profesor-alumno se ha visto favorecida por la inclusión de elementos tecnológicos de comunicación (correo electrónico, foros de discusión) frente a las clases puramente presenciales. Además, el acceso a Internet ofrece al alumno la posibilidad de obtener explicaciones del mismo tema con estilos y enfoques distintos.
- Seguridad: control automático sobre el posible plagio.

En cuanto a los *ejercicios*, núcleo de este artículo, el diccionario de la Real Academia de la Lengua Española los define como "*trabajo práctico que en el aprendizaje de ciertas disciplinas sirve de complemento y comprobación de la enseñanza teórica.* "

Utilizamos la palabra ejercicio para indicar una actividad didáctica que puede ser utilizada o bien en la fase de aprendizaje del alumno (para profundizar en los temas teóricos y aprender a analizar, resolver y evaluar las posibles soluciones de problemas concretos), o bien en la fase de evaluación, por parte del profesor, del conocimiento adquirido por el alumno mediante

tests, exámenes o actividades similares. Los ejercicios pueden ser ejemplos, prácticas, visualización de conceptos o algoritmos.

Entre los distintos tipos de ejercicios, los que permiten una corrección automática inmediata son los problemas de *respuesta cerrada*, donde las posibles respuestas y, entre ellas, las correctas están predeterminadas. Los problemas "tipo test" son aquellos en los que el alumno tiene que seleccionar la única respuesta correcta entre un conjunto de posibles alternativas. De tipo test son también los problemas del tipo "verdadero o falso," donde se trata de evaluar la veracidad de una proposición. Los problemas "de asociaciones" presentan dos conjuntos de objetos y la respuesta correcta es una relación que asocia de forma adecuada a cada objeto del primer conjunto un objeto del segundo. Los ejercicios "de rellenar espacios en blanco" son problemas cuyos enunciados contienen espacios en blanco y el texto completado por el alumno se compara con unas respuestas predefinidas como correctas. Una variación de este último tipo de problemas consiste en restringir a una lista predeterminada las posibles palabras (u objetos) que pueden ser insertadas en los espacios en blanco.

En el contexto de herramientas informáticas de apoyo a las prácticas podemos destacar tres tipos fundamentales:

Las *colecciones de problemas* son almacenes electrónicos de ejercicios que, como veremos, pueden tener distintos niveles de complejidad en cuanto al número de problemas disponibles y las posibilidades de clasificación y de edición de los mismos.

Las herramientas de *generación automática* de problemas utilizan varios métodos y niveles de complejidad para producir ejercicios que satisfagan requisitos sobre el tema teórico asociado, nivel de dificultad, posibilidad de incluir sugerencias o ayudas (penalizadas o no durante la fase de evaluación), método de evaluación, generación automática de una solución correcta o comentarios para el alumno.

Resulta natural, tras la generación automática de problemas, desarrollar herramientas de *corrección automática* de los mismos: la mayoría de los sistemas existentes proporcionan automáticamente una solución correcta y una evaluación del trabajo entregado electrónicamente por el alumno. Tienen también funciones de administración para el proceso de entrega de prácticas, generación de informes para alumnos y profesores, almacenamiento de notas y, finalmente, utilizan técnicas de control sobre seguridad del sistema y plagio.

3. Herramientas de apoyo a ejercicios

Las colecciones, los correctores y los generadores automáticos de ejercicios son tres tipos de herramientas que, aunque estrechamente relacionadas, mantienen unas cierta independencia, ya que existen motivaciones diferentes para construir cualquiera de los tres sistemas por separado: las colecciones representan una fuente de documentación útil por sí misma, independientemente de su grado de estructuración; los correctores automáticos liberan a los profesores de una carga tediosa, además de reducir el índice de fallos humanos y ofrecer al alumno una respuesta rápida; los generadores automáticos constituyen una ayuda eficaz al trabajo del profesor, que en muchas ocasiones debe emplear grandes cantidades de tiempo en encontrar combinaciones de variables que hagan un problema resoluble en una cantidad de tiempo razonable y predeterminada.

Por otra parte, las metodologías, estrategias, modos de implementación e incluso a veces arquitecturas suelen ser diferentes en función del sistema desarrollado.

Estos motivos, junto con la claridad de la exposición, son los que nos han llevado a abordar los tres aspectos de modo independiente.

Es preciso destacar de nuevo, en cualquier caso, que estos aspectos están muy relacionados: es deseable contar con colecciones estructuradas de ejercicios generados y corregidos automáticamente, lo que aúna todas las ventajas de los tres aspectos por separado. Además, conviene subrayar que las colecciones de ejercicios surgen de forma colateral al desarrollar sistemas de corrección de ejercicios, y de forma inherente al desarrollar sistemas de generación automática.

Éstas son las razones por las cuales es difícil hablar de una evolución histórica bien caracterizada de estas herramientas. Intentaremos delinear tendencias, experiencias pasadas y perspectivas futuras pero hay que tener en cuenta

que actualmente se utilizan herramientas de distintos modelos y niveles de complejidad.

3.1. Colecciones de ejercicios

Las colecciones de ejercicios constituyen un depósito útil de problemas adecuados para el conocimiento de una materia. Suelen contener distintos tipos de ejercicios, entre los cuales destacamos los ejercicios de *respuesta cerrada*, que en algunas ocasiones realizan una labor de reconocimiento de los conceptos, aunque en otras evalúan conocimientos avanzados (por ejemplo, posibles salidas de una función, lo que obliga a trazarla, etc.) [10,24,25,40,43]; la realización de *programas parciales o completos* [2, 3, 4, 28, 42] y los problemas *específicos de un área de conocimiento determinada*, por ejemplo problemas de grafos [7, 8].

Estos ejercicios han sido elaborados tradicionalmente de forma minuciosa y manual por los profesores de las materias científicas, con el doble objetivo de evaluar a sus alumnos y de facilitar su autoevaluación [45].

Internet proporciona, actualmente, distintos tipos de colecciones de ejercicios. En algunos casos estas colecciones proceden directamente de las elaboraciones manuales de los profesores, sin apenas utilizar los recursos que ofrece la red (hipervínculos, inserción de imágenes, etc.). Es el caso también de colecciones de problemas a propósito de concursos de programación (una lista detallada de varios concursos de programación se puede encontrar, por ejemplo, en la página: www.links2go.net/topic/Programming_Contests).

En general, estas colecciones se caracterizan por ser no estructuradas, es decir, los sistemas no ofrecen mecanismos de gestión de los ejercicios. Frente a ellas, existen numerosos sistemas que sí incluyen estos mecanismos, permitiendo las operaciones típicas de alta, baja, modificación y recuperación de los ejercicios. Además, habitualmente cuentan con exhaustivas clasificaciones de los problemas atendiendo a su dificultad, su materia u otros criterios, que permiten una fácil adaptación a las necesidades del usuario, sea alumno o profesor.

La necesidad de introducir temas de programación orientada a objetos en las asignaturas básicas de programación justifica la cada vez más frecuente presencia de ejercicios de Java y C++ en la mayoría de las colecciones disponibles. Por otra parte, el uso de la *Web* impone la utilización frecuente de Java en la implementación de las herramientas diseñadas con el fin de crear exposiciones interactivas de estos temas (ver, por ejemplo, [6,34,41]).

Pueden resultar de gran utilidad varias listas estructuradas de enlaces a material didáctico disponibles en la Red como, por ejemplo, *Computer Science Education Links* del SIGCSE (*Special Interest Group on Computer Science Education* de la *Association for Computing Machinery*, ACM) [30] y *Computer Science Education Resources* [5]. Además, *Computer Science Teaching Center* (CSTS) y *Computing Laboratory Repository* [13,23] son importantes librerías digitales aprobadas por la ACM y financiadas por la *National Science Fundation*.

Debemos destacar que algunas de las colecciones existentes se basan en el almacenamiento de los ejercicios a modo de ficheros, lo que incide en la velocidad de recuperación de los problemas [20]; otros, en cambio, utilizan sistemas gestores de bases de datos como SYBASE, dotando a los sistemas de mayor integridad y seguridad [8,10,40].

3.2. Generación automática de ejercicios

Las herramientas de generación de problemas tienen, además de su valor educativo, importantes implicaciones prácticas. Aunque no se pueda siempre afirmar que se han conseguido los objetivos prefijados, el uso de estas herramientas implica un ahorro de tiempo en el trabajo del profesor, una mayor facilidad de compartir recursos y un mayor control sobre el plagio [7,8,25]. Es destacable que, entre las ventajas que suele proporcionar un curso a distancia, el ahorro de tiempo para el profesor no se presenta siempre; existen experiencias que afirman lo contrario: un curso a distancia puede consumir más tiempo que uno presencial, debido fundamentalmente al aumento de consultas por parte de los alumnos y a las tareas administrativas que conlleva [9].

Existen diferentes enfoques a la hora de tratar el problema de la generación automática de ejercicios:

En algunos casos [17] se presenta un problema modelo para un determinado tema y es el propio alumno quien genera distintas

ejemplares de ese mismo problema introduciendo datos diferentes cada vez. De esta forma el alumno puede experimentar y hacer predicciones sobre los distintos resultados que se obtendrán [25].

Un nivel de elaboración automática de nuevos ejercicios un poco más complejo consiste en la utilización de plantillas que contienen una serie de variables que se aleatorizan, proporcionando distintos tipos de problemas [7,8,10,17,24,25,43].

Es también interesante poder utilizar sistemas que permiten elaborar exámenes o pequeñas colecciones de problemas a partir de una base de datos de ejercicios [10,36,40]. La extracción puede ser aleatoria o determinada, considerando incluso grados de dificultad para los ejercicios y, por ende, para las colecciones resultantes [27]. Estas últimas herramientas, si bien no generan ejercicios estrictamente hablando, sí se comportan como editores de los mismos, permitiendo crear problemas con un formato determinado.

Para prevenir y evitar situaciones de plagio durante los exámenes (especialmente si son exámenes a distancia) [2,7,28], algunas herramientas están diseñadas de tal forma que el alumno no puede extraer soluciones del sistema (parciales o completas), sólo puede entregar electrónicamente una versión de su trabajo y no puede recibir ninguna ayuda que no esté autorizada por el profesor.

3.3. Corrección automática de ejercicios

La corrección clara y coherente de los ejercicios es fundamental en el proceso educativo: proporciona información tanto al alumno sobre sus errores como al profesor sobre el nivel de aprendizaje de sus estudiantes. Un proceso de evaluación objetivo y uniforme contribuye a fortalecer el nivel de confianza del alumno (y del profesor) en el sistema [7,16,21,37,46].

El tipo de corrección proporcionada para cada problema depende de los objetivos de la sesión de práctica: si la práctica se está desarrollando para que el alumno vea nuevos ejemplos y aprenda nuevas técnicas, el corrector podrá proporcionar todo tipo de información, ayuda y ejercicios complementarios [ver, por ejemplo, 18], mientras que estas posibilidades se eliminarán del sistema durante sesiones de exámenes.

Desde un punto de vista técnico, la corrección de ejercicios depende drásticamente del tipo de los mismos. En el caso de ejercicios de respuesta cerrada, esta corrección es automática y casi inmediata: se trata de comparar la respuesta dada con la respuesta correcta almacenada [10, 40].

Si nos enfrentamos a problemas en los que exigimos al alumno que escriba código fuente [2, 3,22,42], intervienen más elementos: analizadores léxicos y sintácticos, comparadores de estructuras, generadores de errores, distintas métricas, etc.

Además, la corrección de ejercicios está algunas veces muy relacionada con la visualización [4 y sus referencias, 24,25,43], especialmente en problemas de grafos [7].

Un corrector automático ideal tendría que poder reproducir (y perfeccionar) el trabajo de un corrector humano. Parece ser una opinión compartida por varios de los creadores de sistemas de corrección que todavía queda mucho por hacer en este sentido. Una seria dificultad es que la mayoría de las herramientas no son capaces de distinguir lo parcialmente válido y asignar una nota positiva a un trabajo que no sea completamente correcto. Aunque la realización de un sistema con estas características representa un problema no computable, cualquier herramienta que se acerque lo más posible al modelo ideal es una importante contribución.

Algunos correctores más desarrollados [7,8, 26] intentan mirar no solamente la solución final, sino también los pasos empleados por el alumno para llegar a esa solución. Un error cometido en uno de los primeros pasos puede modificar críticamente el nivel de dificultad del problema, imposibilitando una evaluación automática equilibrada. Existen sistemas, por ejemplo [7,22], que ofrecen "pistas" para resolver el ejercicio que, en el caso de ser utilizadas, disminuyen la calificación final; otros son capaces de "actualizar" el ejercicio y su método de evaluación después de cada error cometido por el alumno, de tal forma que el error tenga un peso más adecuado.

Finalmente merece la pena destacar algunos sistemas que permiten o requieren la participación del profesor en el proceso de evaluación. Si bien no se pueden considerar correctores automáticos *strictu sensi*, facilitan la tarea de corrección, proporcionando editores *ad hoc*, diseñados para almacenar anotaciones y comentarios [7,25,29].

Entre estos hay sistemas que utilizan bases de datos para almacenar los trabajos entregados por los alumnos, las calificaciones de los alumnos y los comentarios que el profesor considere oportunos [11,15] y sofisticados asistentes para la creación de páginas *Web* de asignaturas [33], algunos de los cuales son comerciales.

3.4. Algunas herramientas integradas

Algunas herramientas integran, de algún modo, los aspectos arriba analizados. Destacamos entre ellas las siguientes:

- *Autogenerador y asistente de ejercicios interactivos vía Web* [10]: sistema que gestiona una colección de ejercicios cerrados, utilizando una base de datos centralizada. Permite la generación de nuevos ejercicios, y la recopilación de estadísticas en base a los resultados. Utiliza para su implementación CGIs y el lenguaje JavaScript.
- *PILOT* [7]: "*An Interactive Tool for Learning and Grading"*. Herramienta independiente de la plataforma, basada en la arquitectura Cliente/Servidor y en *applets* de Java, que genera ejercicios basándose en un problema 'tipo' y en ejemplos diferentes del mismo. Permite, además, la corrección parcial de los mismos y está muy vinculado a la visualización.
- *Java 'Problets'* [24,25,26]: sistemas basados en *applets* de Java. Son generadores y correctores de ejercicios basados en una plantilla y una serie de variables que, al aleatorizarlas, proporcionan gran cantidad de problemas parecidos.
- *WebToTeach* [2,3], "*Tester and Visualizer for Teaching Data Structures"* [4]: herramientas que merecen ser destacadas por su capacidad de admitir, a través del *Web*, problemas basados en que el alumno escriba código fuente.
- *Course Master* [12] (antiguamente *Ceilidh)*: creado por el *Learning Technology Research Group* en el departamento de Ciencias de la Computación en la *Nottingham University* (Reino Unido), es un sistema de evaluación automática, administración de notas, soluciones y material de asignaturas. Permite que el profesor pueda observar el trabajo de sus alumnos y generar informes sobre casos de plagio. Está basado en un sistema Cliente/ Servidor y está escrito en Java.

4. Evaluación

La evaluación es una parte fundamental del diseño de un sistema informático que tiene como misión principal recoger información sobre el mismo y a su vez optimizarlo, mediante métodos e instrumentos objetivos [19].

Cómo medir de forma apropiada una herramienta educativa es un problema todavía abierto y se han sugerido varias soluciones [23,32]. Una primera evaluación de un sistema suele estar basada en cuestionarios dirigidos a los alumnos [39, y prácticamente todas las referencias que describen algún tipo de herramientas educativas]. El CSTC ofrece la posibilidad de someter material didáctico a una evaluación que se realiza por medio de formularios y de forma electrónica. La metodología empleada en la elaboración de estos formularios es el resultado de un estudio empírico explicado en detalle en [13] y es consecuencia de la evolución histórica de los métodos de evaluación.

Para la mayoría de los sistemas analizados no existe todavía ningún tipo de evaluación, o ésta es muy informal y poco rigurosa. Hay algunos casos, en cambio, en los que la evaluación es muy tenida en cuenta. Destaca un estudio que defiende la conveniencia de realizar cursos *online* [44], porque reduce el plagio, aumenta la motivación de los alumnos y reduce el problema de contar con alumnos con un nivel previo muy diferente. Como contrapartida cabe destacar el elevado grado de responsabilidad que debe tener un alumno al estar menos controlado por parte del profesor. Estas conclusiones han sido posibles gracias a una extensa evaluación basada en completos cuestionarios dirigidos a los alumnos. Es de destacar, asimismo, la aplicación *Lecturelets* [14] que permite realizar conferencias multimedia a través de la *Web* y que recoge la evaluación a través de ficheros *log*, donde quedan registrados los pasos que da el usuario al utilizar la herramienta, información luego utilizada para la mejora del sistema. En [25,26,43] se evalúa el uso de *applets* de Java para explicar varios conceptos de programación en el lenguaje C++ en cursos de

programación del *Ramapo College* en *New Jersey* (EEUU). Los resultados son positivos en todos los casos y en algunos de ellos reflejan una mejora del 100% en la media de las notas de alumnos que utilizan estas herramientas. El *National Engineering Educational Delivery System* [32] realiza una evaluación de las herramientas educativas no comerciales para la enseñanza de la ingeniería a través de un premio anual, utilizando los siguientes criterios de diseño educativo: diseño de software y contenidos relacionados con la ingeniería.

5. Conclusiones

En este apartado de conclusiones queremos añadir a la anterior exposición de aspectos prácticos algunos aspectos educativos más generales.

Las herramientas analizadas son necesarias en la educación a distancia y presentan claras ventajas pero nos parece importante incluir algunas reflexiones críticas.

En relación al proceso fundamental de evaluación del trabajo del alumno, compartimos la opinión expuesta en [1,16,21,46] de que es imprescindible establecer criterios claros, objetivos y consistentes. Más concretamente, en el contexto de la corrección de programas, J.W. Howatt [21] identifica y valora, en orden decreciente, las siguientes características: diseño, ejecución y adherencia a las especificaciones, estilo del código, comentarios incluidos, creatividad.

P.A. de Marneffe [16] utiliza los conceptos de algoritmo correcto y algoritmo eficiente para evaluar la calidad de un programa: el alumno tiene que demostrar rigurosamente la validez de un algoritmo y predecir su complejidad antes de implementarlo, independientemente del lenguaje de programación empleado. El autor concluye que ejercicios relativos a algunos temas fundamentales en programación no son factibles de corrección y evaluación automática.

En [46] encontramos una afirmación similar de que los métodos de corrección automática no son adecuados para evaluar las habilidades analíticas de los alumnos.

Por tanto no es claro el nivel en el que los métodos de formación a distancia deben ser incluidos en el diseño curricular de los estudios de informática. En cualquier caso, y cuando sea posible, las nuevas herramientas se pueden utilizar, por ejemplo, para incluir un nivel básico de ayuda o, más en general, un curso que sirva como complemento y sea paralelo a las clases tradicionales.

Agradecimientos

A los profesores Roberto Muñoz y Cristóbal Pareja por sus sugerencias, así como al apoyo y trabajo de este último. El trabajo se ha financiado con el proyecto TIC2000-1413 de la CICYT.

Referencias

[1] Aggarwal, A. K. (ed.), *Web-Based learning and teaching technologies: opportunities and challenges.* Idea Group Publishing, 2000. Cap. IX.

[2] Arnow, D., Barshay, O., *On-line programming examinations using WebToTeach.* ITiCSE'99, 21-24.

[3] Arnow, D., Barshay, O., WebToTeach: *an interactive focused programming exercise system.* FIE'99, sesión 12a9, 39-44.

[4] Baker, R.S. et al., *Testers and visualizers for teaching data structures.* SIGCSE'99,261-265

[5] Barnett, L., Computer Science Education Resources, 2002, gum.richmond.edu/~lbarnett /Informatics /CS_Ed.html

[6] Bergin, J. et al., Resources for next generation introductory CS courses: report of the ITiCSE'99 working group on resources for the next generation CS1 course, SIGCSE Bulletin, vol. 31, nº. 4, diciembre 1999, 101-105.

[7] Bridgeman, S. et al., PILOT: an interactive tool for learning and grading. SIGCSE´00, 139-143.

[8] Bridgeman, S. et al., *SAIL: A system for generating, archiving and retrieving specialized assignments using LATEX.* SIGCSE´00, 300-303.

[9] Carrasquel, J., *Teaching CS1 on-line: the good, the bad, the ugly.* SIGCSE'99, 212-216.

[10] Coll-Madrenas, C. et al., *Autogenerador y asistente de ejercicios interactivos vía Web*. CONIED'99.

[11] Computing Curricula, Computer Science Volume, www.acm.org/sigcse/cc2001
[12] CourseMaster, 2000, www.cs.nott.ac.uk
[13] CSTS, 2002, www.cstc.org
[14] Culwin, F., *LectureLets – Web based Java enabled lectures.* ITiCSE'00, 5-8.
[15] Dawson-Howe, K., *Automatic submission and administration of programming assignments.* SIGCSE Bulletin, vol. 28, nº. 2, junio 1996, 40-42.
[16] de Marneffe, P.A., *The problem of examination questions in Algorithmics,* ITiCSE'98, 74-76.
[17] Elenbogen, B.S., Maxim, B.R., McDonald, C., *Yet, more Web exercises for learning C++.* SIGCSE´00, 290-294.
[18] *ELM-ART* ,www.psychologie.unitrier.de:8000/projects/ELM/elmart.html
[19] González Soto, A. P., Medina Rivilla, A., de la Torre, S., *Didáctica general: modelos y estrategias para la intervención social.* Ed. Universitas, 1995.
[20] Gregorio Rodríguez, C. et al., *Exercita automatic publishing of programming exercises.* ITiCSE´01, 161-164.
[21] Howatt, J.W., *On criteria for grading student programs.* SIGCSE Bulletin, vol. 26, nº.3, septiembre 1994.
[22] Jackson, D., Usher, M., *Grading student programs using ASSYST.* SIGCSE'97, 355. orion.ramapo.edu/~amruth/problets
[23] Knox, D et al., *The peer review process of teaching materials.* ITiCSE´99 Working Group Reports, vol. 31, nº. 4, 77-90.
[24] Kumar, A., *Dynamically generating problems on static scope.* ITiCSE'00, 9-12.
[25] Kumar, A.N., *Learning the interaction between pointers and scope in C++.* ITiCSE´01, 45-48.
[26] Kumar, A. N., *On changing from written to on-line tests in computer science I: an Assessment.* ITiCSE´99, 25-28.
[27] Mas, R., Lacosta, I., *Aplicaciones de Internet a la enseñanza: un sistema de autoevaluación.* JENUI, 2001.
[28] Mason, D.V., Woit, D.M., *Integrating technology into computer science examinations.* SIGCSE'98, 140-144.
[29] Mason, D.V. & Woit, D.M., *Providing mark-up and feedback to students with online marking.* SIGCSE'99, 3-6.
[30] McCauley, R., 2001, CEL: www.cs.cofs.edu/~mccauley/edlinks/
[31] Medina Rubio, R., García Aretio, L., Ruiz Corbella, M., *Teoría de la Educación. Educación Social.* UNED, 2001.
[32] NEEDS, 2001, www.needs.org
[33] Nørmark, K., *W-based tools for advance course management.* ITiCSE'00, 65-68.
[34] Northeastern Software Project www.ccs.neu.edu/teaching/index.html
[35] Pareja Flores, C., Velázquez Iturbide, J. Á., *Program Execution and Visualization on the Web*, En Web-Based Education: Learning from Experience, Anil K. Aggarwal (ed.), Idea Group Publishing, en imprenta.
[36] Polo Usaola, M., Ruiz González, F., *easyTest: una herramienta para la generación y corrección automáticas de exámenes tipo test.* JENUI, 2001.
[37] Preston, J.A., Shackelford, R., *Improving on-line assessment: an investigation of existing marking methodologies.* ITiCSE´99, 29-32.
[38] Rickman, J. et al., *Enhancing the computer networking curriculum.* ITiCSE´01,157-160.
[39] Rosbottom, John., *Computer managed, open question, open book assessment.* ITiCSE'97, 100-102.
[40] Sánchez, P.J., Martínez, L.,Muñoz, M.D. *Un sistema de generación y evaluación de exámenes basado en java.*CONIED'99, p.17.
[41] Shirmohammadi, S. et al., *Web-Based multimedia tools for sharing educational resources,* ACM J. of Educ. Resources in Computing, vol.1, no.4, Primavera 2001.
[42] Schorsch, T., *CAP: an automated self-assessment tool to check Pascal programs for syntax, logic and style errors.* SIGCSE'95, 168-172.
[43] Singhal, N., Kumar, A.N., *Facilitating problem-solving on nested selection statements using C/C++.* FIE'2000, sesión T4C-1, 2000, 1-6.
[44] Tetterton, J. C, et al., 2000, *eGrades,* www4.ncsu.edu:8030/~jctetter/Abstract.pdf
[45] Webber, A. B., *The Pascal trainer.* SIGCSE'96, 261-265.
[46] Woit, D.M., Mason D. V., *Lessons from on-line programming examinations.* ITiCSE'98, 257-259.

Resolución de ejercicios de programación en la web

Sergio Luján-Mora, Fernando Llopis
Departamento de Lenguajes y Sistemas Informáticos
Universidad de Alicante
E-03080 Alicante
e-mail: {slujan,llopis}@dlsi.ua.es

Resumen

En el presente artículo se presenta una aplicación web que permite a los alumnos resolver ejercicios planteados en la asignatura *Fundamentos de la Programación* de las titulaciones de Informática de la Universidad de Alicante. Esta aplicación se ha planteado a modo de "competición" o juego, con el fin de motivar al alumno a que realice una gran cantidad de ejercicios sobre la asignatura. Las dos principales ventajas que se logran a la hora de emplear esta aplicación son: por un lado, cada alumno puede verificar cuando desee y desde donde desee los conocimiento adquiridos sobre la asignatura y, por otro lado, los profesores obtienen información que les permite conocer el grado de aprendizaje de sus alumnos y los aspectos que mejor o peor han asimilado.

1. Motivación

En la enseñanza de programación en el primer curso de las titulaciones de Informática confluyen una serie de circunstancias que la diferencian de otras asignaturas. En primer lugar, para la mayoría de los alumnos es la primera vez que se enfrentan a una asignatura de este tipo, aunque es cierto que actualmente el conocimiento sobre informática en general, y el uso de ordenadores en particular, es mucho mayor que hace 4 o 5 años. En segundo lugar, el esfuerzo que deben realizar los alumnos para su aprendizaje debe ser mucho más deductivo que memorístico. En último lugar, también cabe citar la importancia que suele tener esta asignatura dentro de los planes de estudio a nivel de incompatibilidades o recomendaciones sobre otras asignaturas. Todo esto influye en que el alumno se sienta muchas veces "presionado" a aprobar la asignatura, lo que conduce a que el alumno intente aprobar la asignatura mediante la memorización en vez del aprendizaje correcto de los conocimientos que recibe.

La solución que recomendamos a los alumnos para que el aprendizaje de la asignatura sea lo menos costoso y a su vez lo más provechoso posible es la realización del mayor número posible de ejercicios prácticos. Con el objetivo de facilitar esta tarea a los alumnos, se han propuesto a lo largo de los seis últimos años numerosos ejercicios, que han sido recopilados junto con ejercicios de exámenes en un libro [4], donde aparecen la mayoría de los ejercicios con su correspondiente solución.

Parece claro que el disponer de ejercicios con enunciado y solución facilita el aprendizaje de la asignatura, pero en ocasiones puede producir el efecto contrario si el alumno sólo se limita a leer el enunciado y la solución, sin llegar a pasar por el proceso deductivo que teóricamente tiene que realizar para obtener la solución.

Por otro lado, la resolución por parte de los alumnos de los ejercicios propuestos en clase de teoría, formando grupos y realizando pequeñas competiciones entre ellos se ha comprobado que motiva a los alumnos y sí que logra que intenten resolver los ejercicios. No obstante, la masificación de las aulas y la cantidad de tiempo necesaria para llevar a cabo estos ejercicios, impide que se dediquen muchas clases a este método de estudio. Estos problemas no deben impedir que este tipo de clases se sigan impartiendo de dicha forma, pero sí que limita considerablemente su número.

2. Propuesta

Por todos los motivos anteriores, hemos desarrollado una aplicación web, a la que hemos denominado *Aprendizaje en la Web AutoMotivado* (AWAM) [1], que nos permite poner a

disposición de los alumnos una gran cantidad de ejercicios que pueden resolver 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). Con el fin de motivar al alumno, hemos dotado a la aplicación de una apariencia de "competición" o juego que les anime a superarse. El funcionamiento de la aplicación es sencillo: se plantea al alumno una serie de ejercicios que tienen que resolver en un tiempo determinado y el juego finaliza cuando el alumno comete un número determinado de errores o contesta todos los ejercicios. Según el número de ejercicios contestados correctamente, el alumno obtiene una puntuación, que se traduce en una posición con respecto a los resultados de sus compañeros.

Los ejercicios planteados son de tipo test, agrupados por los diferentes temas que conforman el programa de la asignatura. Estos ejercicios inciden en los errores más comunes que cometen los alumnos a la hora de programar. Cada ejercicio tiene asignada una dificultad; si el profesor lo desea, se puede emplear este valor para presentar al alumno los ejercicios de menor a mayor dificultad. Además, también existe la opción de que esta dificultad se actualice en base a las contestaciones de los alumnos (cuantas más veces sea mal contestada una pregunta, más aumentará su dificultad). Por último, también existe la posibilidad de mostrar al alumno un comentario que explique la respuesta correcta de una pregunta cuando falle.

El resto del artículo se ha dividido de la siguiente forma: en la sección 3 se comentan algunos trabajos previos relacionados con nuestra propuesta; en la sección 4 se describen las características principales de la aplicación AWAM; en la sección 5 se detalla la arquitectura de la aplicación; en la sección 6 se presentan las primeras impresiones que hemos obtenido; por último, en la sección 7 comentamos los trabajos futuros que pretendemos realizar.

3. 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. Simplemente como un ejemplo, incluimos las siguiente referencias del año 2001: [2], [3], [5], [6] y [7].

De las anteriores referencias nos interesa destacar dos por su semejanza con nuestra propuesta. Por un lado, en [2] se presenta un sistema de apoyo a la enseñanza presencial basado en Internet. El sistema 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, el sistema presenta varias carencias como son: no existe un control del tiempo que se puede emplear en contestar una pregunta, no se puede indicar la dificultad de una pregunta, todas las preguntas tienen que tener cuatro respuestas, etc.

Por otro lado, en [6] se presenta otro sistema basado en la web, que también posee ciertas similitudes con nuestra propuesta. En este sistema, el alumno puede acceder a través de Internet a listas de problemas generadas de forma aleatoria a partir de una base de datos. Sin embargo, esta propuesta carece de la interactividad e inmediatez de nuestro sistema (no dispone de un módulo para preguntas-respuestas de tipo test con corrección automática), y tampoco presenta el aspecto de juego que creemos motiva la participación del alumno.

4. Descripción de la aplicación

La aplicación AWAM se divide en dos módulos: el módulo del alumno y el módulo del profesor. Para acceder a cualquiera de los módulos, el usuario (ya sea el alumno o el profesor) tiene que disponer de un *nombre de usuario* y una *clave*. Los alumnos además poseen la posibilidad de definirse un alias que será el que utilicen en la realización de los ejercicios. De este modo, pueden realizar los ejercicios de forma anónima, sin tener que preocuparse porque el profesor vaya a tener en cuenta sus posibles fallos.

Toda la información que emplea la aplicación (usuarios, profesores, asignaturas, temas, preguntas y respuestas, etc.) reside en una base de datos. La Figura 1 representa un modelo en UML de la base de datos que emplea nuestro sistema. Cada clase de UML se convierte en una tabla en el

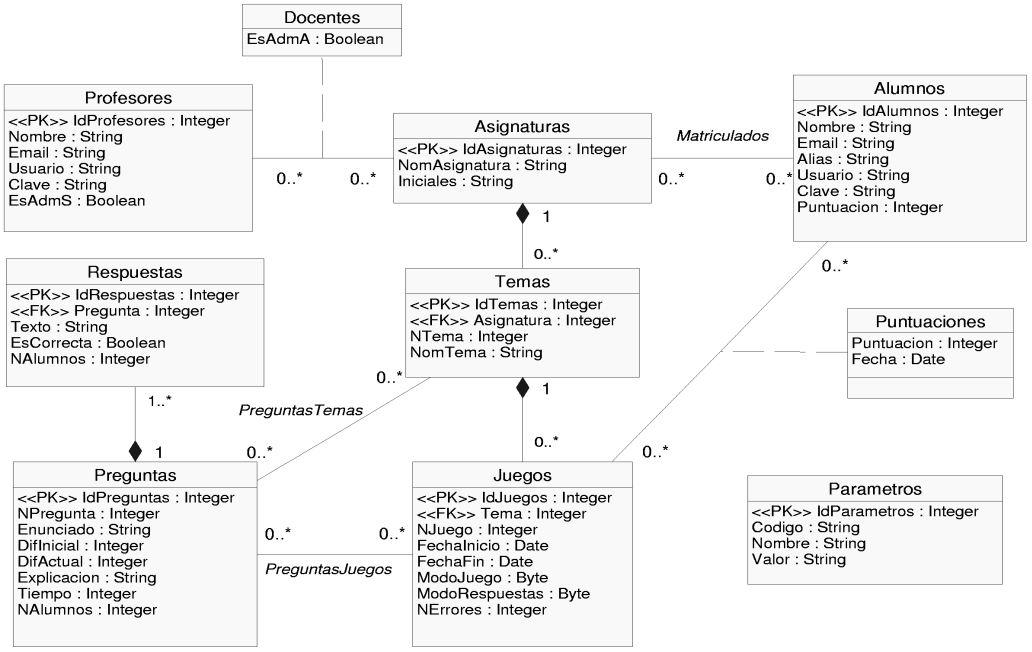


Figura 1. Base de datos del sistema modelada mediante UML

modelo relacional. Las asociaciones que poseen un nombre (PreguntasTemas, Preguntas-Juegos y Matriculados) y las clases asociación (Docentes y Puntuaciones) se convierten en tablas que permiten representar las relaciones muchos a muchos en el modelo relacional. Por último, se han marcado con <<PK>> y <<FK>> las claves primarias y las claves ajenas respectivamente.

Como se puede apreciar en la Figura 1, una asignatura se compone de muchos temas, un tema de muchos juegos, un juego de muchas preguntas, y una pregunta de muchas respuestas (todas las preguntas no tienen por que tener el mismo número de respuestas). Como una pregunta puede pertenecer a varios temas a la vez, se pueden crear temas de repaso que incluyan preguntas de otros temas.

Por último, se ha intentado que la aplicación sea lo más flexible posible. Por eso existe la tabla Parámetros que almacena distintos valores que configuran el funcionamiento de la aplicación. Estos parámetros se pueden configurar a través del menú de configuración de parámetros del sistema (MCPS) que existe en el módulo del profesor.

4.1. Módulo del alumno

A través de este módulo los alumnos pueden contestar las preguntas planteadas por el profesor. Para comenzar a "jugar", el alumno debe elegir primero la asignatura, después el tema y por último el juego en el que quiere jugar. A continuación, la aplicación muestra una a una las preguntas que contiene el juego. El orden en que se muestran las preguntas depende de cómo lo haya configurado el profesor en el momento de crear el juego (de forma aleatoria, de menor a mayor dificultad o según un orden creado por el profesor).

En la Figura 2 podemos ver la página que se muestra al alumno para que conteste una pregunta. En ella podemos observar la siguiente información:

- Enunciado de la pregunta.

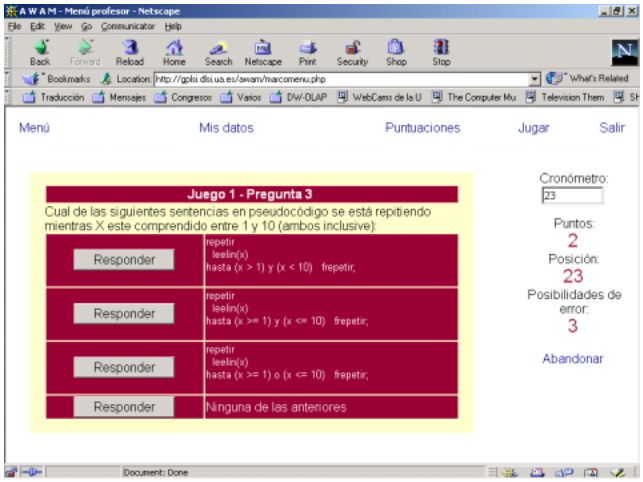


Figura 2. Página con una pregunta y cuatro respuestas

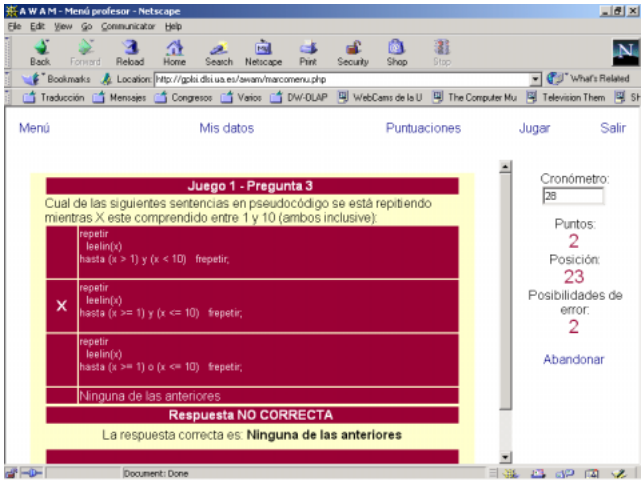


Figura 3. Página con el resultado de la contestación de un alumno

- Posibles respuestas junto con el botón que permite al alumno seleccionar cada una.
- Cronómetro: tiempo restante para contestar la pregunta. Si se cumple el tiempo y el alumno no ha contestado, se considera un fallo.
- Puntos: puntos que ha logrado el alumno hasta el momento en el juego actual.
- Posición: indica la posición del alumno con respecto a las puntuaciones del resto de sus compañeros.
- Posibilidades de error: número máximo de fallos permitidos. El profesor puede fijar este valor a través del MCPS.

Si el alumno acierta la pregunta, sigue el juego; en caso de que falle aparece un mensaje explicando la contestación correcta. Por ejemplo, en la Figura 3 vemos como el alumno ha contestado incorrectamente a la pregunta de la

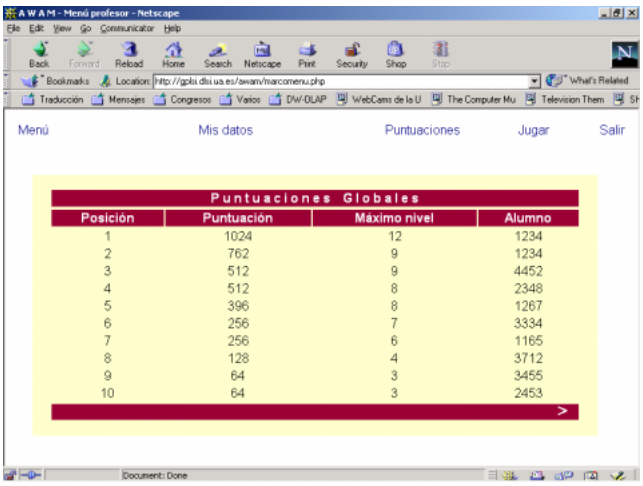


Figura 4. Tabla de puntuaciones globales

Figura 2. Si ya ha cometido todos los errores permitidos por el profesor, el juego finaliza.

Por otro lado, el alumno puede consultar las tablas de clasificación, ya sea la de puntuaciones globales (Figura 4) o la de puntuaciones por temas. Además, también puede consultar en que posición aparece dentro de cada tabla.

4.2. Módulo del profesor

El profesor o profesores, ya que el sistema contempla la existencia de distintas asignaturas, cada una de ellas con sus profesores correspondientes, dispone en su módulo de todas las opciones necesarias para mantener la base de datos del sistema. El profesor puede añadir (borrar, consultar, ...) nuevos temas, nuevos juegos, nuevas preguntas, etc. Como pueden existir distintos profesores, no todos tienen acceso a todas las opciones. Así, por ejemplo, existen profesores administradores que pueden crear asignaturas, crear otros profesores, etc.; profesores que pueden crear temas, juegos o preguntas en las asignaturas a las que pertenecen, y profesores que sólo pueden introducir preguntas en las asignaturas a las que pertenecen.

En la Figura 5 se puede ver la página principal del módulo del profesor. Se puede observar que se encuentra dividida en dos secciones (marcos): en el marco superior se muestran una serie de listas desplegables con acceso a las opciones más usuales; en el marco inferior se muestran distintos contenidos según el profesor va eligiendo opciones (la primera vez se muestra el menú completo del módulo del profesor).

Todo el acceso a la base de datos se realiza mediante formularios HTML. Por ejemplo, en la Figura 6 se muestra el formulario que permite añadir una pregunta nueva. Para cada una de las preguntas se introduce la siguiente información:

- Juego al que pertenece la pregunta. Cuando se añade una pregunta nueva sólo se puede elegir un juego, pero una vez añadida se puede asociar a otros juegos.
- Número de pregunta. Si el profesor lo desea, este número se emplea para mostrar las preguntas ordenas según este valor.
- Enunciado de la pregunta. En el enunciado de la pregunta el profesor puede introducir etiquetas HTML.
- Dificultad inicial de la pregunta. Si el profesor lo desea, este valor se emplea para mostrar las preguntas ordenadas según la dificultad. Además, el profesor también puede indicar que este valor se actualice automáticamente a medida que los alumnos responden a la pregunta de forma correcta o incorrecta.
- Explicación. Permite incluir un breve comentario que explica al alumno, en caso de

que no haya respondido correctamente, el motivo de su fallo. A través de esta opción se puede establecer una conexión con los contenidos teóricos necesarios para contestar correctamente la pregunta.

- Tiempo máximo para contestar la pregunta. Si no se introduce ningún valor, se emplea el tiempo máximo por defecto fijado en el MCPS.
- Posibles respuestas. El número de respuestas es configurable a través del MCPS.
- Respuesta correcta.

Una opción muy interesante de AWAM es el menú de resultados. A través de él, se pueden obtener distintos parámetros estadísticos como pregunta mejor o peor contestada, porcentaje de acierto medio por pregunta, juego o tema, etc.

Por ejemplo, en la Figura 7 se puede ver la página que muestra al profesor los porcentajes de contestación por preguntas dentro de un juego de un tema. Otras opciones disponibles son: ordenar las preguntas por porcentaje de fallo (acierto),

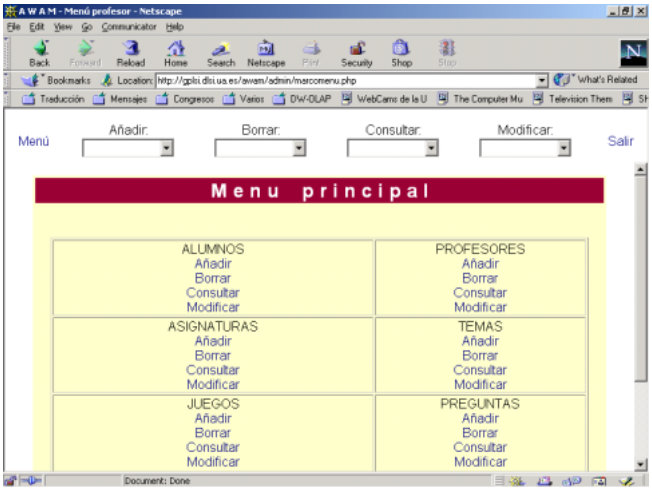


Figura 5. Menú principal del módulo del profesor

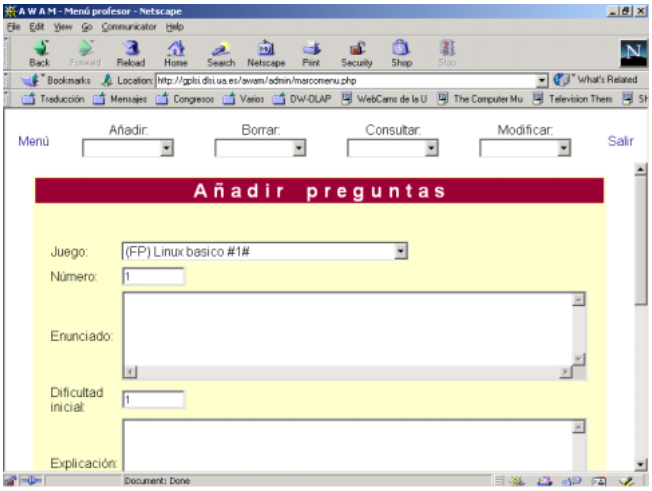


Figura 6. Formulario para añadir una pregunta nueva

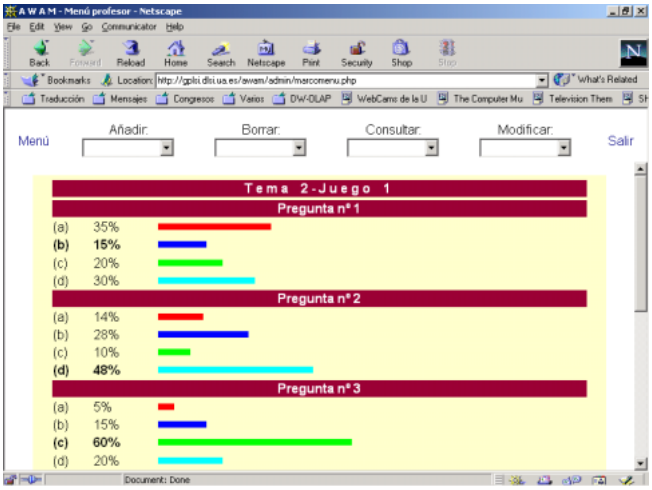


Figura 7. Resultados estadísticos

obtener las preguntas cuyo nivel de dificultad más ha cambiado, ordenar los temas por porcentaje de fallo (acierto), etc.

El profesor también dispone del menú de configuración de parámetros del sistema (MCPS) que permite fijar una serie de parámetros que definen el funcionamiento por defecto del sistema:

- Tiempo máximo para contestar las preguntas.
- Número de preguntas en un juego.
- Número de respuestas por pregunta.
- Posibilidades de error.
- Activar actualización de la dificultad de las preguntas.
- Si se muestra o no la respuesta (y la explicación si existe) cuando un alumno falla una pregunta.
- Modo de juego por defecto: preguntas seleccionadas de forma aleatoria, preguntas ordenadas según la dificultad inicial, según la dificultad actual o según el número asignado por el profesor.

Todos estos valores son por defecto, lo que significa que se pueden variar cuando se desee. Por ejemplo, se puede fijar un tiempo máximo para contestar las preguntas de 30 segundos, pero para algunas preguntas el profesor puede decidir usar otro cantidad.

5. Arquitectura de la aplicación

La aplicación AWAM se puede dividir en 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 los lenguajes HTML y JavaScript. Esta parte de la aplicación se ejecuta en el cliente (navegador web) y se ha intentado hacer lo más compatible posible. Hemos comprobado su correcto funcionamiento en los navegadores más conocidos: Microsoft Internet Explorer, Netscape Communicator y Opera.

La lógica de negocio, encargada de seleccionar las preguntas, verificar las respuestas de un usuario, etc., se ha programado mediante PHP en un servidor web Apache bajo el sistema operativo Linux.

Por último, para la lógica de datos, encargada de gestionar los datos, se ha empleado un servidor de bases de datos MySQL bajo el sistema operativo Linux.

Actualmente, tanto el servidor web como el servidor de bases de datos residen en la misma máquina, pero nada impide que residan en máquinas diferentes.

6. Conclusiones

Las primeras pruebas realizadas parecen indicar que la aplicación incentiva la realización de ejercicios y el aprendizaje de la asignatura. No

obstante, hay que conseguir que el alumno trabaje con la aplicación y que realice los ejercicios de forma conjunta a como va recibiendo los conocimientos en clase de teoría.

Actualmente, el alumno resuelve de forma anónima los ejercicios, aunque estamos planteando la posibilidad de contemplar en la nota de la asignatura la resolución de los ejercicios. No obstante, a pesar de que esto incentivaría a que el alumno entrara en la aplicación a resolver ejercicios, podría ocurrir que los alumnos tuvieran como único objetivo responder correctamente y no aprender.

También consideramos que la aplicación puede suponer una ayuda inestimable en el proceso de aprendizaje/enseñanza. Por un lado, permite que los alumnos puedan darse cuenta de los temas en los que mayores fallos cometen, para poder remediarlo antes de los exámenes. Por otro lado, los profesores pueden hacer mayor hincapié en los temas en los que los alumnos cometen más errores. En cualquier caso, pensamos que esta herramienta no sustituye a las clases presenciales teóricas y/o prácticas ni a los libros de texto, sino que simplemente los complementa. El objetivo que nos hemos marcado con el desarrollo de este sistema es lograr que el proceso de resolución de ejercicios sea lo más ameno posible.

7. Trabajos futuros

Al final de curso deseamos confirmar si los resultados estadísticos obtenidos por el sistema (temas y preguntas con mayor porcentaje de error, etc.) se corresponden con los resultados obtenidos en los exámenes, lo cual permitirá valorar el carácter "predictivo" de la aplicación.

Otro aspecto en el que vamos a trabajar es en ampliar el tipo de preguntas para que no sean únicamente de tipo test sino que incluyan la posibilidad de que el alumno escriba pequeños programas y la aplicación sea capaz de validar su corrección.

Otras mejoras que pensamos llevar a cabo tienen como objetivo aumentar las prestaciones del sistema y su facilidad de uso. Por ejemplo, pensamos incorporar un método adicional de puntuación en el que se tenga en cuenta el tiempo que ha necesitado un alumno para contestar una pregunta (cuanto menos tiempo, más puntuación) y un sistema adicional que permita importar las preguntas y respuestas desde un fichero en formato texto plano.

Aunque esta aplicación ha sido pensada inicialmente para la asignatura *Fundamentos de la Programación*, como la hemos desarrollado con la capacidad de incluir distintas asignaturas y profesores simultáneamente, en el futuro pensamos también aplicarla a otras asignaturas.

Referencias

[1] *Aprendizaje en la Web AutoMotivado* (AWAM). Disponible en Internet (URL): `http://gplsi.dlsi.ua.es/awam/`. Usuario y clave de prueba para acceder como alumno: `jenui/jenui`.

[2] Barchino, Roberto; Gutiérrez, J.M.; García, Elena; Hilera, J. Ramón. *EDVI: Un sistema de apoyo a la enseñanza presencial basado en Internet*. Actas de JENUI 2001, páginas 451-453.

[3] Gayo, Daniel; López, Benjamín; Labra, José E. *Desarrollo del portal web de la E.U. de Ingeniería Técnica en Informática de Oviedo*. Actas de JENUI 2001, páginas 39-44.

[4] Llopis, Fernando.; Pérez, Ernesto; Ortuño, Fernando. *Introducción a la programación. Algoritmos y C/C++*. Publicaciones de la Universidad de Alicante, 2000.

[5] Martel, Ernestina A.; Ojeda, Carmen N.; Hernández, Pablo; Macías, Elsa M.; Monagas, Vidina. *Sistema de Gestión de Asignaturas en Entorno Web*. Actas de JENUI 2001, páginas 33-38.

[6] Más, Ramón; Lacosta, Ignacio. *Aplicaciones de Internet a la Enseñanza: Un Sistema de Autoevaluación.* Actas de JENUI 2001, páginas 500-503.

[7] Merelo, Juan Julián; Castillo, Pedro Ángel; Prieto, Alberto. *Integración de una asignatura en Internet: el caso de Diseño y Evaluación de Configuraciones*. Actas de JENUI 2001, páginas 51-56.

Prototipo de entorno docente virtual adaptable por niveles

José-Carlos Sánchez Alonso
Dpto. de Informática
Universidad de Extremadura
10071 Cáceres
e-mail: jcsanchez@unex.es

Resumen

En este trabajo se presenta un sistema en desarrollo para la generación de cursos, control docente y evaluación automática de alumnos a través de Internet. Este sistema consta de un conjunto de herramientas software que, residiendo en un servidor, permite generar contenidos docentes con un formato estandarizado para su adaptación a distintas materias curriculares, sin necesidad de grandes conocimientos de Informática. Parte de estas herramientas software pueden ser utilizadas en cualquier computadora para preparar los contenidos de los cursos. El resto de las herramientas adaptan el curso para su uso en Internet, combinando diversas técnicas (HTML, JavaScript, CGIs,…) junto con software desarrollado por casas comerciales.

El sistema queda, pues, constituido por tres módulos. El primer módulo es la página Web desde donde los usuarios pueden matricularse en los distintos cursos que se ofrecen, acceder a los contenidos de los mismos e informarse de las últimas novedades. El segundo módulo sirve para que el docente pueda desarrollar los contenidos de los cursos y evaluarlos después del proceso de aprendizaje por parte de los alumnos. El tercer y último módulo es un sistema público de preguntas y respuestas, donde los alumnos pueden realizar consultas. El sistema se encuentra actualmente en fase de desarrollo.

1. Introducción

La enseñanza a distancia a través de Internet es un tema de máximo interés actual en nuestra sociedad. No está en sus inicios, sino en una fase de clara expansión. El aprovechamiento de las nuevas tecnologías para su aplicación en la enseñanza determinará, en un futuro no muy lejano, la diferencia entre las sociedades bien informadas y educadas, con las que se vayan quedando rezagadas.

En España disponemos de un ejemplo claro, la UOC (*Universitat Oberta de Catalunya*) [15], una auténtica universidad pública (con sus títulos homologados, su profesorado, sus materias, sus evaluaciones, etc.), y gestionada a través de Internet, eliminando aulas y mobiliario, optimizando recursos y reduciendo al mínimo los costes de las universidades modernas.

Otros ejemplos los podemos encontrar en multitud de programas de enseñanza "on-line", abarcando campos tan dispares como la educación a distancia en ámbitos rurales, las prácticas tele-asistidas en Institutos, etc.

Es, por tanto, un momento oportuno para implicarse en estas tareas, dedicando esfuerzos en la elaboración de propuestas y prototipos de enseñanza a través de Internet. El grado de innovación y originalidad dependerá de la capacidad de integrar propuestas en un marco amplio de aplicación, de forma que los contenidos docentes a generar y gestionar "on-line" puedan ser utilizados por todos los estamentos docentes (Universidades, Institutos, Colegios, Academias, Instituciones, etc.) en sus tareas de docencia, formación continua, etc.

Con la propuesta que aquí se presenta, se pretende cubrir la falta de explotación, que a mi juicio, existe de las herramientas software comerciales o no, que trabajan en este sentido, y su adecuación al ámbito docente real. Convencido de que un gran número de organismos (Universidades, Institutos, Colegios, Academias y otras instituciones con cuerpos docentes) están

muy interesados en comprobar cómo se aplican estas experiencias.

2. Características

A continuación enumeramos las características más relevantes del sistema:

- Para realizar las pruebas iniciales del sistema, éste puede residir y administrarse en un PC, configurado como servidor (Windows NT/2000) [1] o Linux [16] sobre Internet, como se ve en la Figura 1.
- Permite generar contenidos docentes (por ejemplo, cursos), que se almacenan en su base de contenidos con una estructura determinada. Estos contenidos pueden generarse en cualquier PC, y ser posteriormente administrados por el servidor.
- Los contenidos pueden ser cursos, cuestionarios, seminarios, ..., cualquier tipo de contenido docente del que interese conocer cómo lo utilizan y aprovechan los usuarios (*evaluación automática*).
- Los contenidos docentes deben ser accesibles desde cualquier computador conectado a Internet, desde el que se puede seleccionar un determinado contenido docente e interactuar con él.
- Los resultados de esta interactividad (por ejemplo, respuestas a las preguntas, tiempo de ejecución, porcentajes de éxito, evaluaciones, etc) son gestionados por el servidor
- El servidor monitoriza y almacena, en diversos formatos, los resultados del proceso de aprendizaje del curso, seminario, … Los docentes, solamente desde este servidor, pueden conocer parámetros relativos al aprendizaje de los cursos seleccionados.
- Los cursos generados podrán ser también almacenados en CD para su utilización sin soporte de Internet.

3. Objetivos

El objetivo principal de un Entorno Docente Virtual, como el que se describe en el presente documento, es implantar un sistema de bajo coste y sencillo que permita llevar a cabo un seguimiento de los progresos de los alumnos.

Durante el transcurso de una sesión de tele-enseñanza, este sistema puede obtener información relativa a la fecha y la hora a la que se comenzó y terminó un curso, datos del usuario, preguntas contestadas/no contestadas, índice de aciertos, contenidos más veces consultados, etc. De esa información, el sistema elabora estadísticas tales como: número de alumnos evaluados, porcentajes de éxito, tiempos medios de respuesta, generación de notas y puntuaciones de las pruebas y exámenes, etc.

De todo esto se deduce que este tipo de Entornos Docentes Virtuales son idóneos para su aplicación en:

- La evaluación de la docencia y calidad docente de los centros educativos, aplicables tanto a alumnos, como a profesores.
- El reciclamiento docente del profesorado mediante seminarios que pueden cursar cómodamente y en cualquier horario.
- La realización y evaluación de prácticas a distancia.
- Promoción de cursos.
- Dar a conocer publicaciones, artículos y todo aquello que se considere de interés general para el alumnado, profesorado, etc.

Además del objetivo principal, descrito anteriormente, se pretenden conseguir objetivos tales como la exportabilidad al sistema educativo de enseñanzas primarias, medias y universitarias, haciendo útil las nuevas tecnologías en la enseñanza; la reducción de costes sobre la educación tradicional; la capacidad de adquisición de las herramientas comerciales; la obtención de presentaciones consistentes que integren todas las características multimedia a la enseñanza, unidas a la facilidad de manejo. Y por último, cabe destacar la reducción en el tiempo de aprendizaje y en mejor rendimiento del estudiante (se ha demostrado que las aplicaciones interactivas mejoran el aprovechamiento en un 25% sobre el aprendizaje convencional).

4. Desarrollo

Los inicios de este proyecto se enmarcan dentro del I PRI+DT (I Plan Regional de I+D

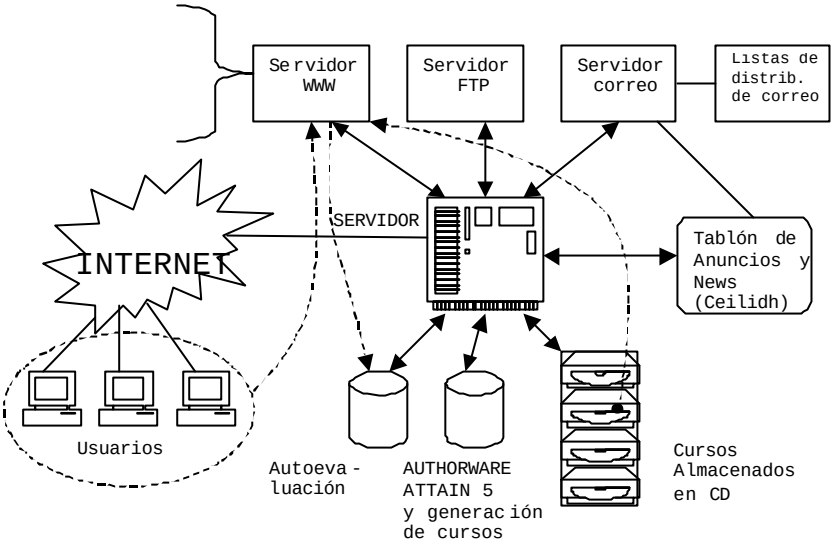


Figura 1. Esquema del sistema EDONET

Tecnológico) de la Junta de Extremadura y el Fondo Social Europeo, en el año 2000, con el proyecto «Sistema piloto para la docencia a través de Internet», donde se desarrolló el sistema en pruebas EDONET (Entorno para la Docencia sobre InterNET). En la figura 1 se aprecia el esquema de funcionamiento de EDONET.

Este sistema se implementó sobre un PC con Windows NT Server 4.0 ™, disponiendo así de un servidor utilizado como soporte al sistema docente a través de Internet. El servidor ofrece servicios de administración y publicación de páginas web (WWW), de transferencia de archivos (FTP), correo electrónico (POP3 y WebMail) y listas de distribución de correo compatible con Majordomo. También posee diversos dispositivos multimedia: webcam, capturadora de vídeo, ... para la realización y composición de vídeos explicativos y sesiones prácticas que puedan contener los cursos. Así mismo, se dispone de software comercial para el diseño de páginas web, sistemas de preguntas y respuestas, exámenes electrónicos y elaboración de contenidos docentes.

Todos estos elementos se integran en un único sistema, que funciona como un todo, de forma transparente al usuario, constituyendo un sistema que aúna todos los requisitos esperados. La integración y el desarrollo del sistema, así como las ampliaciones y mejoras del proyecto que se están llevando a cabo a día de hoy, se centran, sobretodo, en la elaboración de una serie de *rutinas* y una *jerarquía de librerías y objetos* que contengan *modelos y esqueletos* de fácil aplicación para la construcción de contenidos docentes, *minimizando* el tiempo de desarrollo de cursos, seminarios, pruebas, ...

4.1. Punto de acceso al sistema

La gran telaraña mundial (WWW) se ha convertido en el medio más difundido de acceso a la información a través de Internet, debido sobre todo a su capacidad para mostrar contenidos en multitud de formatos, su interactividad y su fácil acceso por parte de todos los usuarios de la Red, tanto desde redes privadas, como desde el hogar. Por este motivo, es uno de los medios más idóneos para la promoción y difusión de la educación a distancia. Los usuarios pueden decidir en qué momento desean estudiar y qué contenidos son los que más les interesan, todo ello de una forma muy sencilla.

Debido a estas características, se hace patente la necesidad de implantación de sistemas docentes accesibles a través del web.

Mediante la utilización de técnicas de diseño web: lenguaje HTML [2][5][6], Java [2], JavaScript [7][8] y CGIs [2][3][4], se confecciona la página web principal que permite el acceso al sistema, y desde donde los usuarios pueden:

- Matricularse en los cursos que le interesen.
- Acceder al contenido docente de los cursos, tanto si son «on-line», como «off-line».
- Ser evaluados y supervisados en tiempo real.
- Permanecer informados de las últimas novedades.
- Participar de forma activa en foros de debate e intercambiar información con el resto del alumnado y profesorado.
- Descargar material didáctico del servidor.

En la figura 2 se observa un ejemplo de edición de un curso en el que puede apreciarse una secuencia de vídeo.

4.2. Generación de contenidos docentes

La generación de los contenidos de los cursos se realiza con una herramienta de software de autor comercial, con licencia especial para educación. Mediante *Macromedia Authorware 5 Attain* [14] se preparan, generan y gestionan cursos o cualquier tipo de contenido docente. Debido a las características de esta herramienta, se permite la integración de textos, imágenes, vídeos, sonidos, animaciones, … configurados como cursos, por parte de personas poco expertas en informática, para su posterior portabilidad y adecuación en el sistema. Esta herramienta permite tanto la edición de cursos para su implantación en un servidor WWW, visible desde cualquier navegador, como su preparación para la distribución en formato CD.

Para ello, se están desarrollando una serie de estructuras genéricas y librerías con una serie de elementos en *Authorware* [12][13][14] comunes a todos los cursos, que sirvan como esqueleto en el diseño de los contenidos docentes, de modo que su integración en el sistema se haga de una forma sencilla, eficaz y rápida, con un coste de desarrollo mínimo. De este modo, todos los cursos

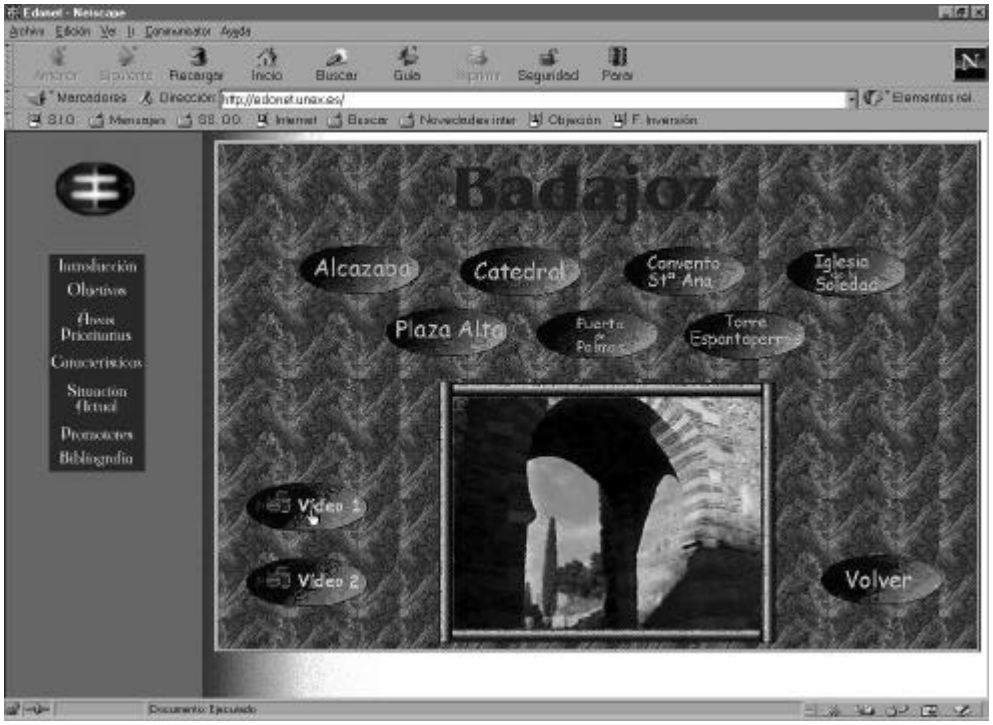


Figura 2. Ejemplo de curso interactivo a través de la red

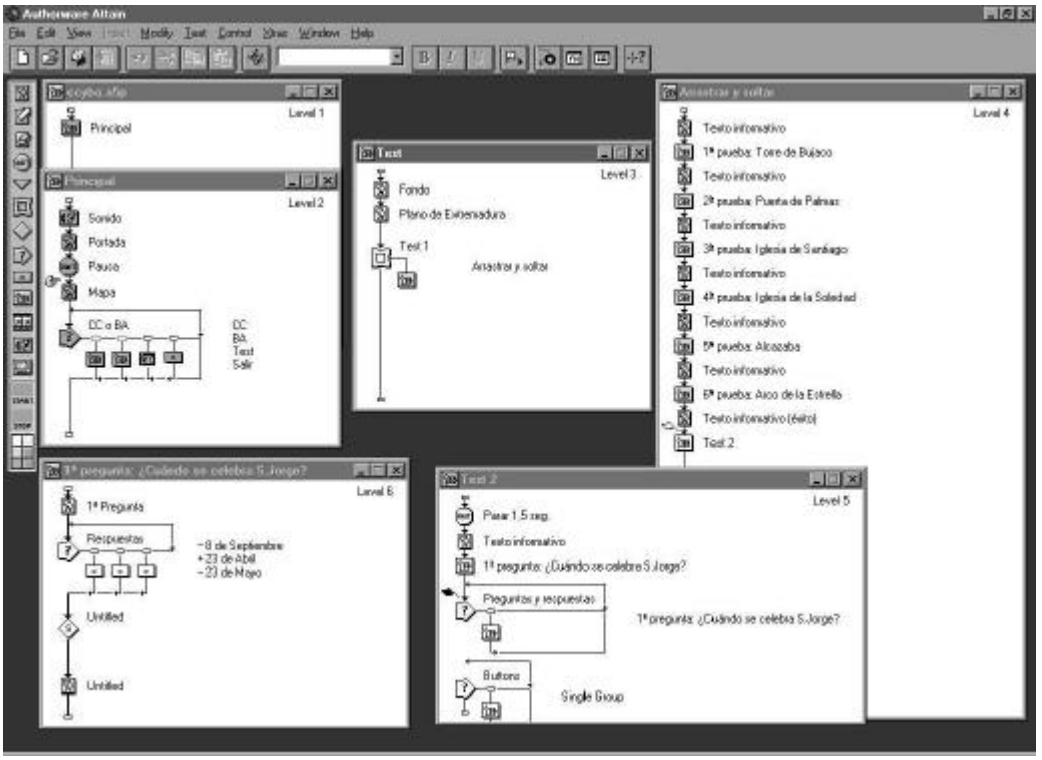


Figura 3. Estructura y uso de librerías diseñadas mediante Authorware Attain 5

siguen unos mismos estándares, bastante flexibles, que permiten, entre otras cosas:

- Desarrollo de una interfaz común a todos los cursos.
- Botones de navegación, búsqueda, avance de página, …
- Enlaces a otros cursos, bibliografía, páginas Web, …
- Navegación hipertextual inteligente, ya sea de forma local al propio curso o bien a otro curso existente en el sistema.
- Marcos para imágenes estáticas, imágenes en movimiento, vídeos, …
- Generación de índices.
- Control de acceso a los cursos (nombre de usuario y contraseña).
- Inclusión de diversos formatos sonoros.
- *Frames* para rótulos, avisos, alertas, notas, …
- Otras clases de elementos informativos.

En la figura 3 se puede observar un ejemplo de las estructuras implementadas con *Authorware* para el diseño de cursos. En concreto es el desglose de esta jerarquía de estructuras y librerías que se están actualmente diseñando, y que han sido utilizadas en el ejemplo de la figura 3 para demostrar lo sencillo que resulta montar una prueba *autoevaluable* como la que se muestra en la figura 4.

Los componentes realizados con *Authorware* siguen un diseño *top-down*, es decir, cada componente de nivel N, se desglosa al hacer doble clic sobre él, en una serie de componentes de nivel N+1. Cada uno de esos componentes de nivel N+1, se descomponen a su vez en más componentes de nivel N+2, y así sucesivamente, aumentando progresivamente el nivel de detalle.

Figura 4. Ejemplo de realización de pruebas y de evaluación automática

En la figura 3 podemos observar cómo el *nivel 1* corresponde a un único componente que representa la totalidad del curso. Haciendo doble clic sobre este elemento, se despliega el *nivel 2*, compuesto por las distintas secciones del curso. Dentro de estas secciones, se encuentra *test*, cuyo contenido de *nivel 3* muestra sus características internas. En el *nivel 4* aparecen las preguntas que componen el *test*. Y así podemos continuar hasta el nivel de detalle que queramos.

Las librerías y estructuras de cursos que se están diseñando, comprenden multitud de niveles y subniveles. De este modo, un usuario poco experto, puede utilizar componentes de alto nivel para generar sus contenidos docentes con muy poco esfuerzo y dificultad. Si por el contrario, el usuario ya tiene cierta experiencia, puede ahondar un poco más en el desarrollo de los contenidos y realizar un curso a su medida, configurando multitud de parámetros.

Sin embargo un sistema de tele-enseñanza no estaría completo si no se dispusiese de un sistema que permitiera la evaluación de los alumnos cuando realizan un curso por la red. Mediante esta herramienta se posibilita un sistema de evaluación "on-line", capaz de llevar a cabo una comunicación automática entre el alumno y el curso, permitiendo la tutorización y corrección en los momentos oportunos.

El control y la evaluación se realizan en tiempo real, gestionando la información relativa al estado de utilización del sistema en un momento determinado, o bien, durante un periodo de tiempo concreto. De este modo se pueden conocer ciertos detalles de interés, tales como: el número de alumnos que han accedido a un determinado curso en un instante o periodo de tiempo concreto, el tiempo que invirtió cada alumno en completar un seminario docente, número de respuestas acertadas/falladas, número de intentos, tiempo invertido en resolver las cuestiones propuestas, etc.

En la figura 4 puede verse un sencillo ejemplo de supervisión por parte del sistema. Toda la información de control generada como consecuencia de la interactividad del usuario con el sistema, puede registrarse de forma totalmente transparente para él o no, según convenga. En ocasiones, puede resultar interesante que el alumno conozca el índice de aciertos o el grado de avance en un determinado curso. Esto se contempla en las estructuras que se están diseñando. Sin embargo, en otras ocasiones puede interesar que toda esa información se almacene internamente, sin que el usuario tenga conocimiento de ello. Esta información puede resultar de gran interés para los educadores, de hecho, constituirán los únicos elementos de juicio de que dispongan para poder evaluar el rendimiento o aprovechamiento del curso por parte del alumno, y esto habrá sido generado por el propio sistema de forma automática. Se asegura la confidencialidad de los datos obtenidos, mediante los métodos telemáticos oportunos, enviándose dichos resultados al sistema central.

4.3. Intercambio y participación

Al igual que sucede en un sistema educativo tradicional, se hace patente la posibilidad de poderse poner en contacto con los educadores, o bien con otros alumnos que siguen el mismo curso, para intercambiar impresiones, participar en foros de debate o simplemente resolver una duda. Internet ofrece numerosas posibilidades para que dos o más personas puedan entablar una conversación o entrar en contacto, sin embargo no todas son las más idóneas para sistemas basados en la tele-enseñanza. Correo electrónico, videoconferencias, chats, etc., son formas conocidas de establecer conversaciones, pero no se ajustan adecuadamente a los objetivos que se pretenden.

Un sistema adecuado para resolver este tipo de problemas de comunicación e intercambio de información entre alumnos y educadores es un *tablón electrónico*. La filosofía del tablón electrónico es similar a la de las *news* en Internet. Se dispone de un lugar en el sistema donde se envían correos electrónicos, de modo que esos correos electrónicos se hacen públicos, y pueden ser leídos por cualquier integrante de un curso. Además, permanecen un tiempo indefinido en el sistema (al menos hasta que el curso se dé por finalizado). De esta manera, un alumno que plantee una duda o problema al tablón electrónico, podrá obtener respuesta, y éstas (pregunta y respuesta) podrán ser consultadas no sólo por el alumno que las formuló, sino por todos aquellos alumnos matriculados en el mismo curso. Esto agiliza enormemente la labor de tutorización que ejercen los educadores.

Para la implantación de un sistema de tablón electrónico, se ha optado por la utilización de *Ceilidh* [10][11], una herramienta de libre distribución a través de Internet. *Ceilidh* permite realizar réplicas y contrarréplicas, y encadenar mensajes con otros concernientes al mismo tema de una forma visual. Para enviar un mensaje al tablón de anuncios, se procede de la misma manera a como se haría para enviar un correo electrónico a otra persona, sólo que aquí la dirección del destinatario corresponde a una dirección específica para el tablón electrónico. Sin embargo, el usuario no tiene por qué preocuparse de esto, puesto que puede hacerlo desde el propio tablón. Los mensajes contenidos en el tablón electrónico se visualizan a través del navegador, como si de una página web se tratase. Los mensajes aparecen encadenados por temas: un mensaje, su correspondiente contestación anidada a dicho mensaje, e incluso una contestación anidada a la contestación anterior. No hay límite.

Cuando un usuario quiere enviar una réplica o contrarréplica al tablón, simplemente debe seleccionar el mensaje correspondiente de la lista despleglable y contestarlo.

5. Conclusiones

En este trabajo se presenta el estado del arte de un sistema que se podrá utilizar para la generación de material docente, el control y la evaluación de su aprendizaje. El sistema, que aún se encuentra en fase de desarrollo, permitirá la realización de experimentos de tele-enseñanza a través de Internet en breve.

Se busca una forma sencilla de implementar un sistema de bajo coste, que pueda ser utilizado por personas no expertas en informática (aunque sí con ciertos conocimientos) sin dejar de lado todas las posibilidades que pueden ofrecer otros sistemas similares existentes más complejos.

Con todo, también tiene sus limitaciones: Puede parecer excesiva la centralización de todos los servicios (servidor de páginas web, de FTP, de correo, …) en una única máquina, en lugar de mantenerlos distribuidos en varios equipos. Tampoco se hace mención de los mecanismos de seguridad de red (firewall, proxy, ...), aunque sí se emplea cierto software de libre distribución para la protección del servidor [17].

Recientemente también han aparecido en el mercado nuevas herramientas de autor a precios competitivos, por lo que sería conveniente comenzar un estudio comparativo.

Referencias

[1] J.M.Álvarez, P.Díaz, G.Galeano, N.Pavón, J.C.Sánchez. *Windows 2000 Professional.* Anaya Multimedia, 2000. ISBN: 84-415-1015-6.

[2] Stanek, William Robert; Ketzler, Mark; DeRose, Steven J. *HTML, Java, CGI, VRML, SGML Web Publishing Unleashed.* Sams, 1996. ISBN: 1575210517

[3] Stein, Lincoln. *Official Guide to Programming With Cgi.Pm.* John Wiley & Sons, 1998. ISBN: 0471247448

[4] Weinman, William E. *The CGI Book.* New Riders Publishing, 1996. ISBN: 1562055712

[5] Heslop, Brent D. *Html Publishing on the Internet : Covers Html 4 and Dynamic Html : Everything You Need to Create Professional-Looking Web Page.* Ventana Communications Group Inc., 1998. ISBN: 1566046254

[6] G.Galeano, P.Díaz, J.C.Sánchez. *HTML 4.* Anaya Multimedia, 2000. ISBN: 84-415-1024-5.

[7] Wooldridge, Andrew; Morgan, Mike; Darnell, Rick; Honeycutt, Jerry; Reynolds, Mark C. *Using Javascript (Special Edition Using...).* Que, 1997. ISBN: 0789711389

[8] McComb, Gordon. *Javascript Sourcebook : Creative Interactive Javascript Programs for the World Wide Web.* John Wiley & Sons, 1996. ISBN: 0471161853

[9]Cranford Teague, Jasón. *How to Program HTML Frames : Interface Design and Javascript.* Ziff Davis Pr., 1997. ISBN: 1562764950

[10]Hughes, Richard J.; Shewmake, Jake; Okelberry, Christopher R. *CEILIDH: Collaborative Writing on the Web* (*Utah State University*). Atlanta, GA (U.S.A.). The Association for Computing Machines's 1998 Symposium on Applied Computing, Feb 28, 1998.

[11]Ceilidh Web Site: *http://www.lilikoi.com/*

[12]Roberts, Nick. *The Official Guide to Authorware 4 : The Comprehensive Reference from the Multimedia Labs of Macromedia.* Peachpit Press, 1997. ISBN: 0201688999

[13]Kellog, Orson; Ziajka, Judy. *Authorware 4 Authorized.* Peachpit Press, 1997. ISBN: 0201696339

[14]Kellog, Orson; Ziajka, Judy.*Authorware 5 Attain Authorized.* Peachpit Press, 1998. ISBN: 0-201-35411-X

[15]Universitat Oberta de Catalunya: *http://www.uoc.es*

[16] Gazo Cervero, A.; González Sánchez, J.L. *Manual Avanzado de Red Hat Linux 7.* Anaya Multimedia, 2001. ISBN: 84-415-1133-0

[17] ZoneAlarm, de *ZoneLabs. http://www.zonelabs.com*

SHAAD: sistema hipermedia adaptable, adaptativo y dinámico para la entrega de contenidos hipermedia

David Mérida, Ramón Fabregat
Institut d'Informàtica i Aplicacions (IIiA)
Universitat de Girona (UdG)
david.merida@udg.es, ramon.fabregat@udg.es

Resumen

Para implementar una eficiente adaptación de contenidos en la red se pueden analizar diversas variables que involucran a los tipos de usuarios, los dispositivos de acceso del cliente, los tipos de acceso, el estado de la red y el estado de carga del servidor. La creciente utilización de objetos hipermedia para la creación de contenidos y la no consideración de estas variables, ha ocasionado en muchas situaciones el problema de la entrega de contenidos inadecuados. En este paper se propone un modelo para la entrega de contenidos hipermedia que considere estas variables. Este modelo, elaborado a partir de un análisis de diversas investigaciones, considera los diferentes puntos de vista observados y tiene como objetivo final definir un Sistema Hipermedia Adaptable, Adaptativo y Dinámico (SHAAD) como punto de partida para la consideración del problema de adaptación de contenidos a través de un sistema totalmente modular.

1. Introducción

Actualmente, la gran heterogeneidad en términos de tipos y capacidades de los dispositivos de acceso, ancho de banda de la red y necesidades-preferencias de los usuarios no son tenidas en cuenta por un servidor cuando sirve contenidos web ricos en imágenes, audio y video. Por ejemplo, el servidor entregará el documento solicitado aunque el terminal (WebTV, Personal Digital Assistants (PDAs) o teléfonos móviles) utilizado no pueda acceder a estos contenidos debido a las limitaciones del monitor, de las capacidades de almacenamiento, de procesamiento o de acceso a la red.

Para solucionar este problema se deben desarrollar alternativas que permitan un acceso universal a cualquier tipo de material, desde cualquier tipo de dispositivo y que tengan en cuenta las preferencias del usuario y el estado de carga de la red y del servidor.

El concepto de *Adaptación* ha sido muy estudiado en el campo de los sistemas hipermedia y se ha mostrado que dentro de estos ambientes puede proporcionar mejores entornos de utilización y rendimiento. Muchos son los grupos dedicados a la tarea de solucionar la problemática de la adaptación de contenidos y diferentes también las consideraciones hechas al respecto a la hora de implementar dicha adaptación:

- UMA (Universal Multimedia Acces) [1] tiene en cuenta las nuevas clases de dispositivos inteligentes y portables. La finalidad del proyecto es permitir a los dispositivos de acceso con limitaciones en las capacidades de comunicación, procesamiento y visualización acceder a cualquier contenido multimedia.
- MONADS [3][7][10] está dirigido a solucionar la demanda de los servicios de datos "anytime-anywhere-anyhow" a través de las nuevas tecnologías de dispositivos móviles. La adaptabilidad de los servicios de datos a los cambios de entorno de los usuarios nómadas es el tema principal de este proyecto. Se entiende por nómadas a los usuarios que cambian permanentemente de lugar y posiblemente de dispositivo y de tipo de acceso. En este proyecto los agentes inteligentes juegan un papel muy importante en la implementación de la adaptabilidad.

- Grupos de investigación de Hewlet-Packard [13][14] y de Microsoft Research China [7] trabajan en Adaptive Delivery Systems y tienen en cuenta el tipo de dispositivo de acceso, el estado de la red y las preferencias del usuario.
- Paul de Bra [6] a partir de un Adaptive Hipermedia System (AHS) y teniendo en cuenta las preferencias del usuario (modelo del usuario) se plantea el cambio de los contenidos o de la presentación de los nodos alterando la estructura de los enlaces.

En los trabajos anteriores se han considerado diversas variables que influyen en la adaptación de contenidos. Todas ellas serán adoptadas en este trabajo:

1. *Las características y preferencias del usuario* [2][6][8]. Bajo este concepto se incluye todo lo referente a preferencias y conocimientos del usuario. Siendo preferencias lo relativo a la manera de recibir el material hipermedia (p. ej. explicita o resumida), a las características de aprendizaje (p. ej. si es un usuario textual o visual), a las características personales (p. ej. si es un usuario extrovertido o introvertido), etc. Y conocimiento es todo lo relativo a conocimiento previo o evolutivo.
2. El *dispositivo de acceso del cliente* [1][3][7][10][13][14]. Paralelamente a la creciente explosión de Internet ha devenido un amplio desarrollo tecnológico de los dispositivos de acceso a la red (desde los tradicionales ordenadores de escritorio hasta la nueva generación de dispositivos móviles y PDAs). Estos dispositivos presentan grandes diferencias tales como su capacidad de almacenamiento, poder de procesamiento, resolución de pantallas, etc. Nos encontramos ante una nueva problemática: el tener que entregar los ricos contenidos multimedia disponibles a una amplia gama de dispositivos/clientes que en muchos de los casos no tendrán capacidades para tratarlos adecuadamente.
3. *Tipo de acceso a la red* [13][14]. Anchos de banda que van desde los 9.6K o 28.8K de las conexiones para telefonía celular o módems; pasando por los comprendidos entre los 128K u los 1.5M de las conexiones ISDL, DSL y cable-módems y los de 10M y 100M de las redes de área local o Ethernet, generan un nueva característica que hay que considerar: *la velocidad de acceso a la red*.
4. *Estado de la red* [4][5][6]. A la diversidad de tipos de conexión se suma el estado de congestión que la conexión utilizada puede presentar en un momento dado. La carga que presenta la conexión en todo momento no es uniforme. Es decir, no podemos inferir a partir del hecho de disponer de un buen ancho de banda, el tener en forma permanente un buen estado de la conexión.
5. *Estado de carga del servidor* [11][12]. Por parte de los usuarios es más que conocida la experiencia de situaciones de accesos fallidos o rechazados por parte del servidor ante la consulta de algún material multimedia. Esta situación muchas veces es ocasionada por un exceso de carga en dichos servidores como consecuencia de una inusitada cantidad de peticiones. Para solucionar dicha sobrecarga puede ser preferible entregar contenidos con recursos de menor calidad antes que tener que rechazar o generar una falla en las conexiones que se están realizando.

La sección 2 plantea una revisión sobre adaptabilidad versus adaptatividad a fin de llegar a un acuerdo sobre la terminología a utilizar. En la sección 3 definimos nuestro SHAAD. En la sección 4 se plantean diversas variantes del modelo para las diferentes variables de adaptación y finalmente se presenta un modelo único para nuestro sistema.

2. Adaptabilidad versus adaptatividad

El término de *adaptatividad* ha sido utilizado por distintos autores en diferentes ámbitos. Dependiendo del entorno considerado varían los objetivos perseguidos por tal adaptación. Por esta razón se hace necesario en primer lugar llegar a un acuerdo respecto a que entendemos por *adaptación* de contenidos hipermedia en un entorno de variables cambiantes como las mencionadas en el apartado anterior.

Abdelzaher [12] considera la adaptación de contenidos web como un mecanismo para mejorar el rendimiento del servidor sobrecargado

Wei-Ying Ma [13] hace consideraciones sobre la entrega de contenidos adaptados en ambientes heterogéneos a fin de mejorar la accesibilidad de los contenidos.

Brusilovsky [8] define un Sistema Hipermedia Adaptativo como aquel que construye para cada usuario un modelo de objetivos, preferencias y conocimientos. Utiliza este modelo a través de la interacción para adaptarse a las necesidades del usuario.

Oppermann [9] tiene en cuenta las características del usuario y distingue entre:

- *Sistemas Adaptables*: permiten al usuario cambiar ciertos parámetros del sistema y adaptar de esta manera su comportamiento.
- *Sistemas Adaptativos*: Se adaptan al usuario automáticamente basándose en las suposiciones que el sistema realiza de las necesidades del usuario.

Por otro lado, De Bra [6] teniendo en cuenta las preferencias del usuario como variable que decide la adaptación, clasifica los entornos hipermedias o sitios Web construidos con la capacidad de realizar algún tipo de personalización:

- *Hipermedias adaptables*: sistemas en los que el usuario puede proveer algún perfil (p.ej. a través de cuestionarios) para que el sistema pueda proveer una versión de la aplicación hipermedia teniendo en cuenta ese perfil.
- *Hipermedias adaptativos:* sistemas que monitorean el comportamiento de los usuarios y adaptan la presentación teniéndolo en cuenta. En estos casos la evolución, tanto en preferencias como en conocimientos, podría ser captada o deducida por el sistema a partir de los accesos realizados a las páginas. Por otra parte, muchas veces podrían necesitar cuestionarios o test a fin de obtener información más fiable respecto a las preferencias del usuario. La mayor parte de la adaptación es realizada, sin embargo, en base a las acciones de navegación del usuario y también considerando el comportamiento de otros usuarios.
- *Hipermedias dinámicos*: El comportamiento de los usuarios es monitoreado como en los sistemas hipermedia adaptativos pero la adaptación en lugar de ser cambiada seleccionando presentaciones predefinidas es generada a partir de unidades atómicas de información. Es decir, es reconstruida dinámicamente a partir de los objetos individuales que componen la página tomando los que son mas adecuados en base a las características del usuario.

3. Definición del SHAAD

A partir de lo comentado en el apartado anterior se define un Sistema Hipermedia Adaptable, Adaptativo y Dinámico (SHAAD) como *aquel sistema que atendiendo al estado de las variables mencionadas y a la variedad multimedia que presentan los contenidos web, intenta adecuar dinámica o estáticamente la información disponible y entregarla de la manera más eficiente posible.*

El modelo está formado por 4 módulos:

1. *Mecanismos para la definición de variables*. Tienen como finalidad realizar la definición de las variables ya mencionadas anteriormente: características y preferencias del usuario, dispositivo de acceso del cliente, tipo de acceso a la red, estado de la red y estado de carga del servidor.
2. *Módulo de contenidos*. Tiene como función entregar los contenidos peticionados, ya sea a través de una generación dinámica a partir de los elementos atómicos que constituyen la página web (generación on-line) o la selección entre diferentes versiones estáticas de esos contenidos previamente generados, (generación off-line).
3. *Motor de decisión*. Es el núcleo del sistema y el lugar en el que se evalúan las variables de decisión y los contenidos disponibles, a partir de los que se infiere cuales son los mecanismos para entregar el material en la forma más adecuada al usuario final.
4. *Mecanismos de adaptación*. Teniendo disponible el nuevo sitio web generado por el módulo de contenidos, implementan los

mecanismos de adaptación decididos por el Motor de decisión.

4. Descripción de las variables

A fin de realizar una descripción detallada de los aspectos a considerar en cada variable, éstas se analizan por separado y se define el modelo utilizado en cada caso.

4.1 Características y preferencias del usuario

De forma muy general, los conceptos que nos permiten describir a nuestro usuario son: objetivos vs. tareas, conocimiento previo o background, experiencia, preferencias en el modo de recibir la información, intereses, rasgos del usuario, etc.

Estos conceptos tienen en cuenta diferentes *tipos* de características [8]: las relativas al conocimiento (previo o evolutivo), las que consideran las preferencias sobre la forma de recibir la información (gráfica o textual, desarrollada o resumida), las que tienen en cuenta la personalidad del usuario (introvertido o extrovertido, verbal o textual, ...), etc.

Los módulos particulares de este modelo son (figura 1):

- *Mecanismos para la definición de las características y preferencias del usuario*:
  1. *Cuestionarios* de respuesta directa por parte del usuario a fin de obtener las características y preferencias del mismo.
  2. *Monitoreo* del comportamiento del usuario para deducir las características del mismo a partir de su interacción. Analizando la manera en que se accede o navega a través del material disponible en el sitio Web el sistema intenta inferir los rasgos que definen al usuario.
- *Mecanismos de Adaptación* [6]:
  1. *Adaptación de enlaces*. El sistema intenta guiar al usuario hacia los aspectos más importantes de la información disponible, apartándolo de los menos importantes. Este objetivo se consigue manipulando la estructura de los enlaces o la presentación de los mismos.
  2. *Adaptación de contenidos*. El Sistema Hipermedia Adaptativo proporciona información adicional o alternativa a fin de asegurar al usuario el perfecto entendimiento del material expuesto.

La terminología encontrada en otros autores puede diferir de la expuesta, pero los mecanismos implicados no difieren sustancialmente con respecto a los aquí presentados.

4.2 Acceso a la red: Dispositivo del cliente y Tipo de Acceso / Estado de la Red

Aunque estos tres conceptos son diferentes los consideramos juntos pues están íntimamente relacionadas en lo que a tecnología se refiere. De estas variables dependerá, en última instancia, la *calidad del formato del contenido* que se entrega al usuario.

La gama de tipos de dispositivos de acceso a la red existentes es muy amplia. Podremos resaltar diferencias tales como la potencia de procesamiento de información, la disponibilidad de reproductores de sonido, la interfaz de acceso a la red y fundamentalmente el tipo de medio que utiliza para video. Esta última característica juega

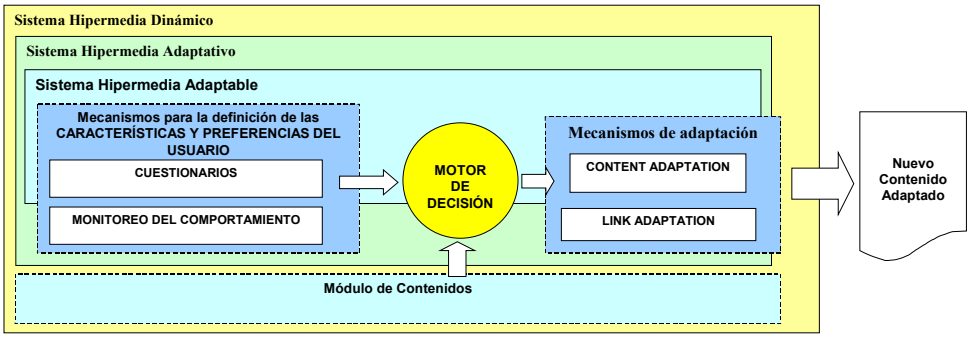


Figura 1 – Modelo para Adaptación de Contenidos según las características y preferencias del usuario

un importante rol a la hora de pensar en una adaptación de los contenidos.

Además, estos dispositivos tienen una estrecha relación con el tipo de conexión que disponen y tienen una diferencia muy importante en lo que a velocidad de acceso se refiere.

A estas consideraciones debemos agregar además: *el estado de la red* en un momento dado.

De la experiencia como usuarios, sabemos que de la relación *excelente dispositivo de acceso / excelente tipo de conexión* no tenemos necesariamente como resultado una *excelente disponibilidad de contenidos*. Más aún, bajo ciertas condiciones muchas veces hemos experimentado la frustración de no poder acceder al sitio web solicitado. Esto puede ser originado por el estado de la red.

Para este modelo es necesario describir particularmente dos módulos (figura 2):

- *Mecanismos para la definición de las características del dispositivo del cliente y el tipo de acceso y estado de la red.* Este módulo cumple una función parecida al módulo de *Mecanismos para la definición de las características del usuario* de la figura 1. Los métodos disponibles y que estamos considerando para la determinación del tipo de dispositivo, conexión y estado de la red son [13]:

  1. *Protocolo http*. En el protocolo utilizado para la entrega de documentos, la cabecera de las peticiones http contienen información relevante y útil acerca del cliente (p.ej. la información relativa al tamaño de la pantalla que se está utilizando en el cliente) o para asociar indirectamente a partir del tipo de navegador utilizado el tipo de dispositivo con que se está accediendo. Por otra parte la World Wide Web Consortium (W3C) está desarrollando un estándar para descubrir las capacidades del cliente y las preferencias del usuario [13].
  2. *Sugerencias de usuario*. En este caso, como en el modelo para las características del usuario, se pueden implementar cuestionarios o plantillas de personalización para permitir la definición de las características del dispositivo de acceso por parte del usuario.
  3. Además es necesario generar *herramientas* específicas para la determinación del estado de la red. La tecnología de *agentes inteligentes* es una de las herramientas más interesantes e importantes que se pueden utilizar.

- *Mecanismos de Adaptación* [12][13]:

  1. *Algoritmos de Compresión de datos*: presentan resúmenes del texto a mostrar, previsualizaciones de pequeñas imágenes del material disponible en la página, etc.
  2. *Algoritmos de Transformación de datos*: modifican el formato del material multimedia presentado p.ej. bajando la resolución de las imágenes o modificando la presentación de un archivo de vídeo como una sucesión de cuadros de imágenes.

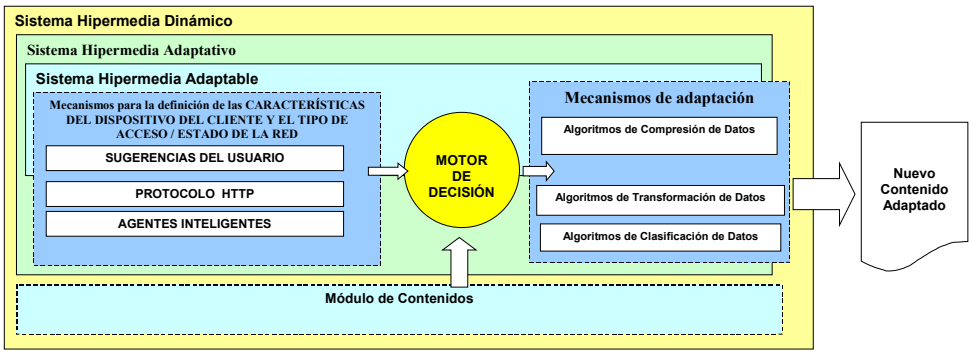


Figura 2 – Modelo para Adaptación de Contenidos según el acceso a la red

3. *Algoritmos de Clasificación de datos*: clasifican los objetos que se muestran en una página otorgándoles niveles de importancia a fin de decidir si dichos objetos serán mostrados o no.

4.3 Estado de carga del servidor

El problema de sobrecarga del servidor [11][12] también ha sido una de las consideraciones tomadas en el modelo propuesto para nuestro SHAAD, ya que del servidor y de su estado dependerá principalmente la capacidad de atender en un dado momento la cantidad de peticiones entrantes por parte de los usuarios.

En principio, estos servidores están dimensionados de tal forma que pueden servir la totalidad de las peticiones. Sin embargo, en la práctica se dan frecuentes casos en los que como usuario nos vemos ante la imposibilidad de realizar una conexión debido justamente a la sobrecarga que el servidor presenta en ese momento y a la imposibilidad que tiene para atender todas las peticiones. Como se puede producir esta sobrecarga es necesario encontrar soluciones para tales situaciones.

El procesamiento de los contenidos hipermedia sobre el servidor pueden ser *transformaciones on-line u off-line*. Un proceso *on-line o dinámico* es aquel que se realiza a partir de considerar el estado de carga del servidor, permite generar *dinámicamente* el contenido de las páginas web a partir de los elementos atómicos que la conforman. En el proceso *off-line o estático* se dispone de diferentes versiones del contenido hipermedia, en diferentes formatos y calidades, a fin de poder seleccionar alguna de esas versiones cuando las condiciones de carga lo demanden. A partir de aquí y teniendo en cuenta el estado de las variables mencionadas en los puntos 4.1 y 4.2 se acomoda el contenido hipermedia resultante a través de los otros mecanismos de adaptación.

El Módulo de Contenidos interacciona directamente con el Motor de Decisión, al cual entrega la versión de contenido que surja de considerar el estado de carga del servidor. Este motor de decisión, con el contenido hipermedia seleccionado y el estado de las otras variables definidas, implementa los mecanismos de adaptación de ese nivel que determine convenientes.

En este caso es necesario describir dos módulos (figura 3):

- *Mecanismos para la definición del estado de carga del servidor*. Las herramientas evaluadas para realizar la mencionada definición van desde las herramientas propias del servidor web utilizado hasta la utilización de tecnología de agentes inteligentes implementados en Java.
- *Módulo de Contenidos*. Se encarga de generar dinámicamente o de seleccionar a través de versiones estáticas el contenido adecuado.

Tiene los siguientes bloques:

1. *Generación dinámica a partir de elementos*

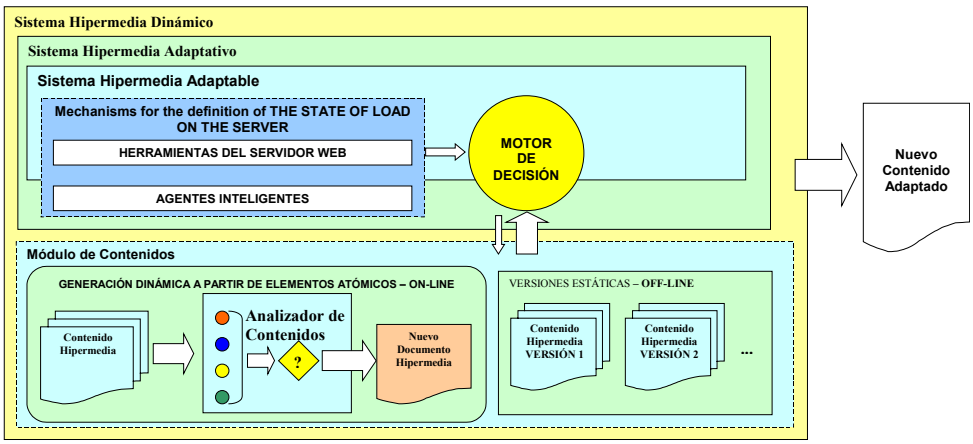


Figura 3 – Modelo para Adaptación de Contenidos según el estado de carga del servidor.

atómicos. Genera dinámicamente la nueva página a partir de los elementos que componen la página original. El Analizador de Contenidos, que se esquematiza en la figura 4, es una herramienta desarrollada en XML y Java. Este analizador tiene como objetivo convertir un documento a partir de un formato tradicional HTML a formato XML y separar de esta forma, a través de las ventajas que nos proporciona el XML, los objetos que conforman el documento original. Una vez obtenida esta estructura, a través de condicionantes y de la intervención del motor de decisión, selecciona los objetos convenientes para conformar el nuevo documento HTML.

2. *Versiones estáticas hipermedia*. En este bloque se encuentran disponibles versiones de las páginas web en diferentes calidades de formato y contenido. Estas versiones estáticas están disponibles para las situaciones en las que el Motor de Decisión determine utilizar alguna de estas versiones en el caso de que el estado de carga del servidor así lo demande.

5. Arquitectura del SHAAD

A través de los modelos propuestos en el apartado 4 hemos intentado inducir por medio de bloques homogéneos la obtención de un modelo único. De esta forma, la figura 4 representa la propuesta del SHAAD teniendo en cuenta todas las variables de adaptación consideradas.

Los diferentes módulos esquematizados han sido objeto del correspondiente análisis a lo largo del presente paper. Por otro lado, establecimos como primera instancia de nuestras investigaciones, el hecho de considerar una generación dinámica de contenidos (on-line) teniendo fundamentalmente en cuenta el estado de carga del servidor. A priori, esto parece ser la mejor forma de implementar nuestros objetivos, pero no podemos perder de vista que esta solución no puede ser exclusiva y al final los procesos de nuestro motor de decisión tendrán que incluir soluciones mixtas en las que se consideren la combinación de los estados de las variables de adaptación definidas.

6. Conclusiones

Hemos presentado en este paper el modelo SHAAD (Sistema Hipermedia Adaptable, Adaptativo y Dinámico) para la adaptación dinámica de contenidos. Este modelo intenta cubrir desde diferentes puntos de vista la amplia gama de trabajos relacionados con la adaptación de contenidos hipermedia. Así, a través del análisis de diversas técnicas para la definición de diversas variables (características del usuario, características del dispositivo de acceso del cliente, tipo de acceso, estado de la red y carga de la red) hemos tratado de englobar los distintos puntos de vista y definir a través de un modelo único nuestro punto de partida para la adecuación

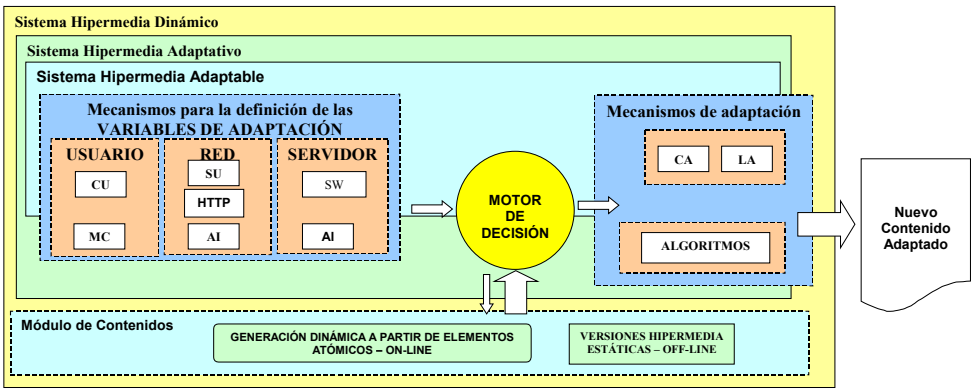


Figura 4 – Modelo final para un SHAAD

de contenidos

Algunas de las herramientas de definición consideradas se encuentran en una etapa de implementación. Este es el caso de nuestro analizador de contenidos implementado en XML y Java a través del cual se podrá realizar una selección inteligente de los objetos que conforman una página; o de las herramientas investigadas para un testeo eficiente de las condiciones de la red que se están evaluando a través del mismo protocolo http o de agentes inteligentes implementados en Java.

Nuestro modelo estrictamente modular nos permitirá trabajar sobre el amplio espectro de aspectos analizados y plantear de esta forma soluciones parciales a un problema que globalmente puede resultar excesivamente complejo debido a las muchas variables que hay que analizar.

Agradecimientos

Este trabajo ha sido parcialmente financiado por la CICYT TEL-99-0976 y por el proyecto Galecia (UE-SOCRATES - MINERVA).

Referencias

[1] Andrew Perkis, Yousri Abdeljaoued, Charilaos Christopoulos, Touradj Ebrahimi, Joe Chicharo, *"Universal Multimedia Access from Wired and Wireless systems"*, Subm.to Birkhauser Boston transactions on Circuits, Systems and Signal Processing; Special issue on Multimedia Communications, Vol. 20. , No. 3, 2001, pp. 387-402

[2] Daniel Billsus, *"Improving User Model Acquisition from Labeled Text Documents"*, Proceedings of the 2nd Workshop on Adaptive Systems and User Modelling on the WWW.

[3] Kimmo Raatikainen, Lassi Hippeläinen, Heimo Laamanen, Matti Turunen, *"Monads – Adaptation Agents for Nomadic Users"*, World Telecom '99

[4] Mauro Marinilli, Alessandro Micarelli and Filippo Sciarrone, *"A Case-Based Approach to Adaptive Information Filtering for the WWW"*, 7th International Conference on User Modeling, Banff, Canada, 1999.

[5] Narendra Shaha, Ashish Desai & Manish Parashar, *"Multimedia Content Adaptation for QoS Management over Heterogeneous Networks"*, International Conference on Internet Computing (IC 2001), Nevada, USA, June 2001.

[6] Paul De Bra, *"Design Issues in Adaptive Web-Site Development"*, Proceedings of the 2nd Workshop on Adaptive Systems and User Modelling on the WWW

[7] Pauli Misikangas, Mikko Mäkelä, Kimmo Raatikainen, *"Predicting QoS for Nomadic Applications Using Intelligent Agents"*, Impact'99 Workshop.

[8] Peter Brusilovsky, *"Adaptive Hipermedia"*, User Modelling and User-Adapted Interaction 11: 87-110, 2001, Kluwer Academic Publishers, Netherlands

[9] Reinhard Oppermann, Rossen Rashev, Kinshuk, *"Adaptability and Adaptativity in Learning Systems"*, Knowledge Transfer (Volume II) (Ed. A. Behrooz), 1997, pAce, London, UK, pp173-179 (ISBN –900427-015-X)

[10] Stefano Campadello, Heikki Helin, Oskari Koskimies, Pauli Misikangas, Mikko Mäkelä, Kimmo Raatikainen, *"Using Mobile and Intelligent Agents to Support Nomadic Users"*, 6th International Conference on Intelligence in Networks (ICIN2000), 2000, Bordeaux, France.

[11] Tarek F. Abdelzaher, Nina Bhatti, *"Web server qos management by adaptive content delivery"*, Int. Workshop on Quality of Service, June 1999.

[12] Tarek F. Abdelzaher, Nina Bhatti, *"Web Content Adaptation to Improve Server Overload Behavior"*, The 8th International World Wide Web Conference, Toronto, Ontario, Canada, 1999.

[13] Wei-Ying Ma, Ilja Bedner, Grace Chang, Allan Kuchinsky, and HongJiang Zhang, *"A framework for adaptive content delivery in heterogeneous network environments"*, MMCN2000, San José, California, 2000.

[14] Yudong Yang, Jinlin Chen, and Hongjiang Zhang, *"Adaptive Delivery of HTML Contents"*, 9th International World Wide Web Conference – The Web: The Next Generation, Amsterdam, 2000.

*G*estión *A*utomática de entrega de *P*rácticas vía web

Juan Carlos Rodríguez del Pino
Dept. de Informática y Sistemas
Universidad de Las Palmas de Gran Canaria
35017 Las Palmas de Gran Canaria
jcrodriguez@dis.ulpgc.es

Resumen

Este artículo presenta un programa llamado GAP, Gestión Automática de entrega de Prácticas vía web, el cual es capaz de almacenar las prácticas de los alumnos y que a la vez permite a los profesores el acceso a éstas para su revisión y evaluación. GAP interactúa tanto con los alumnos como con los profesores. Para tener total acceso a las características del programa es suficiente con disponer de un acceso a internet y un navegador web. También es posible usar el programa GAP para detectar plagios en un gran conjunto de ficheros fuente.

1. Introducción

Desde el punto de vista del alumno, el programa GAP es un sistema encargado de recibir y almacenar sus prácticas, además de informarle de la valoración personal realizada por parte del profesor de su trabajo. Desde el punto de vista del profesor el sistema se encarga de mostrárselas y almacenar su evaluación.

Entre otras características el programa mantiene un registro por alumno de asistencia a sesiones de prácticas, calcula la nota final para cada alumno y dispone de un sistema anticopia de prácticas.

Este artículo tiene como contexto y ámbito la experiencia del uso del programa GAP como apoyo en las prácticas de las asignaturas de Metodología de la Programación I y II, Estructuras de Datos I y II y Lenguajes de Programación en las titulaciones de Ingeniería Informática e Ingenierías Técnicas en Informática de Gestión y Sistemas en la Universidad de Las Palmas de Gran Canaria.

1.1 La era preGAP

No están muy lejos los tiempos en que la documentación de las prácticas a realizar por parte de los alumnos se distribuía en papel. Como sistema de revisión de prácticas, al final del curso, se examinaban y calificaban los trabajos, debiendo el profesor recibirlos en papel o en disquete. La llegada del World Wide Web como medio ideal y universal para la transmisión de todo tipo de información, hizo que se produjera una mejora en el suministro de información sobre las prácticas, pasando del papel a las páginas Web.

1.2 Los problemas

El primer paso en las mejoras con la llegada de internet fue poder recibir las prácticas vía correo electrónico. En el primer cuatrimestre del curso 1999/2000 se cambia el medio de recepción de prácticas: se pone a disposición de los alumnos una cuenta de correo electrónico a la que debían enviar sus trabajos. Este sistema, aunque en principio parecía mejorar la situación ya que no se requerían las prácticas en papel, tenía otros inconvenientes: Los alumnos enviaban sus trabajos desde cuentas de correo que no eran las suyas, lo que dificultaba su identificación; no enviaban en el mensaje la información de referencia sobre la práctica entregada; consultaban frecuentemente sobre si su práctica se había recibido correctamente..., y a esto se unían, además, todos los problemas propios de este medio inseguro de comunicación.

1.2 La búsqueda de la solución

Estos problemas plantearon la necesidad de buscar un sistema mejor para la recepción de prácticas. Los requisitos a cumplir eran que fuese gratuito, sencillo y flexible. Aunque se han venido usando soluciones

más o menos adaptadas al problema, en general orientadas a entornos UNIX[7] y al uso de correo electrónico[2, 8], comportaban poca flexibilidad de uso y una interfaz poco amigable. También se disponía de herramientas de docencia vía web, como Webct [10], y experiencia sobre ella [4], que aunque muy interesantes, no encajaban en la solución buscada. Por ello se planteó desarrollar un programa accesible vía Web que se encargase de automatizar la gestión del almacenamiento de prácticas.

Tras un par de meses de trabajo, se sacó a la luz la primera versión, que aunque no tenía toda la funcionalidad que se describe en este trabajo, ya realizaba suficientemente la tarea principal para la que fue concebido. El programa se ejecuta en una máquina con S.O. Linux en colaboración con el servidor web Apache. Está escrito en C++ y actualmente tiene más de 10.000 líneas de código.

2. Características

El núcleo del programa es un almacén de prácticas de alumnos. Alrededor de este elemento se han añadido características que han tomado gran relevancia como son: gestión de usuarios, calificación de las prácticas por los profesores, el sistema de comparación de prácticas para detectar su plagio o el control de asistencia a sesiones de prácticas.

2.1 Usuarios

El programa dispone de un esquema de autentificación, mediante nombre de cuenta de usuario y clave, integrado con el servidor Web. Las cuentas de usuario son de distinto tipo, lo que define el comportamiento del programa hacia ese usuario. Los tipos de usuarios son: “invitado”, “alumno”, “profesor”, “profesor corrector” y “profesor administrador”. Los usuarios “invitados” representan personas con derecho a navegar por las páginas restringidas en el servidor. El tipo de usuario “alumno” puede entregar prácticas, ver aquellas entregadas previamente y su calificación. Los “profesores” pueden acceder a la información de los alumnos almacenada en el sistema, ver sus datos personales y sus prácticas. Los “profesores correctores”, además, pueden calificar las prácticas. Los “profesores administradores” pueden realizar todo lo anterior, más las operaciones necesarias para la gestión del sistema, como pueden ser: definir prácticas, criterios de calificación global, crear cuentas, etc.

2.2 Prácticas

Un administrador puede crear, modificar o borrar definiciones de prácticas. Una definición de una práctica establece un título, una página Web opcional donde se describe, unas fechas de entrega

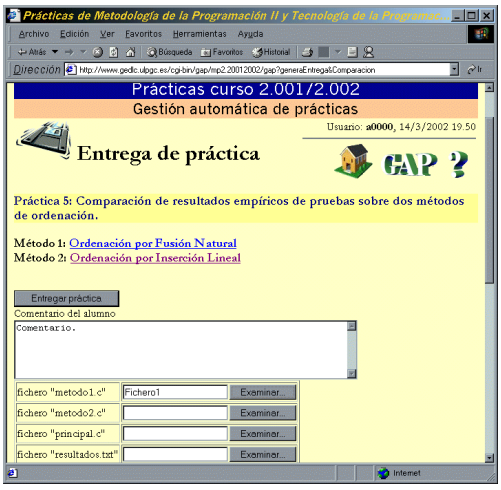


Figura 1

y los ficheros a entregar. Las fechas establecen el período de entrega de prácticas que se puede dividir en dos, opcionalmente. Las prácticas entregadas en el segundo período tendrían una penalización automática en su valoración, por retraso en la entrega. La configuración de los ficheros a entregar permite establecer qué ficheros y qué restricciones deben cumplir todos o cada uno de ellos. Las restricciones establecen el número mínimo de líneas, el número máximo y el número máximo de caracteres. Si la entrega de la práctica por parte del alumno no satisface los requisitos establecidos, el sistema rechaza la entrega. Otra característica añadida en las últimas versiones del programa es la posibilidad de que la práctica a realizar por cada alumno sea seleccionada aleatoriamente de entre un grupo prefijado. Estos tipos de prácticas tienen múltiples ventajas como se muestra en otros trabajos [1].

2.3 Revisión y visualización

La recepción, almacenamiento y visualización de prácticas es el punto fuerte del sistema. Para entregar una práctica el alumno accede a la opción del GAP correspondiente que muestra a éste un formulario a rellenar. En este formulario el alumno puede introducir los ficheros solicitados además de un comentario adicional para el profesor corrector. Ver figura 1.

Una vez entregada la práctica, los alumnos pueden comprobar la información almacenada en el sistema y verificar cómo éste se la presentará al profesor corrector, pudiendo alterarla si no le parece correcta y aun no ha sido evaluada. Si la práctica ha sido calificada por un profesor el sistema mostrará al alumno los datos de la revisión: fecha y hora de revisión, el profesor que la ha hecho, los comentarios de éste sobre la práctica y su valoración. El sistema muestra las prácticas en forma de página Web que contiene el comentario del alumno, una lista de ficheros entregados con enlaces para poder descargarlos, si se desea, y el contenido de los ficheros. Los tipos de ficheros que puede mostrar directamente son códigos fuente de programas, ficheros texto y gráficos. Los ficheros con código fuente en Java, C, C++, Ada 95, Scheme y Prolog se presentan con resalte sintáctico, mostrando en distinto color las palabras clave, los comentarios, las ristras literales, etc (ver figura 2) Si el usuario al que se muestra la práctica es un profesor corrector o administrador el sistema le ofrecerá la posibilidad de evaluarla.

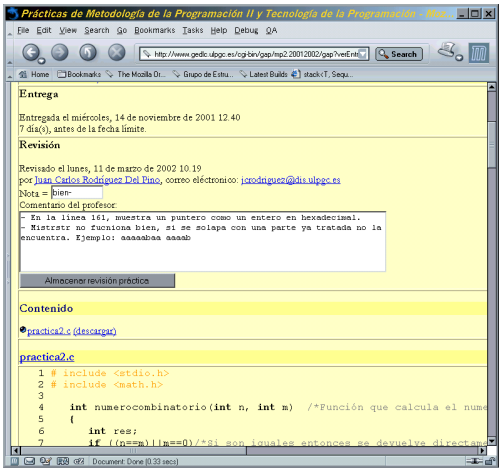


Figura 2

Los profesores pueden solicitar al sistema un listado de los alumnos junto con la información de si han entregado o no cada práctica y su valoración. Desde esta página es posible ver y revisar cualquier práctica de cualquier alumno.

2.4 Control de plagio

Al poco tiempo de poner el programa en marcha y comenzar a recibir entregas de prácticas, se descubrió que los alumnos con el nuevo sistema tenían más facilidad para copiarse. Para comprobar esta posibilidad se escribió código que verificase si varios alumnos habían entregado el mismo fichero. La sorpresa surgió al comprobar que el número de coincidencias era mayor de las esperadas, lo que motivó la adición de código que detectase el plagio de trabajos.

Los primeros sistemas de medida de similitud entre ficheros con código fuente se basaban en métricas puramente textuales que eran fácilmente vulnerables. Partiendo de un fichero original, un alumno puede, sin mucha dificultad, quitar, añadir o modificar comentarios, modificar ristras literales, cambiar nombres de variables, procedimientos o funciones, alterar el formato de línea añadiendo o eliminando espacios, o retornos de carro. También puede reordenar la ubicación de funciones o procedimientos o incluso grupos de instrucciones. Por ello se han desarrollado sistemas que han tenido en cuenta estos aspectos. Los más avanzados como YAP [11], Sim [3] y otros [6], usan distintas métricas que se basan en filtrar el código original, mediante un análisis sintáctico, para después compararlo usando generalmente una adaptación de técnicas conocidas en el ámbito textual. El filtrado del código tiene el fin de limitar al máximo el efecto en la comparación de las posibles alteraciones que se puedan haber introducido. El sistema de detección de copias empleado en GAP usa básicamente este método. Para cada lenguaje de programación se tiene un pequeño analizador sintáctico que permite eliminar las partes del código que pueden ser alterables fácilmente. Esta “firma” del programa es la que se compara con las obtenidas de los otros ficheros. GAP emplea como métrica una variante de DIT [9]. DIT es una distancia entre ristras que es invariante a la posición donde aparecen los

caracteres. Esto permite que GAP no se vea afectado por la reordenación del código, como ocurre en otros sistemas mencionados.

Actualmente el programa está preparado para comparar ficheros con código fuente en Java, C, C++, Ada 95, Scheme y Prolog. El programa ofrece a los profesores dos posibilidades de comprobar las similitudes entre los ficheros. Por un lado permite generar un informe sobre la similitud de los ficheros de una práctica comparándolos todos con todos. Por otro lado posibilita comparar todos los ficheros entregados por un alumno concreto, con los entregados por el resto de sus compañeros. Esta segunda alternativa muestra si un alumno se copia por sistema, o si tiene muchas similitudes siempre con el mismo compañero. El informe del sistema da una relación de ficheros cuya comparativa muestra un grado de similitud por encima de un umbral solicitado. El sistema no es infalible y se pueden dar casos de falsas similitudes, aunque la experiencia, incluso con gran número de prácticas, demuestra que estos falsos plagios son infrecuentes. En cualquier caso, es responsabilidad del profesor valorar el posible plagio encontrado ya que el sistema sólo da índices de similitud entre ficheros.

El empleo de este sistema ha dado resultados sorprendentes y todos los profesores que lo han empleado están muy satisfechos con él. En algunas asignaturas de primer curso y cuando los alumnos no sabían que se disponía de este elemento, los niveles de copias confirmadas alcanzaban el 15% de las prácticas entregadas. Seguramente aún escapan algunos plagios puesto que el sistema no puede detectar fraudes en el que el alumno que se copia conoce suficientemente el lenguaje y entiende la práctica para transformarla por encima del umbral establecido, o porque otro alumno le ha hecho la práctica. El primer caso no plantea problemas graves ya que el alumno demuestra tener suficientes conocimientos para superar la práctica. El segundo se puede neutralizar realizando exámenes prácticos.

2.5 Asistencia y accesos

GAP mantiene un registro personal de los accesos de cada alumno a las distintas partes del programa y de las páginas Web que se desee controlar. El profesor puede ver este registro de cada alumno que incluye desde qué máquina, a qué hora y qué día realizó el acceso.

Figura 3

Antes de usar el programa GAP, uno de los medios empleados para que los alumnos no dejasen la realización de los ejercicios para el último día era valorar la asistencia a las sesiones de prácticas. El profesor en cada sesión de prácticas tenía que pasar lista para llevar una contabilidad de asistencias. Con el GAP el control de asistencia se realiza de la siguiente manera: el profesor establece cuándo pasar lista e informa a los alumnos de que seleccionen la opción de contabilizar asistencia a la sesión. Esta información permite saber en todo momento a qué sesiones ha asistido cada alumno y qué alumnos asistieron a cada sesión. Ver figura 3.

3. Experiencia y resultados

Desde el punto de vista de los profesores los resultados han sido excelentes, todos coinciden en que una vez que han usado el GAP no se plantean volver a los antiguos métodos de entrega. El GAP ha hecho desaparecer muchos de los problemas que tenían los otros sistemas, siendo lo suficientemente flexible para no limitar el tipo de prácticas que pueden entregar los alumnos. Otro aspecto que ha cambiado es la forma en que se ven los plazos de entrega de prácticas. Muchos profesores sugieren que el plazo debe reducirse en lo posible, con el fin de que los alumnos trabajen regularmente y lleven las asignaturas al día. Las reuniones de coordinación se centran ahora en los criterios de corrección o en cómo plantear las prácticas de forma que su evaluación sea lo más simple y justa posible.

Para los alumnos el cambio ha supuesto unos plazos de entrega de prácticas más estrictos, pero

como mejora disponen de una más rápida y mejor realimentación. Reciben antes la calificación de sus prácticas, acompañadas de la descripción de sus errores.

Un cambio que ha afectado a ambos colectivos es la mejora de acceso a las prácticas. Los profesores pueden revisar prácticas desde su despacho, su casa o desde el extranjero. Los alumnos pueden realizar y entregar sus prácticas, ver sus calificaciones desde, el laboratorio, sus casas o desde cualquier lugar con acceso a internet.

4. Perspectivas de futuro

Con respecto a la gestión de prácticas se puede trabajar hacia una automatización de su evaluación [5, 8]. También existen perspectivas más factibles ya que partiendo del esquema de autentificación disponible, se pueden añadir capacidades relacionadas con el resto de los elementos de la docencia como pueden ser: la gestión de eventos y noticias, suministro de documentación y software, generación y realización de encuestas y test, etc. Aunque muchas de estas características están resueltas mediante páginas Web, su integración en el GAP mejoraría su manejo.

Referencias

[1] Bridgeman, S. SAIL:A System for Generating, Archiving, and Retrieving Specialized Assignments Using LATEX. *The Proceedings of the Thirty-first SIGCSE Technical Symposium on Computer Science Education* SIGCSE 2000. pp.300-304.

[2] Dawson-Howe, K. Automatic Submission and Administration of Programming Assigments. *SIGCSE Bulletin*, Vol. 28 No. 2 June 1996. pp. 40-42.

[3] Gitcheli, D.; Tran N. Sim: A Utility For Detecting Similarity in Computer Programs. *The Proceedings of the Thirtieth SIGCSE*. SIGCSE'99. New Orleans, Lousiana.

[4] Goldberg, Murray W. WebCT and first year: student reaction to and use of web-base resource in first year Computer Science. *SIGCSE Bulletin*. Vol. 29, No. 3, (September 1997), pp. 127-129.

[5] Jackson, D. A Semi-Automated Approach to Online Assessment. *Proceedings of the 5th Annual SIGCSE/SIGCUE Conference on Innovation and Technology in Computer Science Education.* July 2000. Helsinki, Finland

[6] Leach R. Using Metrics to Evaluate Student Programs. *SIGCSE Bulletin*. Vol. 27 No. 2, June 1995. pp. 41-43,48

[7] Macpherson, P. A.. A Technique for Student Program Submission on UNIX System. *SIGCSE Bulletin*. Vol. 29, No. 4, (December 1997), pp. 54-56.

[8] Reek K.A. A software Infraestructure to Support Introductory Computer Science Curse. *SIGCSE Bulletin*. Vol. 28 No. 1 March 1996. pp. 125-129. pp. 130-134.

[9] Santana, O.; Díaz, M. Esquemas y Estructura para la Búsqueda de las Palabras Más Similares a una dada. *Actas de la XIII Conferencia Latinoamericana de Infromática*. Bogotá, Colombia. Noviembre, 1987. Vol. II. pp. 1169-1189.

[10] WebCT. home page *http://www.webct.com*.

[11] Wise M.J. YAP3: Improved Detection of Similarities in Computer Program and Other Texts. *SIGCSE Bulletin*. Vol. 28 No. 1 March 1996.

Prácticas internacionales integradas de e-business

Francisco Araque Cuenca
Facultad de C.C. E.E. y E.E.
Universidad de Granada
España
faraque@ugr.es

Juan José Gaitán Pinto
Facultad de C.C. E.E.
Universidad Nacional del Litoral
Argentina
jjgaitan@fce.unl.edu.ar

Vlasta Hlavickova
Fac. Relaciones Internacionales
Univ. de Económicas de Praga
República Checa
hlavicko@vse.cz

Resumen
En este trabajo presentamos la experiencia docente, la tecnología utilizada y los métodos de trabajo empleados en asignaturas de informática impartidas en Facultades de C.C. E.E. y E.E.

1. Introducción

A raíz del intercambio de experiencias realizado en ocasión de las estancias de los autores en ambas universidades, en cumplimiento de sendas Becas de la AECI, se fue gestando un proyecto de colaboración en el área docente de tecnologías de la información de asignaturas impartidas en titulaciones de Facultades de C.C. E.E. y E.E. tales como: Licenciado en Administración y Dirección de Empresas, Diplomado en Empresariales, Diplomado en Administración y Gestión Pública, etc. De este modo se delinearon algunas propuestas conducentes al aprovechamiento de Internet y sus servicios básicos como vehículo para la realización de experiencias pedagógicas.

Las asignaturas relacionadas con Introducción a la Informática e impartidas en Facultades de C.C. E.E. y E.E suelen tener un contenido similar en la mayoría de los casos. Básicamente se puede dividir en:

La parte teórica: Introducción (Conceptos, Sistemas de Información), Fundamentos tecnológicos (software, hardware, bases de datos, redes), Tecnologías avanzadas (DSS, EIS, EIP, DW, E-Business, etc).

Y la parte práctica: Conceptos básicos (Windows, Correo electrónico), Paquetes Integrados (Procesador de textos, Hoja de cálculo, Bases de Datos) y Comunicaciones (Navegación, Diseño de Páginas Web).

Además se imparte en el primer o segundo curso y, en la mayoría de los casos son obligatorias.

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.

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) y por otra parte la Universidad Nacional del Litoral (UNL, Argentina). Está previsto que a medio plazo, y con la idea de fomentar el uso del inglés, se adhiera a la realización de las prácticas la Universidad de Económicas de Praga (República Checa) y la Facultad de C.C. E.E. de la Universidad de Trieste (Italia).

En este trabajo pretendemos presentar la experiencia adquirida en el desarrollo de las prácticas conjuntas entre la UGR y la UNL.

A continuación se expone el programa de las asignaturas involucradas en la experiencia docente. En la sección tercera se comentan las distintas fases que se han realizado o que están en fase de desarrollo, en la cuarta se comentan algunos de los beneficios encontrados y acabaremos con unas conclusiones.

2. Las asignaturas

La asignatura de *Introducción a la Informática* se imparte en el primer curso de la Diplomatura en Ciencias Empresariales de la UGR [12]. Es obligatoria con 4 créditos teóricos y 2 prácticos. El programa es el siguiente:
Teoría:
1. Introducción: Conceptos básicos.

2. El PC y la automatización de oficinas.
3. Descripción funcional del hardware.
4. Tipos y estructuras de datos.
5. Software de una computadora.
6. Redes y comunicaciones.
7. La información como recurso.
8. Los SI en la empresa: DSS, EIS, SIE.
9. Uso de los MIS para adquirir ventaja competitiva.

Prácticas:

1. Sistemas Operativos.
2. Procesador de texto.
3. Hoja de Cálculo.
4. Bases de Datos.
5. Internet (e-mail y Navegación)

Dentro de la teoría y las prácticas, las partes más importantes, en cuanto a conceptos y tiempo dedicado, son las de bases de datos y redes.

La asignatura *Informática* se imparte en el segundo curso de la Licenciatura en Administración de Empresas de la UNL [10]. Es obligatoria y el sistema de créditos no es aplicable en esa Universidad. El programa es el siguiente:

Teoría:

1. Sistemas de Información.
2. Interpretación de ofertas de equipamiento.
3. Adquisición del software.
4. Planilla de Cálculo.
5. Bases de Datos.
6. Redes locales y globales en la empresa.
7. Internet aplicado a la empresa.
8. Seguridad informática.
9. Nuevas tecnologías aplicadas a los negocios.

Prácticas:

1. Hoja de Cálculo.
2. Bases de Datos.
3. Internet (e-mail y Navegación)

También en esta asignatura se pone un especial énfasis en las bases de datos y redes.

La primera experiencia la realizamos hace dos cursos con algunos grupos de alumnos de la asignatura de la UGR y los participantes de un seminario de Ecommerce que se realizaba el la UNL. De esta forma coincidíamos en fechas y se realizaban cuando las prácticas de la asignatura de la UGR estaban un poco avanzadas. Las dos asignaturas cuyo programa se detalló anteriormente no coinciden en fechas, sin embargo, en función de los resultados obtenidos, las autoridades han considerado la posibilidad de analizar la modificación del dictado de la asignatura de la UNL, para lo cual en el año 2002 se impartirán algunas grupos de prueba a partir del mes de marzo.

3. Prácticas integradas

A continuación exponemos las diferentes fases del proyecto. Un trabajo que lleva ya varios años de intercambio permanente de trabajo, experiencias, etc a través de correo electrónico. Las dos primeras etapas ya están concluidas y la tercera se encuentra en fase de desarrollo.

3.1 Primera aproximación

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 [5], [7].

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 (como por ejemplo lámparas, faros, ventiladores, decoración, 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. Otros de los supuestos en que se basa el caso, es que ninguna de las empresas mantiene stock de productos elaborados, es decir que ante el pedido de compra del consumidor final, deberá proceder a elaborar el producto final. Puesto que en el mismo intervienen materiales de elaboración propia y otros no elaborados por la empresa, la misma deberá adquirirlos a alguna o varias de las otras empresas. Cada empresa dispone de 4 áreas funcionales: Producción (P), Compras (C), Finanzas (F) y Ventas (V).

Finalmente, un ordenador es utilizado por el Docente, quien actúa como cliente externo, demandando mercaderías a cada una de las empresas, desencadenando así el proceso de operaciones comerciales entre ellas. De esta manera, utilizando los recursos que ofrece la conectividad en red, se realizan las operaciones utilizando el correo electrónico y el acceso remoto a la información. En

este caso el procedimiento sería el que a continuación se comenta.

Ante una nueva compra del cliente minorista, el departamento de ventas envía una orden de fabricación al departamento de producción. Este último envía un pedido de compra de productos al departamento de compras. El departamento de compras, antes hacer efectiva la compra, envía una petición de presupuesto a las distintas empresas que elaboran lo productos. El proveedor remitirá el presupuesto al departamento de compras solicitante.

Compras recibe los presupuestos remitidos por las distintas empresas a las cuales lo solicitó y procede a realizar un análisis comparativo de los presupuestos. En base a los mejores precios, confecciona las órdenes de compra y las envía a los respectivos proveedores. Se asume que cada proveedor elabora productos de la misma calidad y que la forma de pago y financiación es similar en cada uno de ellos, y que por lo tanto el único factor decisivo de la compra es el precio. El proveedor recibe la orden de compra de su cliente al por mayor y procede a enviar las mercadería. Comunica vía correo electrónico a la empresa compradora que las mercaderías van en camino, a los efectos de que el departamento de finanzas ponga a disposición los fondos necesarios para efectuar el pago.

Cuando compras recibe de los proveedores el mensaje mediante el cual éstos le indican que las mercaderías están en camino, Compras envía un mensaje a Producción indicando que el trámite de compra está concluido, y por otro lado informa a Finanzas que ha procedido a efectuar una compra, enviando a este departamento una copia a efectos de que Finanzas planifique sus pagos. Cuando los productos están elaborados, Producción comunica la disponibilidad a Ventas, quien procede a enviarla al cliente, informando a Finanzas de la realización de la venta. Finalmente se realiza la facturación de la mercadería y oportunamente se acredita el pago de la misma por parte del cliente.

Los recursos técnicos utilizados fueron los disponibles en cualquier aula de informática: clientes de correo electrónico y software de bases de datos (Microsoft Access). Además, se disponía de una página web básica para incorporar información relativa a la organización de las prácticas, foro de consulta, etc.

En esta primera fase, ya concluida, los alumnos diseñaban la base de datos (con ayuda de una guía detallada) y la utilizaban en la gestión administrativa de su "empresa virtual". Cada vez que fuera necesario, los alumnos actualizaban la base de datos utilizando directamente Microsoft Access. Algunos grupos más avanzados y de manera voluntaria, diseñaban y utilizaban un formulario para la modificación de la base de datos.

El valor de esta experiencia radica en la integración de los conocimientos en una práctica concreta muy cercana al ejercicio profesional. La propuesta fue bien acogida por los alumnos, quienes demostraban su entusiasmo y asumían con responsabilidad los roles que se le asignaban dentro de la organización de la cual formaban parte.

Anticipando lo que sería la integración entre ambas universidades, en esta primera fase se decidió que las empresas, las bases de datos y las cuentas de correo serían las mismas en todas los grupos de práctica, de manera que los alumnos iniciaban cada clase con operaciones pendientes de finalización y solicitudes aún no atendidas. De este modo la integración se realizaba entre todos los alumnos del curso.

Quizá, el saldo de esta experiencia fue la certeza de que era necesario profundizar y mejorar todo lo logrado, y la sospecha de que esta decisión podría llevarnos a resultados enriquecedores y, a la realización de más trabajo "extra".

3.2 Segunda aproximación

Después de la experiencia adquirida en la primera fase, y con la idea de integrar también el *Comercio Electrónico*[1], nos propusimos construir una aproximación a esta nueva realidad que implicaba el ejercicio comercial de muchas empresas. 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. 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 consciente 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, para aquellas personas autorizadas previamente por el docente. De este modo, se anima a empresas a participar de una manera fácil y gratuita de las ventajas del comercio on-line. Una arquitectura descriptiva la podemos ver en la figura 1.

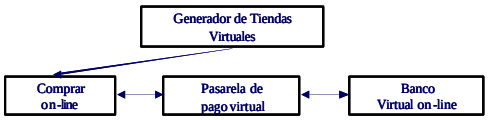


Figura 1. Arquitectura de la 2ª aproximación.

El funcionamiento sería el siguiente: un empresario (grupo de alumnos) decide crear una tienda virtual. Ese empresario posee unos datos bancarios y de sociedad. Se conecta vía Web con el generador de tiendas virtuales. Este generador recoger los datos del nuevo usuario así como los datos de los productos de dicha tienda. En la figura 2 podemos ver la captura de pantalla de la parte de introducción de productos del catálogo de una tienda.

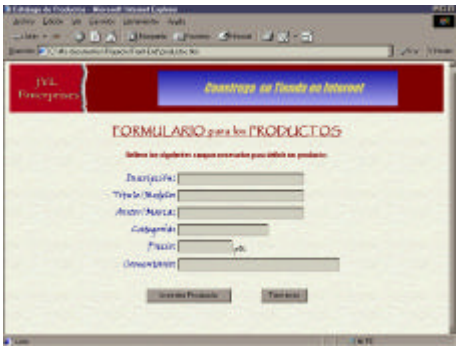


Figura 2. Introducción de productos

Posteriormente, la tienda se conecta con la pasarela de pago indicándole los datos financieros y bancarios del usuario. Si los datos son correctos la pasarela otorga un certificado cifrado único con el que firma sus órdenes de compra.

Una vez recibido el certificado por el generador de tiendas virtuales éste lo guardará de forma segura quedando la información delicada (como el nº de cuenta) en la pasarela de pago. Con este último paso quedaría creada la nueva tienda virtual en la red y lista para funcionar.

La experiencia continúa cuando un cliente decide comprar en esta tienda. Se conecta a la tienda, realiza su compra y sólo si lo desea aportaría sus datos personales. Una vez realizada la compra la tienda envía el resultado de la compra a la pasarela de pago. Ésta muestra de nuevo el detalle de compra para que pueda ser verificada por el cliente que ahora sí introduce sus datos de tarjeta de crédito identificándose mediante contraseña. Una vez realizado este paso la pasarela realizará una transferencia bancaria entre las cuentas del cliente y de la empresa de forma confidencial para las entidades bancarias involucradas.

Finalmente, la pasarela, como notario de la operación, enviará (vía email) un justificante de compra firmado por ella a cada una de las entidades involucradas en la operación de compra: la empresa que vende y el cliente. Con esto se evitará fraudes posteriores de ambos ya que legalmente la palabra de la pasarela es la que vale.

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. La idea fundamental es la construcción de una entidad bancaria ficticia que simule las operaciones más comunes realizadas en una entidad bancaria real.

Esta aplicación, a pesar de ser una simulación, controla la seguridad protegiendo la confidencialidad y contenidos de la información de los clientes pertenecientes a la entidad. Esto se realiza utilizando protocolos de seguridad estándar que permiten establecer una comunicación entre las dos partes involucradas en la operación: el banco y el cliente. La aplicación consta de una serie de opciones de configuración que permiten al cliente de la entidad financiera personalizar fácilmente el interfaz, haciendo más agradable su interacción con la aplicación. Se pueden realizar también toda clase de operaciones bancarias usuales tales como, ingresos, transferencias, petición de préstamos (bajo autorización del docente), etc. En la figura 3 podemos apreciar una parte del interfaz vía web

del banco virtual una vez que se ha identificado el usuario.

Figura 3. banco Virtual

La pasarela de pago actúa como medio neutral en las transacciones entre las tiendas dadas de alta en la pasarela y las entidades bancarias asociadas a esta alternativa de compra. La ventaja principal de esta pasarela de pago es el cumplimento de los cuatro criterios de seguridad: confidencialidad, integridad, autentificación y no repudio.

Los recursos técnicos utilizados han sido los siguientes[4]:

- Banco virtual on-line: Windows NT Server, IIS y Active Server Pages (ASP).
- Generador de tiendas: Linux, Apache y MiniVend.
- Pasarela de pago virtual: Windows NT Server, IIS, Active Server Pages(ASP) y SSL.

El desarrollo se hizo a medida del problema a resolver pero, para la gestión de las tiendas, se utilizó el software *MiniVend*[11]. Este producto es un completo sistema que implementa una tienda virtual de venta por catálogo, también conocido como "carrito" o "cesta de la compra", de forma rápida. Es un sistema de acceso y recuperación de datos enfocado al Comercio Electrónico. Muy potente, con una base de datos funcional y de fácil manejo para el cliente. Altamente complejo, puede manejar fácilmente catálogos de más de un millón de artículos con excelentes resultados. El problema de este software es que carece de interfaz y todo hay que realizarlo a través de ficheros texto, cambiando configuraciones complejas para un usuario no experto. Por este motivo se desarrolló el generador de tiendas virtuales antes comentado.

Cada grupo de prácticas comenzaba las prácticas con un saldo mínimos en el banco virtual (500.000 pts/ 3005 €). Los pasos a seguir, una vez organizados los grupos de prácticas (3 o 4 personas) eran:

- Pensar algún tipo de empresa.
- Delinear un plan estratégico básico.
- Dar de alta la tienda utilizando el generador de tiendas virtuales.
- Crear una cuenta en el banco con los datos asignados al grupo.
- Realizar una mailing informando de la creación de la empresa al resto de alumnos (de ambas Universidades).
- Realizar transacciones comerciales (compras y ventas) con otros grupos de alumnos. Se supone que bs alumnos compran según las necesidades de su empresa (porque necesitan algún producto para su propio negocio) o como consumidores atraídos por los productos que otra empresa vende.
- Pagar las compras utilizando sus cuentas corrientes y tarjetas de crédito ficticias, creadas en el mismo banco que se ofrece en el entorno virtual de esta práctica.
- Realizar informes comerciales de su ingresos y ventas.
- Acabar el periodo de prácticas con saldo positivo en su cuenta de resultados (cuenta bancaria).

Como puede observarse, se trata de una notable evolución de aquella primera fase, pero además se puede apreciar aquí el enorme potencial que se nos presentaba, ya que a los objetivos propios de cada asignatura, podríamos incorporar algunos nuevos, planteando un escenario ambicioso pero al mismo tiempo estimulante:

- Diseñar y poner en funcionamiento un Sistema de Simulación de Comercio Electrónico para Empresas Virtuales con el objeto de ser utilizado con fines académicos, por alumnos de instituciones educativas de distintas partes del mundo.
- Facilitar la enseñanza del funcionamiento y uso de los métodos de comercio electrónico, tanto desde la posición del consumidor final, como desde la propia actividad empresarial.

- Estimular la creación de empresas virtuales que ofrezcan al mercado internacional los productos manufacturados en la industria local de cada institución educativa, favoreciendo así el conocimiento de la actividad económica regional.

Ante ésta nueva realidad, entendiendo que las dificultades tecnológicas y operativas no pueden hacernos perder la voluntad de alcanzar –aunque sea parcialmente- los objetivos mencionados, decidimos avanzar progresivamente en el diseño de una nueva etapa, esta vez más vinculada aún a las nuevas tecnologías que el *e-business* [1], [6] propone para el funcionamiento de las empresas de la nueva economía, y para el ejercicio profesional de los futuros graduados.

3.3 Tercera aproximación

Vistas las carencias de las dos experiencias anteriores, lo siguiente que nos planteamos es intentar subsanarlas en la medida de los 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. Además habrá que actualizar la plataforma de prácticas con tecnologías actuales así como aplicar los nuevos conceptos surgidos.

Para hacer realidad esta nueva fase, será interesante incorporar al nuevo escenario algunas de las últimas tecnologías de internet y los negocios, aquellas que marcan un hito contundente entre la nueva y la vieja economía y que habrán de impactar severamente en el desempeño profesional de los futuros profesionales. Es casi una obligación que esta tercera etapa se nutra de todo lo nuevo, contribuyendo así a vencer la inercia natural con la que estas innovaciones se integran a la enseñanza.

Lo que nos planteamos como trabajo en desarrollo y futuro es la utilización de un *Portal* en el que se integren varias cuestiones:

- Punto de encuentro entre docentes para intercambio de experiencias, aportación de conocimientos, etc. Para este punto nos proponemos utilizar tecnologías basadas en el manejo del conocimiento (Knowledge Management, KM) [3] y no quedarnos sólo en un foro de intercambio de mensajes.
- Punto de encuentro para alumnos. Foros de consulta, intercambio de apuntes, dudas, etc.
- Incorporar al Portal, sistema de Juegos de Empresa, donde las distintas empresas virtuales compitan por el logro de ciertos objetivos comerciales y empresariales, en un entorno globalizado [9].
- Punto de encuentro para la realización de las prácticas. Este punto, por ser el de más interés será comentado con más detalle a continuación.

Respecto a la realización de las prácticas integradas se ha pensado la implantación de un e-hub[6], [8]. 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.

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 tran-

sacción y en muy pocas ocasiones la relación entre el comprador y el vendedor es duradera. Un clasificación la podemos ver en la tabla 1:

Cómo ↓		
Sistemático	*MRO hubs* Ariba MRO BizBuyer	*Catalog Hubs* Chemdex SciQuest ElectricalWeb
En el acto	*Yield* Employase NTE CapacityWeb	*Exchanges* E-Steel PaperExchange ChemConnect
Qué →	Funcionamiento	Fabricación

Tabla 1. Clasificación de los E-Hub

Nosotros nos hemos centrado en los e-hub llamados *Exchanges*. En este caso, 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 [2], 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 la segunda fase, está previsto que los alumnos diseñen una base de datos para la parte de productos ofertados, etc y que ésta se integre (en base a consultas SQL [5] 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 bases de datos haría el papel del sistema existente en la empresa y que tienes que ser integrada en el e-hub.

En la figura 4 podemos ver la arquitectura básica de un e-hub integrada con el banco virtual desarrollado en la segunda fase.

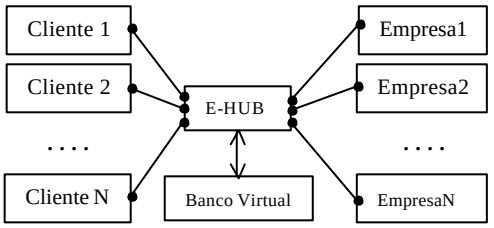


Figura 4. Arquitectura general de un e-hub

En este caso los pasos a seguir por el grupo de alumnos sería (sujeta a modificaciones ya que está en proceso de desarrollo):

- Comienzo con un saldo mínimos en el banco.
- Pensar algún tipo de empresa. Esto estaría condicionado por el tipo de e-hub que se ofertara y por las empresas preexistentes en dicho e-hub.
- Diseñar y realizar en access (con ayuda de un guión) una base de datos que se ajuste a la empresa en cuestión.
- Delinear un plan estratégico básico.
- Dar de alta la empresa en el e-hub.
- Realizar una mailing informando de la creación de la empresa al resto de alumnos (de ambas Universidades).
- Realizar transacciones comerciales (compras y ventas) con otros grupos de alumnos.
- Seguimiento de las transacciones y estadísticas generadas por el e-hub para extraer conclusiones comerciales.
- Realizar informes comerciales de ingresos y gastos consultando sus cuentas bancarias.

- Acabar el periodo de prácticas con saldo positivo en su cuenta bancaria).

Esta tercera fase, así como la segunda lo fue, está siendo desarrollada por alumnos de proyectos de 5 de la Universidad de Granada.

4. Beneficios

A continuación se comentan algunos de los beneficios encontrados en base a la experiencia desarrollada:

- La comunicación internacional con estudiantes de otras universidades, favorecerá la adopción de una actitud positiva hacia la tecnología y las asignaturas, a la vez que dará origen a redes de comunicación informal de información académica.
- La comprensión del rol protagónico de las nuevas tecnologías en el ejercicio profesional, estimulará su incorporación activa a las distintas asignaturas, potenciando el proceso de enseñanza-aprendizaje.
- Ofrecer al ambiente académico un recurso potenciador de la capacidad creativa de los docentes, posibilitando la participación activa en un sistema abierto, dando así oportunidad de que el mismo sea utilizado para fines académicos diversos, con objetivos y perspectivas de logros distintos y adaptados a cada realidad particular.
- Favorecer la transferencia de conocimientos significativos acerca de la importancia que el impacto de las nuevas tecnologías de la información tendrán sobre la actividad de las empresas en escenarios futuros.
- Comprender las oportunidades comerciales que se generan a partir de la globalización de los mercados, preparando a los futuros graduados para trabajar en empresas y organizaciones que atiendan las demandas locales, nacionales e internacionales.
- Integrar los distintos contenidos programáticos de las asignaturas informáticas de las distintas carreras de Ciencias Económicas y Empresariales en una práctica concreta, favoreciendo los procesos cognitivos y motivacionales de los alumnos.

5. Conclusiones

En el presente trabajo hemos intentado dar nuestra visión y nuestra experiencia en la realización de prácticas relacionadas con asignaturas impartidas en Facultades de C.C. E.E. y E.E. El objetivo no era citar una serie de recursos tecnológicos para la realización de las prácticas sino más bien contar como nosotros planteamos la incorporación de las nuevas tecnologías a las asignaturas ya comentadas. Ni que decir tiene que todo lo expuesto antes sirve también, y de una manera bastante práctica, para explicar conceptos teóricos.

Queda mucho trabajo por realizar y, por supuesto por mejorar y es nuestra intención desarrollarlo a lo largo de los próximos años.

Referencias

[1] Ana Rosa del Águila, Antonio Padilla. *E-business y Comercio Electrónico: un enfoque estratégico*. Rama, 2001.
[2] F. 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.
[3] Joseph Firestone. *Enterprise Knowledge Portals: What They Are and What They Do*. Journal Of Knowledge And Innovation, October, 2000
[4] Manuel García, Javier Martínez. *Simulación del Comercio Electrónico a través de Empresas y Bancos Virtuales*. Proyecto Final de Carrera. 1998, Universidad de Granada.
[5] María Teresa Martín, Luis Alfonso Ureña. *Ejercicios de bases de datos relacionales*. Universidad de Jaén, 2001.
[6] Mark M. Davydov. *Corporate Portals and E-business integration*. McGraw-Hill, 2001.
[7] Silberschatz Korth. *Fundamentos de Bases de Datos*. Mc-GrawHill, 1993.
[8] Steven Kaplan, Mohanbir Sawhney. *E-Hubs: The new B2B marketplaces*. Harvard Business Review, may-June 2000.
[9] Thompson Stappenbeck. *The Business Strategy Game*. McGraw-Hill, 1998.
[10] www.fce.unl.edu.ar
[11] www.minivend.com
[12] www.ugr.es/~fcee

Nuevas Tecnologías de la Programación en Internet

Antonio Fernández, José Antonio Piedra, Miguel Ángel Plaza

Departamento de Lenguajes y Computación
Universidad de Almería
{afm, jpiedra, maplaza} @ ual.es

Resumen

La asignatura Nuevas Tecnologías de la Programación (NTP) establece un planteamiento novedoso a la hora de presentar contenidos que apenas aparecen en los planes de estudios y cuando lo hacen se encuentran dispersos por varias asignaturas. Esta asignatura pretende ofrecer al alumno una visión completa de las tecnologías utilizadas en el desarrollo de aplicaciones web. Partiendo del diseño de páginas estáticas (HTML) y de las tecnologías orientadas a la presentación (CSS, JavaScript), repasa tecnologías de cliente (applets Java), para mostrar luego tecnologías de programación para servidores (Servlets, JSP y EJB), completando el recorrido con una visión general del acceso a base de datos a través de Internet (con JDBC). Estos contenidos se imparten a través de la utilización de metodología de trabajo en grupo y el apoyo de herramientas de teleformación o docencia virtual que se describen con detalle en otros artículos presentados en JENUI [3], [4] y [5].

1. Introducción.

La situación actual de la programación vive unos momentos excitantes. La popularidad de Internet ha obligado a que los programadores dominen las tecnologías más punteras y que tengan capacidad para crear aplicaciones ejecutables sobre Internet. Un desarrollador de aplicaciones web necesita conocer una lista enorme de tecnologías cuyo flujo de información en Internet nos bombardea con una montaña de datos. Por este motivo, no es poco el esfuerzo que debe hacerse por parte de los programadores para comprender los conceptos relacionados con Internet y el desarrollo de aplicaciones web.

Hace sólo unos pocos años, los programadores se podían especializar en áreas más o menos estrechas sin preocuparse de otras disciplinas. En los entornos de programación actuales, los diseñadores no sólo tienen que estar al tanto de las nuevas tecnologías, sino que tienen que tener, al menos, un conocimiento intermedio de las mismas.

Desde el punto de vista profesional, los programadores necesitan tener una amplia visión de lo que ofrece y hasta donde puede llegar las tecnologías y herramientas ofrecidas en el mercado en cuanto a requerimientos técnicos profesionales. Un desarrollador de aplicaciones web, necesita conocer una lista enorme de tecnologías: lenguajes de programación de páginas web, tecnologías de programación en el lado cliente, en el lado servidor, tecnologías de acceso a base de datos a través de Internet y otras tecnologías más complejas: Servlets, componentes EJB, COM, DCOM, CORBA, ODBC, JDBC, etc...

No cabe duda que la formación de un desarrollador en todas estás tecnologías es muy difícil. La asignatura NTP plantea centrarse en las tecnologías propias de alguno de los grandes fabricantes evitando así una formación extensiva a todas ellas.

El Computing Curricula 2001 [1] recoge algunas recomendaciones sobre cuales deben ser las características de un graduado en el área de Ciencias de la Computación.

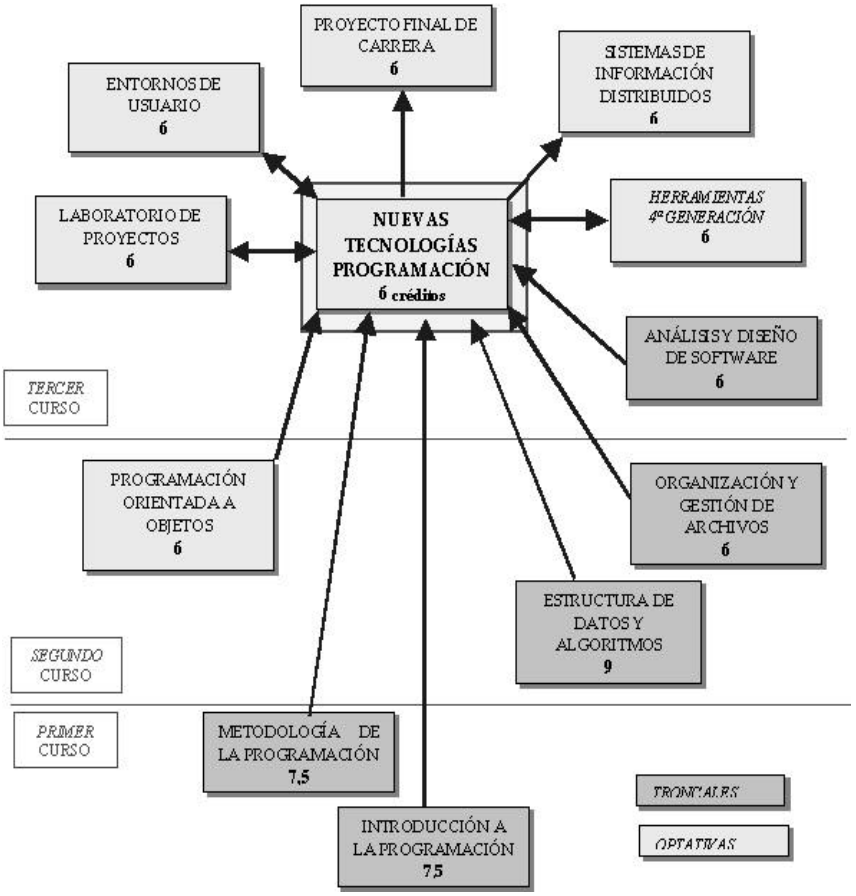


Figura 1. Relación de la asignatura NTP con otras del plan de estudios de ITIG.

Entre estas características destacan: El *modelado* (utilización de los conocimientos y la comprensión para el diseño y modelado de sistemas basados en computadores) y los métodos y herramientas (aplicar de manera apropiada teorías, prácticas, y herramientas para la especificación, diseño, implementación y evaluación de sistemas de información).

En este sentido, la asignatura NTP intenta ser un resumen de dichas características para los titulados Ingenieros Técnicos en Informática de Gestión (en adelante, ITIG) fomentando en el alumnado la capacidad de trabajo en grupo y el uso de las tecnologías y herramientas de programación actuales.

2. La asignatura NTP en el plan de estudios de ITIG.

La asignatura Nuevas Tecnologías de la Programación, es una optativa del tercer curso de la titulación de Ingeniero Técnico en Informática de Gestión (ITIG) que se imparte en la Escuela Politécnica Superior de la Universidad de Almería durante el segundo cuatrimestre, tiene asignados un total de 3 créditos teóricos + 3 créditos prácticos.

Según el plan de estudios actual (BOE de 11 de Julio, nº 165/2000), los descriptores de la asignatura NTP son:

Nuevas Tecnologías de la Programación. Programación en Internet. Diseño de componentes. Aplicaciones basadas en componentes remotos.

NTP es una asignatura de carácter optativo y se sitúa en el último cuatrimestre de la titulación de Ingeniero Técnico. Por tanto las interacciones que tiene con otras asignaturas son bastante complejas. Por un lado recibe influencias de otras asignaturas impartidas con anterioridad (*Programación Orientada a Objetos*). Por otro, va a influir decisivamente en otras asignaturas (*Sistemas de Información distribuida*). En algunas ocasiones hay una relación bilateral (por ejemplo con *Entornos de Usuario*).

En la figura 1, aparece representadas de manera gráfica todas estas interacciones. Cabe destacar que los contenidos de esta asignatura son esenciales para otras asignaturas (como Sistemas de Información Distribuida) que se imparte simultáneamente pero que supone una continuación natural de los conceptos introducidos en NTP. Por último también debemos señalar que cada día son más los alumnos que recurren a las tecnologías que han conocido a lo largo de NTP para desarrollar su Proyecto Fin de Carrera.

3. Particularidades de la asignatura.

Destacamos en este apartado las particularidades que presenta la asignatura en relación al plan de estudios de ITIG:

- Se requieren conocimientos básicos de POO (Programación orientada a objetos).

Son varias las estrategias a la hora de elaborar la docencia de la programación en los cursos introductorios de las titulaciones de Ingeniero Técnico en Informática (*paradigma tradicional imperativo, paradigma orientado a objetos, programación funcional*). El profesorado del Área de Ciencias de la Computación de la Universidad de Almería aboga por la utilización del paradigma orientado a objetos para la formación del alumnado desde las asignaturas básicas de programación.

- Ha resultado difícil identificar los contenidos de NTP con asignaturas presentes en recomendaciones internacionales.

NTP es una asignatura basada en contenidos muy novedosos y por tanto abierta a cierta experimentabilidad. Este puede ser uno de los motivos por lo cual no aparece expresada explícitamente como una asignatura en el CC2001 [1] sino que sus contenidos se encuentran repartidos entre otras asignaturas.

4. Objetivos.

NTP a través de su contenido y metodología docente presenta tres objetivos generales:

- Formar e informar al alumno de la situación actual de la programación a través de las nuevas tecnologías que han aparecido en esta área.
- Fomentar entre el alumnado la metodología de trabajo en grupo mediante el uso de la docencia virtual como herramienta de apoyo.
- Incrementar la capacidad de autoaprendizaje del alumno en la selección de tecnologías de la programación en Internet.

El primer punto se desglosa en una serie de objetivos particulares relacionados con cada uno de los temas de teoría y prácticas de la asignatura:

- Conocer los principales conceptos implicados en la programación de Internet y cómo han evolucionado los servicios y tecnologías hasta llegar a la situación actual.
- Saber elegir para cada problema un grupo de tecnologías que de manera conjunta aporten la solución más eficiente.
- Proponer el diseño de un sitio web sencillo pero desde todas sus vertientes: diseño del sistema de información, diseño de contenidos, diseño gráfico, elección del sistema operativo, servidor web, gestor de base de datos, tecnologías de programación en cliente y en servidor.

- Comprender que algunas de las tecnologías que se conocen en esta asignatura están al borde de la obsolescencia, mientras que otras se encuentran muy afianzadas y tienen un futuro prometedor y por último hay otras que acaban de nacer y su futuro es bastante incierto.
- Enmarcar todos los términos y tecnologías conocidas en un diccionario de vocablos que todos los alumnos deben conocer, comprender y utilizar con suma propiedad.

5. Programa de teoría.

El programa de teoría de la asignatura responde a los objetivos planteados anteriormente. Con NTP, se ofrece una visión global del desarrollo de una aplicación web. En la figura 2 aparece un esquema de un sistema que incluye la mayor parte de tecnologías de programación que se van a incluir como contenidos de la asignatura.

Tema 1	Visión general de la programación en Internet	2 horas
Tema 2	Servidores de aplicaciones web.	1 horas
Tema 3	Tecnologías de diseño de páginas web.	2 horas
Tema 4	Tecnologías de programación web en el lado cliente.	2 horas
Tema 5	Tecnologías de programación en el lado servidor	9 horas
Tema 6	Tecnologías de acceso a la información a través de Internet: ASP vs JSP	8 horas
Tema 7	Tecnologías de desarrollo de componentes: JavaBeans, ActiveX, COM y DCOM	4 horas
Tema 8	Otras tecnologías de programación web: XML	2 horas

Tabla 1. Descripción de la asignatura NTP.

A lo largo de los temas de teoría se describen todos los aspectos y tecnologías utilizadas para una aplicación web en sus diferentes niveles.

Se comienza con una visión general del mundo y de la programación en Internet. En el segundo capítulo se presenta los contenidos y características básicas de los servidores web.

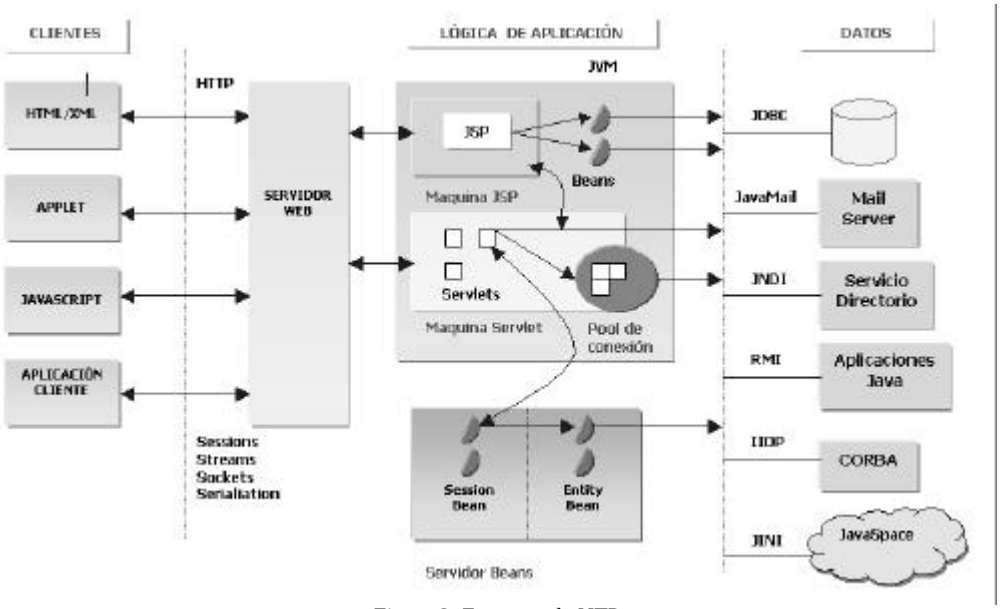


Figura 2. Esquema de NTP.

Se termina el temario de la asignatura con la descripción de componentes y aplicaciones basadas en componentes remotos.

Debido al carácter innovador de la asignatura, hay muy poca bibliografía en castellano. Además la bibliografía anglosajona aborda los contenidos más desde el aspecto profesional que docente (lo cual no es un problema en absoluto). Por este motivo, el alumno se ve en la necesidad de buscar y consultar información presente en Internet que describa dichas tecnologías e incluso acudir a los tutoriales de algunas de estas ofrecidos por fabricantes o sitios web especializados.

El contenido de la asignatura se ve apoyado con la realización de dos seminarios introductorios a tecnologías de programación actuales durante los cuales se pretende dar a conocer a los alumnos algunas tecnologías que sin estar muy relacionadas con las planteadas en la asignatura si que son unas magnificas complementarias de las mismas. Con ello se pretende promover el trabajo de investigación, autoformación y exposición de los conocimientos adquiridos por los alumnos que preparan el seminario. Los seminarios impartidos son:

- Flash, tecnología para la animación de páginas web.
- WAP/WML, tecnologías para programar aplicaciones accesibles a través de teléfonos móviles.

También sería aconsejable invitar a profesionales de empresas lider en el sector para que describan las ventajas e inconvenientes a la hora de llevar a la práctica este tipo de tecnologías.

6. Las prácticas: un aspecto fundamental.

Las prácticas de la asignatura NTP son de carácter fundamental para el desarrollo de la asignatura ya que a través de estas el alumno trabajará con las tecnologías descritas durante las clases de teoría. Podemos resumir los objetivos de las prácticas de NTP:

- Conseguir que el alumno conozca a nivel instrumental las tecnologías de las que se le ha informado en las clases de teoría. No se pretende que adquiera una profunda formación en cada tecnología sino que conozca como se instala y se utiliza a nivel instrumental para que a partir de ese momento él pueda profundizar por propia iniciativa en cada una de las tecnologías.
- El alumno va a montar su propio sitio web y va a ir sumando características técnicas novedosas a lo largo del curso, partiendo de una página HTML hasta llegar a la publicación de la información de una base de datos en Internet.
- Se pretende que el alumno aplique a su sitio todos aquellos criterios corporativos que cualquier empresa aplicaría a sus páginas web (homogeneidad y criterios de diseño gráfico en base a una imagen corporativa, organización adecuada de la información de cara al usuario de la página, etc.).
- De manera general va a conocer, debido a las visitas que debe realizar para confeccionar el diccionario de términos, muchos sitios web de interés para cualquier desarrollador de aplicaciones web. Esta forma de trabajar va a servirle para cuando en su desarrollo profesional tenga que conocer por sí mismo cualquier nueva tecnología emergente y tenga que bucear en Internet para obtener información referente a la misma.
- Todo el trabajo anterior se realiza en grupo, con las consiguiente ventajas que aporta la interacción con otros compañeros, ya que en un futuro también trabajará en grupo dentro de su puesto de trabajo.

De cara a cubrir el temario y objetivos de la asignatura se han preparado las prácticas que se muestran en la Tabla 2.

El sitio web desarrollado durante las prácticas será expuesto en una presentación final bajo los cánones de cualquier grupo de desarrollo que expone al cliente el trabajo realizado, utilizando una presentación multimedia de nivel profesional y cuidando los elementos de marketing que vayan dirigidos a convencer al cliente sobre las virtudes de la solución desarrollada.

P. 1.	Diseño de una página Web en HTML.	2 horas
P. 2.	Inclusión en la página web de un diccionario de términos de NTP.	2 horas
P. 3.	Instalación de un servidor web utilizando Apache Tomcat.	2 horas
P. 4.	Utilización de elementos Javascript y hojas de estilo a la página web.	2 horas
P.5.	Introducción a una herramienta de desarrollo de aplicaciones web: Forte for java.	4 horas
P.6.	Inclusión de applets en una página web.	2 horas
P.7.	Diseño de una aplicación de acceso a base de datos a través de web utilizando tecnología JSP.	6 horas
P.8.	Diseño e Implementación de componentes JavaBean.	6 horas

Tabla 2. Descripción de las prácticas de NTP.

7. Metodología docente.

En la asignatura NTP se utiliza una metodología docente orienta a grupos [3, 5] que promueve el trabajo continuado y optimiza el aprovechamiento de las tutorías. Además, se realiza el apoyo docente con la publicación de la asignatura en el CAMPUS VIRTUAL de la Universidad de Almería y la utilización de la herramienta de docencia virtual WebCT.

Esta metodología se centra en el trabajo en grupo por lo que las innovaciones de mayor importancia aplicadas en dicho método van encaminadas al aprovechamiento de esta forma de trabajo, como son la elaboración de cuestiones que cubran los objetivos de cada tema (realizar preguntas de examen), la realización de diferentes tipos de prácticas (que incluyen "concursos de definición de términos" y defensas de proyectos), y un mayor seguimiento de la evolución del alumno mediante un nuevo planteamiento de las tutorías, donde además de utilizar nuevas tecnologías (como apoyo mediante enseñanza virtual a través de Internet) se realizan entrevistas periódicas con los grupos.

El potencial de comunicaciones que ofrecen redes como Internet debe aprovecharse para facilitar la comunicación entre el profesor y el alumno. En nuestro caso, cada alumno dispone de una cuenta de usuario de la herramienta WebCT que les posibilita acceder a todos sus servicios: correo electrónico, foros, charlas, etc. Esta forma de comunicación va a permitir que el alumno pueda obtener información, formular preguntas o consultas sin tener que desplazarse o esperar a la hora de tutoría, evitándose las molestias que ocasionan las colas a la puerta del despacho. Por otro lado, la utilización de nuevas tecnologías acerca al alumno a las posibilidades técnicas de sus futuros puestos de trabajo y le prepara adecuadamente para que domine estos nuevos medios.

La metodología docente basada en trabajo en grupo apoyada por la herramienta de docencia virtual en NTP queda descrita en otro articulo presentado a esta misma edición de JENUI [4].

8. Sistema de evaluación.

La evaluación se basará en una serie de notas que se van tomando a lo largo del curso intentando realizar una *evaluación continua* de cada uno de los grupos y si fuera posible también individual. La calificación continua se fundamenta en las notas de los siguientes elementos:

- Ejercicios resueltos de manera individual o en grupo en las clases de problemas y en las sesiones de prácticas.
- Calificación de las listas de preguntas de examen presentadas para cada tema.
- Corrección de cada una de las sesiones de prácticas por separado.
- Participación en la preparación de alguno de los seminarios.

Para aprobar la asignatura *es imprescindible superar* cada una de las siguientes partes:

- *Relación de prácticas* a entregar en diferentes momentos del curso y en especial la práctica final (que debe ser un compendio de todo lo aprendido en el resto de sesiones) y que será presentada en clase bajo criterios de presentación comercial de una aplicación.
- *Prueba final* sobre los contenidos de la asignatura que estará compuesta por cuestiones de tipo test.

Para aprobar la asignatura debe superarse tanto el examen final cómo las prácticas. En caso de suspender alguna de las dos partes, se examinará en la siguiente convocatoria sólo de la parte suspensa.

La nota final de curso se obtendrá a partir de la media de la nota del examen y la de prácticas, corregida al alza por las anotaciones recogidas durante el curso y por la propia nota de prácticas.

9. Conclusiones

Desde 1998, año en el que comenzó a impartirse NTP, los resultados obtenidos en relación con el contenido de la asignatura y la metodología utilizada orientada a grupos de trabajo han sido muy satisfactorios. El gran interés por parte de los alumnos queda plasmado en los resultados de los trabajos realizados por los grupos, que rebasan con creces a las expectativas del profesor y los requisitos del proyecto.

Además se muestra una gran satisfacción con el sistema de evaluación. El alumno llega al convencimiento de que son ellos mismos los que se evalúan y el sistema planteado anteriormente hace que la nota obtenida sea un fiel reflejo de la formación y capacitación obtenida en la asignatura.

Figura 3. Algunos ejemplos del trabajo realizado por los grupos en NTP.

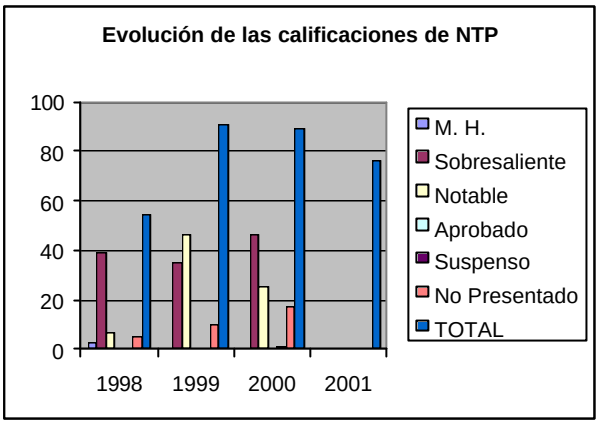


Figura 4. Evolución de las calificaciones de NTP.

Por otra parte, la utilización de una herramienta de docencia virtual como apoyo a la docencia de la asignatura fomenta al alumno en la participación del propio desarrollo de la asignatura ya que a través de esta herramienta pueden interactuar y comunicarse mediante los foros de discusión, listas de distribución para los grupos, chats con distintas habitaciones de dialogo, realización de test a través de Internet, calendarios y agendas de actividades, etc...

Referencias

[1] *Computing Curricula 2001 (CC-2001)*. Computer Science Volume. IEEE Computer Society and Association for Computing Machinery, Diciembre., 2001. http://www.computer.org/education/cc2001/

[2] Denning P.J. *Educating. A New Engineer.* Communications of the ACM. pp 83-97. Diciembre. 1992.

[3] Fernández A., Padilla N. y Peralta M. *Metodología docente orientada a grupos de trabajo*. III Jornadas de Enseñanza Universitaria de Informática (Jenui'97) 1997, pp 471-478

[4] Fernández A., Piedra J.A., Plaza M. A. *La docencia virtual como herramienta de apoyo en una metodología orientada a grupos de trabajo. Aplicación a la asignatura Nuevas Tecnologías de la Programación*. Propuesta de ponencia para las VIII Jornadas de Enseñanza Universitaria de Informática (Jenui'2002).

[5] Padilla N., Peralta M., Fernández A. *Metodología docente en introducción a la programación. Primeros resultados*. IV Jorn. de Enseñanza Universitaria de Informática (Jenui'98) 1998, pp 443-449.

La docencia virtual como herramienta de apoyo en una metodología orientada a grupos de trabajo. Aplicación a la asignatura Nuevas Tecnologías de la Programación

Antonio Fernández, José Antonio Piedra, Miguel Ángel Plaza

Departamento de Lenguajes y Computación
Universidad de Almería
{afm, jpiedra, maplaza} @ ual.es

Resumen

Nuestra metodología de trabajo en grupo ha sido muy satisfactoria para otras asignaturas (Introducción a la Programación) tal y como se expuso en las publicaciones presentadas en las Jornadas de Enseñanza Universitaria en Informática JENUI [2] y [4]. Por ello se pretende ir más allá en esa metodología mediante la incorporación de elementos de docencia virtual adaptados al trabajo en grupo y a su comunicación.

Por un lado, se plantea una metodología docente mediante elementos clásicos pero adaptados y potenciados por otros elementos innovadores (listas de preguntas de examen, concurso de definición de términos, presentación de proyectos, etc.), todo ello desde la perspectiva del trabajo en grupo.

Por otro, se pretende la máxima interacción entre el profesor y los miembros de un grupo, y además la interacción entre los diferentes grupos entre sí (a través de la docencia y las tutorías). Esta comunicación se realiza mediante la utilización de la herramienta de docencia virtual WebCT [5].

1. Introducción.

"Nuestra misión en la Universidad debe ser la de preparar a los estudiantes para afrontar la vida y para que sepan desenvolverse en el mundo laboral, dominado por las leyes de mercado e inundado por las tecnologías de la información". Esta cita de Denning [1] debe ser, a nuestro entender, el objetivo principal a conseguir por todo docente universitario. Debemos enseñar a nuestros alumnos a ser unos buenos Ingenieros, a realizar su trabajo de forma rigurosa y disciplinada, y a que sean capaces de autoaprender.

La educación universitaria se encuentra en continua evolución. Para poder formar al universitario según el modelo de Denning, anteriormente comentado, necesitamos realizar una serie de innovaciones que cambien los modos utilizados actualmente. Las transformaciones que se deben abordar engloban principalmente ámbitos como el currículum del estudiante, la actitud del mismo ante el nuevo modelo educativo y la metodología del docente. En este articulo nos centraremos en intentar innovar en estos dos últimos aspectos.

La metodología docente que proponemos se basa en elementos metodológicos clásicos (lecciones magistrales, tutorías, etc.) pero adaptadas y potenciadas por otros elementos innovadores (listas de preguntas de examen, concurso de definición de términos, presentación de proyectos, etc.), todo ello desde la perspectiva del trabajo en grupo. El resultado de la metodología de trabajo en grupo ha sido muy satisfactoria en otras asignaturas (Introducción a la Programación) y fue motivo de sendas publicaciones presentadas en las Jornadas de Enseñanza Universitaria en Informática JENUI [2] y [4]. Por ello no vamos a entrar a describir con detalle las características de esta metodología sino que vamos a avanzar algo más desde el punto de vista instrumental y se van a presentar elementos

de docencia virtual adaptados al trabajo en grupo y a su comunicación.

El objetivo último de esta metodología aplicada a la asignatura de Nuevas Tecnologías de la Programación es reproducir el ambiente laboral de los futuros profesionales mediante la adopción de los roles de un desarrollador que se encuentran ante el diseño de una aplicación para Internet.

Un profesional en estas circunstancias se verá obligado a: *colaborar con un grupo de trabajo*, a veces bastante heterogéneo desde el punto de vista profesional (puede incluir diseñadores gráficos), y *formarse en las últimas tecnologías* aplicables a dicha solución. En ese momento necesitará de habilidades de autoformación, así como recurrir a tutoriales y docencia virtual a través de Internet como medio más inmediato y económico para recibir esta formación en nuevas tecnologías.

2. Breve descripción de la asignatura.

Asignatura:	NUEVAS TECNOLOGÍAS DE LA PROGRAMACIÓN
Estudios:	INGENIERÍA TÉCNICA EN INFORMÁTICA DE GESTIÓN
Centro:	ESCUELA POLITÉCNICA SUPERIOR
Ciclo:	1º
Curso:	3º
Cuatrimestre:	2º
Carácter:	OPTATIVA
Créditos teóricos:	3
Créditos prácticos:	3
Área:	CIENCIAS DE LA COMPUTACIÓN
Descriptores BOE:	Nuevas Tecnologías de la Programación. Programación en Internet. Diseño de componentes. Aplicaciones basadas en componentes remotos.
Nº alumnos:	76

Tabla 1. Descripción de la asignatura de NTP.

Para conocer con detalle esta asignatura y su contenido recomendamos la lectura de la ponencia "Nuevas Tecnologías de la Programación en Internet" presentada en las jornadas de este año [3].

3. La enseñanza virtual como parte de la metodología docente y de evaluación.

Las Universidades se están viendo abocadas a convivir con dos modos de enseñanza: por un lado, la enseñanza presencial y por otro la enseñanza virtual o teleformación.

La Teleformación es, según FUNDESCO, *"un sistema de impartición de formación a distancia, apoyado en las TIC (tecnologías, redes de telecomunicación, videoconferencias, TV digital, materiales multimedia), que combina distintos elementos pedagógicos: instrucción clásica (presencial o autoestudio), las prácticas, los contactos en tiempo real (presenciales, videoconferencias o chats) y los contactos diferidos (tutores, foros de debate, correo electrónico)"*.

Este tipo de formación se está incorporando paulatinamente en la enseñanza reglada de las Universidades. Los formadores pueden establecer una metodología de la enseñanza-aprendizaje basada en un enfoque constructivista de la formación a través de la interacción y colaboración entre alumnos-alumnos y alumnos-tutor. Los materiales docentes que se pueden utilizar serán cualquier recurso que en formato electrónico pueda ser enviado a través de la red y se pueden usar una gran variedad de herramientas de comunicación tanto síncronas (como chat, voz-IP, videoconferencias, etc.), como asíncronas (foros de debate, correo electrónico, etc.).

En los últimos años se ha estado trabajando en la Universidad de Almería en un producto específico que facilita enormemente la labor del formador para cumplir con gran parte de su labor. Esta aplicación es WebCT [5], que ha sido desarrollado por el Departamento de Ciencia Informática de la Universidad de British Columbia, Canadá. Es una herramienta que facilita la creación de ambientes educativos basados en la Web tanto para cursos 'on-line' como para publicar simplemente materiales que complementen a cursos ya existentes (enseñanza reglada). Los usuarios necesitan conocer sólo el uso de algún navegador estándar para el acceso y para las tareas de diseño del curso, si es el

instructor. WebCT proporciona una interfaz para diseñar el aspecto del curso (colores, diseño de la página), una serie de herramientas educativas para facilitar el aprendizaje, la comunicación y la colaboración, y una serie de herramientas administrativas para ayudar al instructor en la distribución del curso.

WebCT se caracteriza principalmente por poseer:

- Posibilidades multimedia;
- Herramientas de auto-evaluación de los estudiantes y de evaluación on-line.
- Mantenimiento y distribución de notas.
- Un sistema de conferencias, que permite la presencia de un moderador.
- Sistema de correo-electrónico y de charla en tiempo real.
- Archivo de imágenes que se pueden buscar.
- Áreas de presentación de estudiantes y creación de páginas de presentación.
- Herramientas de diseño y gestión del curso.
- Control de seguridad y acceso.
- Posibilidades de grabación y ejecución del curso.

A continuación vamos a ver cómo hemos instrumentalizado la metodología docente orientada a grupos de trabajo a través de WebCT.

4. Trabajo en grupo.

El objetivo principal de esta metodología es que el alumno llegue a dominar lo aprendido, y el camino más corto es mediante el aprendizaje activo *(aprender a hacer, haciendo*) y cooperativo (*aprendizaje con otros*). Por ello creemos que nuestro modelo debe caracterizarse por:

- Promover el conocimiento por comprensión.
- Crear la necesidad de seguir aprendiendo.
- Animar al alumno a que asuma la responsabilidad de aprender convirtiéndose en el protagonista del aprendizaje.
- Crear un ambiente de trabajo personal y de colaboración entre alumnos.

En función de lo expuesto, se realiza la división del alumnado en grupos de trabajo con el objetivo de desarrollar el hábito de discusión de los problemas, la colaboración entre compañeros que tienen un objetivo en común y el flujo de conocimiento entre los mismos.

Los grupos van a funcionar tanto para realizar labores de teoría como de prácticas, constituyendo de esta forma una unidad de trabajo permanente para todos los aspectos de la asignatura y a lo largo de todo el cuatrimestre. Aunque prevalecen las actividades en grupo, no se van a excluir otras que son puramente individuales.

Como se puede apreciar en la siguiente figura, la interacción entre los agentes que intervienen en el proceso de enseñanza no se restringe a las relaciones entre los miembros del grupo, sino que además existe una interacción entre los diferentes grupos y además con el profesor (principalmente a través de la docencia y las tutorías). Este fenómeno de comunicación es el que vamos a instrumentalizar con la utilización de una herramienta de docencia virtual (WebCT).

Para facilitar la comunicación se ha proporcionado a cada alumno una dirección de correo electrónico y una clave de acceso a WebCT.

A partir de este momento toda la comunicación se realiza a través de los elementos propios de esta herramienta: correo electrónico, chat, pizarra, foros y calendario. Además se dispone de otros elementos de gestión y organización de la información propia de la asignatura: contenidos de teoría y práctica, autoevaluación, listado de términos y listado de bibliografía. Además, iremos desarrollando otros elementos complementarios, como por ejemplo el concurso de términos y una herramienta para FAQs (preguntas más frecuentes.

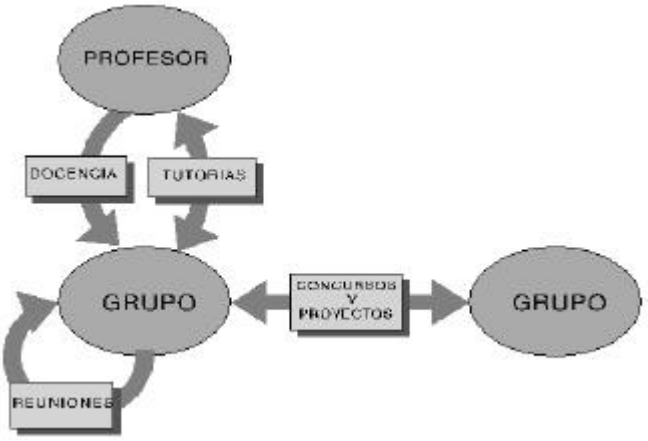


Figura 1. Interacciones en el modelo de enseñanza basado en grupos de trabajo.

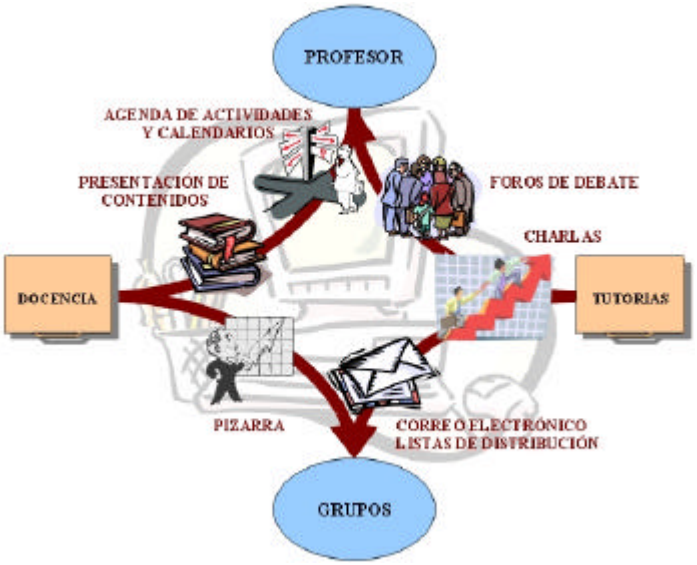


Figura 2. Interacción mediante elementos de docencia virtual.

Para facilitar la comunicación entre los miembros de un mismo grupo de trabajo vamos a definir una serie de elementos propios del grupo además de los elementos públicos definidos para todos los alumnos de la asignatura:

- Listas de distribución de correo cerradas para cada uno de los grupos.
- Un foro para debate y opinión cerrado para cada grupo.
- Además van a diseñar una página web propia donde van a publicar sus practicas así como aportaciones como grupo al conocimiento de la asignatura (por ejemplo enlaces de interés).

5. Clases de teoría y de prácticas.

Vamos a ver cuales son los elementos didácticos innovadores que complementan las clases de teoría y prácticas. Este conjunto de elementos cambia de manera considerable el punto de vista de los alumnos. Hasta este momento se establecía como condición necesaria para un apropiado seguimiento y aprendizaje de la asignatura la asistencia del alumno tanto en las clases presenciales de teoría como de prácticas y tutorías. A partir de la incorporación de estos elementos de docencia virtual, aunque es muy aconsejable el aprovechamiento al máximo de las horas presenciales, se abren muchas posibilidades para que un alumno con dificultad para asistir a clase pueda realizar un seguimiento apropiado de la asignatura gracias a toda la información y herramientas proporcionadas a través de Internet.

En primer lugar hay que destacar que tanto los *índices de contenidos* de teoría como de practica se encuentran disponibles en WebCT. Estos índices son sensibles a su selección de manera que despliegan información detallada sobre los contenidos, enlaces web de interés, descripciones bibliográficas, archivos con código fuente y código ejecutable, etc.

La organización y temporización de la asignatura se encuentra perfectamente descrita a través del *calendario* que proporciona WebCT, donde además podemos indicar fechas de entrega de prácticas, reuniones o eventos puntuales. Por tanto, este calendario va a mantener una agenda con todas las actividades de la asignatura y esto va a ser muy útil sobre todo a aquellos alumnos que tienen problemas de asistencia presencial a las clases.

Al finalizar cada tema de teoría, los grupos de trabajo confeccionarán una serie de preguntas, junto con sus respuestas, cubriendo cada uno de los objetivos propuestos inicialmente, es lo que hemos denominado *listado de pregunta de examen*. El perfil de las preguntas debe orientarse a la comprensión de la materia evitándose las puramente memorísticas. Las diversas listas de cuestiones serán intercambiadas entre los miembros del mismo grupo para que puedan autoevaluarse con las preguntas de los compañeros. Después serán recopiladas por el docente y, tras un proceso de revisión, se remitirán al resto de los grupos. El objetivo de este trabajo es que el alumno repase de forma continuada la asignatura y que comprenda cada uno de los objetivos propuestos.

Una vez que el profesor haya comprobado que las preguntas son apropiadas, se incluirán dentro del catálogo de preguntas de evaluación de WebCT. Un par de semanas después de

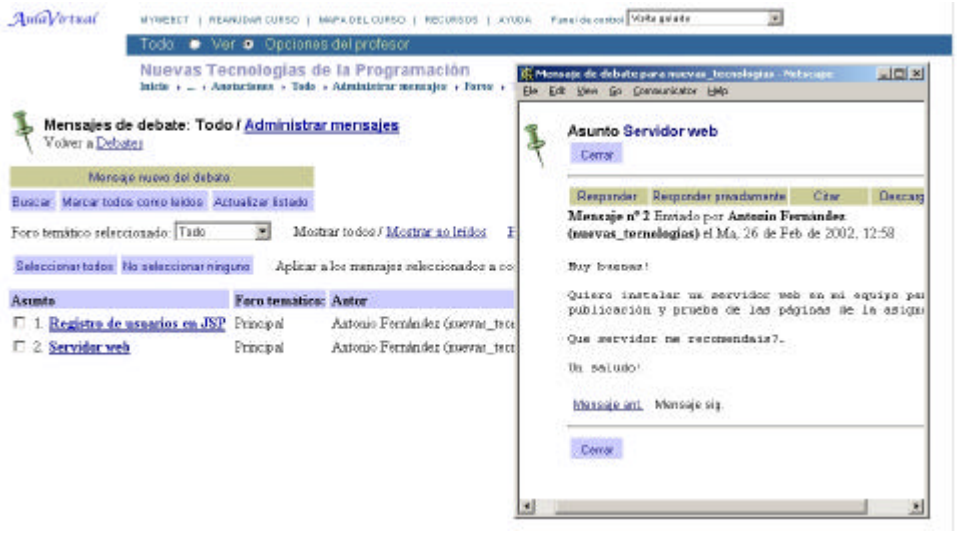


Figura 3. Foro sobre Nuevas Tecnologías de la Programación en WebCT.

acabar un tema cada alumno pasara una *autoevaluación* basada en preguntas de tipo test extraídas del catalogo confeccionado por sus compañeros de clase. Esta prueba no tiene tanto interés desde el punto de vista de la evaluación como de comprobar la asimilación de contenidos. Al final del curso y siguiendo esta misma dinámica se realizará un examen final cuyo objetivo es establecer una nota de teoría como parte de la calificación final.

En cuanto a las *prácticas*, cada grupo tendrá que resolver unos cuantos ejercicios en grupo, siguiendo el proceso de discusión ya mencionado, y cada miembro de grupo resolverá individualmente algunos más. Las últimas prácticas de la asignatura serán pequeños proyectos de poca complejidad. Esto permitirá al alumno desarrollar las fases de análisis y diseño antes de implementar el programa y adjuntar una documentación al desarrollo, a pesar de que éste sea muy simple.

Una vez evaluadas las prácticas por el profesor se irán publicando en la página web propia de cada grupo.

6. Tutorías.

El tercer gran pilar de la metodología docente que se plantea es la tutoría u horas de atención al alumnado. En realidad gran parte de este tiempo (3 horas semanales) se ha reconvertido en espacios dedicados a entrevistas con los grupos de trabajo, cambiando así el concepto tradicional de tutoría. Cada grupo dispondrá de una sesión de 30 minutos aproximadamente cada dos semanas, a la cual van a asistir todos los integrantes del grupo. Ésta se dedicará a:

- Resolución de dudas que una vez discutidas por el grupo no hayan sido clarificadas satisfactoriamente.
- Presentación del trabajo pendiente (como puede ser entrega de prácticas, de listas de cuestiones sobre los objetivos de cada tema, etc.).
- Charla sobre la evolución de la asignatura o sobre cualquier problema personal que afecte al trabajo en grupo.

Además de estas entrevistas se disponen de horas de tutorías individualizadas donde cada alumno puede hacer consultas propias, bien de manera presencial en el despacho o a través de los mecanismos de comunicación no presénciales que vamos a describir a continuación:

- *Correo electrónico,* el alumno se dirige al profesor para hacerle una consulta privada y asíncrona. El profesor va a publicar el horario que va a dedicar para responder estos correos, al menos un par de veces a la semana.
- *Foros de debate,* (Figura 3) el profesor participará en estos foros al igual que en el caso del correo electrónico al menos un par de veces semanales. El foro proporciona un método público y asíncrono de comunicación que facilita (a falta de un sistema FAQ) un catálogo de preguntas y respuestas a consultar por otros alumnos con idénticas dudas. En nuestro caso se han definido los siguientes foros: Foro de Teoría, Foro de Practicas, Foro de Nuevas Tecnologías de la Programación de manera general (recoge comentarios o tecnologías que no se encuentran enmarcadas de manera estricta en los contenidos propios de la asignatura), Foro Libre que permite una comunicación sin ningún tipo de reglas ni condiciones (se pueden tratar temas de ocio, sociedad, universidad, etc.). A estos foros se suman aquellos definidos para cada uno de los grupos que son privados para los miembros del propio grupo y donde ni siquiera el profesor tiene acceso.
- *Chat,* esta forma de comunicación proporciona un medio público y síncrono de comunicación no presencial. Esto conlleva la sustitución de dos horas de tutorías presénciales en despacho por la disponibilidad durante las mismas del profesor a través de chat. Se han abierto diferentes canales chat: Chat para Tutorías (abierto sólo dos horas a la semana), Chat para contenidos y organización de la asignatura y Chat Libre.
- *Pizarra,* consiste en una ventana con herramientas de texto y dibujo que permite al profesor expresarse con más detalle gráfico que en los foros y chats. Aunque es un medio de comunicación síncrono, también se pueden grabar las

contestaciones más importantes para presentarlas más adelante cuando algún alumno vuelva a preguntar por la misma cuestión. Dentro de la asignatura se organiza durante el mismo tiempo dedicado al chat, de manera que se alternará entre uno u otro sistema dependiendo de si la conversación chat necesita de una explicación gráfica que se puede proporcionar con la pizarra.

7. Sistema de evaluación.

Como ya hemos comentado anteriormente el sistema de evaluación va a ser continuo a lo largo de todo el curso. La evaluación final de la asignatura es el resultado de un compendio de calificaciones o evaluaciones parciales:

- Entrega de practicas, es fundamental el seguimiento de la asignatura día a día y aunque no sea obligatoria la presencia si que es conveniente que se realicen entregas periódicas de prácticas y ejercicios que comprueben el adecuado ritmo de trabajo y seguimiento de la asignatura por parte de los alumnos.
- Defensa en grupo y/o individual de alguna de las prácticas más importante realizadas.
- Evaluación de las preguntas de examen presentadas para cada tema.
- Realización de seminarios.
- Examen final de tipo test.

La *defensa en grupo* consiste en un simulacro de presentación comercial. Los alumnos del grupo asumen el rol de un grupo de desarrolladores que presentan de manera convincente una aplicación a sus clientes. Para ello los alumnos utilizan todos aquellos recursos que estimen necesarios (presentación PowerPoint, cañón multimedia, memorando técnicos, etc.). El profesor elige "in situ" a un portavoz, de esta manera se asegura que todos los miembros de grupo se han preparado la presentación. Y será este el que despliegue la presentación ante el resto de compañeros de clase que se dispondrán a preguntar sobre cualquier detalle técnico o funcional propio de la aplicación en vías de establecer a eficiencia y calidad de la solución. Al finalizar la presentación serán los alumnos de clase los que otorguen la nota mediante su satisfacción con lo expuesto. Esta es una forma de desmitificar el hecho de que las notas las pone el profesor.

El *examen final* se realizará a través de WebCT. Agilizándose de manera significativa el proceso de evaluación pues los resultados se obtienen de manera inmediata y además se proporciona al alumno en tiempo real el catálogo de soluciones correctas, con lo cual el proceso de evaluación tiene un marcado carácter formativo.

8. Resultados de la metodología.

Los primeros resultados se han obtenido a partir de la interpretación de las encuestas de calidad de enseñanza que realiza la Unidad de Evaluación de la Universidad de Almería anualmente, y aunque se realizaron antes de que se introdujeran los elementos de docencia virtual, puede ser muy significativa desde el punto de vista de la metodología de trabajo en grupo:

- La metodología propuesta en este articulo suscita un mayor interés por la asignatura, el alumno consigue una buena compresión y asimilación de los conceptos y una considerable satisfacción con lo aprendido.
- Además se muestra una gran satisfacción con el sistema de evaluación.
- La actitud y actuación de los grupos es muy positiva lo cual genera un buen ambiente de trabajo en clase.
- Los alumnos muestran gran interés por participar en la toma de decisiones que tienen que ver con la marcha de la asignatura y con el proceso de enseñanza.
- La valoración del profesor se puede considerar buena; contrastándola con la del año anterior se deduce que esta metodología favorece el grado de satisfacción del alumno con relación a un mismo profesor. El hecho de que el alumno participe en el proceso de calificación final favorece la "imagen" del profesor que se desprende de la "culpa" de ser quién aprueba o suspende.

9. Conclusiones.

Si analizamos el experimento metodológico que se esta llevando a cabo se puede concluir que esta metodología encuentra *algunos problemas*:

- Logísticos, como el hecho de que para el trabajo en grupo los alumnos necesitarían salas adecuadas equipadas con medios bibliográficos y de comunicación.
- El experimento ha sido posible gracias a que el número de alumnos matriculados no ha sido muy elevado, ya que si este número fuera superior habría que replantear algunas de las actuaciones que hemos descrito.
- Otro problema es que el alumno no está habituado a esta forma de trabajar, con lo cual pasa un cierto tiempo hasta que llega a confiar en ella, aunque demuestra mucho más entusiasmo que con los sistemas tradicionales.

Los resultados obtenidos indican algunas *ventajas*:

- El alumno adquiere una formación más rica pues además de llegar a dominar las cuestiones relacionadas con los contenidos propios de la asignatura, dispone de mayor capacidad de autoaprendizaje, de colaboración y de comunicación de todo lo aprendido.
- El hecho de que el alumno sea capaz de aprender nuevas habilidades por sí sólo es fundamental debido a que se va a desenvolver en un ambiente tecnológico en continua transformación.

Por último, sólo nos quedaría desear que este método se extendiera a un buen número de asignaturas, lo cual permitiría su total aceptación. La propuesta es factible debido a que es aplicable a asignaturas de características similares (sobre todo optativas) de las que abundan en los nuevos planes de estudios.

Referencias Bibliográficas.

[1] Peter J. Denning. *Educating. A New Engineer*. Communications of the ACM. pp 83-97. December. 1992.

[2] Antonio Fernández, Nicolás Padilla y Mercedes Peralta. *Metodología docente orientada a grupos de trabajo.* III Jornadas de Enseñanza Universitaria de Informática (JENUI´97) 1997, pp 471-478. Madrid. 1997.

[3] Antonio Fernández, José Antonio Piedra y Miguel Ángel Plaza. *Nuevas Tecnologías de la Programación en Internet.* Propuesta para las Jornadas de Enseñanza Universitaria de Informática (JENUI´02). Cáceres. 2002.

[4] Nicolás Padilla, Mercedes Peralta y Antonio Fernández. *Nueva Metodología docente en Introducción a la Programación*. IV Jornadas de Enseñanza Universitaria de Informática (JENUI'98), pp 443-449. Andorra. 1998.

[5] Tutorial WebCT. Dirección: http://www.WebCT.com Enlaces de interés.

Enlaces de interés.

- http://www.ciberaula.es/
  Instituto Calasanz de Ciencias de la Educación. Entre sus objetivos están la investigación educativa, la formación del profesorado y dar servicios integrales a centros educativos
- http://www.ciberaula.es/quaderns/
  QuadernsDigital.net es una revista de nuevas Tecnologías en la Educación. Incluye Base de datos de recursos educativos formada por alrededor de 2.200 URLs
- http://www.educared.net/
  Página Web que ofrece infraestructuras y servicios sobre educación, además de experimentar con metodologías que incorporen nuevos usos de la red para las innovaciones pedagógicas

Utilización de Internet para la enseñanza de sistemas digitales

Miguel A. Vega Rodríguez, Juan M. Sánchez Pérez, Manuel Rubio del Solar,
Francisco Chávez de la O, Juan A. Gómez Pulido
Departamento de Informática. Universidad de Extremadura
Escuela Politécnica. Campus Universitario, s/n. 10071 Cáceres. Spain
E-mail: mavega@unex.es. Fax: +34-927-257202

Resumen

Las nuevas tecnologías de la información y la comunicación han creado grandes expectativas en educación, pues permiten superar las limitaciones de tiempo y lugar, abaratando incluso los costes. Además, no debe olvidarse que los medios informáticos ofrecen una serie de características que favorecen el aprendizaje significativo a través de actividades de tipo interactivo. Por eso, nos parece importante su aplicación a la enseñanza de la Informática, y en particular de los sistemas digitales. En esta ponencia se presenta un sistema multimedia, basado en Internet, que se está desarrollando con el fin de aplicarlo a la docencia de sistemas digitales en la asignatura Fundamentos de Informática. La ponencia presenta una descripción general de dicho sistema, así como de las herramientas y métodos utilizados para su construcción.

1. Introducción

Internet se está convirtiendo en un importante recurso docente gracias a que permite superar las limitaciones de lugar y tiempo, abaratando costes. Tampoco debe olvidarse el efecto que la interactividad tiene en el proceso de aprendizaje. Creemos, por tanto, muy importante dedicar esfuerzos en la elaboración de propuestas y prototipos de enseñanza para impartir docencia a través de Internet. En esta línea, desde 1998 nuestro grupo de investigación ha trabajado en diversos proyectos como EDONET ([4], proyecto de investigación IPR98A029) o TEDA ([7], proyecto de investigación IPR00A003). Gracias a EDONET (Entorno para la DOcencia sobre interNET) se ha creado un entorno generalista para la enseñanza a través de Internet, entorno que ha sido aprovechado en proyectos posteriores como TEDA (Tele-Enseñanza para Discapacitados Auditivos). El proyecto TEDA ha permitido el desarrollo de una plataforma audiovisual basada en Internet para ayuda a la enseñanza de la lengua. Aunque principalmente se está utilizando para la tele-enseñanza de la lengua a discapacitados auditivos, son muchos los posibles campos de aplicación. Por ejemplo, se puede utilizar sin cambio alguno para la enseñanza del Español a extranjeros o en educación infantil, con unos beneficios similares a los que se están consiguiendo para el caso de discapacitados hipoacúsicos.

En la actualidad, nos encontramos inmersos en el proyecto SD2I (Sistema para la Docencia de Sistemas Digitales a través de Internet). Este proyecto surge de la aplicación de nuestra investigación a la docencia que impartimos. Como ocurre con el proyecto TEDA, SD2I también se apoya en EDONET. El objetivo global es el desarrollo de un sistema para la enseñanza, control docente y evaluación del aprendizaje a través de Internet de parte de la materia de la asignatura Fundamentos de Informática. En particular, este sistema se centra en el temario impartido durante el primer cuatrimestre y dedicado a los sistemas digitales.

La asignatura Fundamentos de Informática es una asignatura obligatoria anual, de 15 créditos, del primer curso de la Ingeniería Técnica de Telecomunicaciones: Sonido e Imagen (ITTSI), impartida en la Escuela Politécnica de Cáceres de la Universidad de Extremadura (UEX), y con un total de 119 alumnos en el curso académico actual. El objetivo fundamental de la asignatura es dar una visión general y práctica de la Informática en sus dos vertientes: hardware y software. Por este motivo, y al ser anual, se ha divido en dos

grandes módulos, impartiéndose cada uno en un cuatrimestre:

- Módulo I: Introducción. Representación de la Información. Sistemas Digitales.
- Módulo II: Programación. Estructuras de Datos. Sistemas de Comunicación.

El sistema SD2I se centra en el primer módulo, y en particular en los temas siguientes:

- Tema 3. Especificación e implementación de sistemas combinacionales.
- Tema 4. Módulos combinacionales básicos.
- Tema 5. Especificación e implementación de sistemas secuenciales.

El lector puede consultar la referencia [6] para obtener más información sobre el programa de la asignatura, y los temas específicos dedicados a la enseñanza de sistemas digitales.

El resto de la ponencia se organiza como sigue. En la sección 2 se da una visión general de la plataforma que se está construyendo para la enseñanza de los sistemas digitales, indicando sus características fundamentales. La sección 3 detalla brevemente las herramientas y métodos que se siguen para la generación de los contenidos docentes, ejercicios y control del alumnado. Mientras en la sección 4 se indica el trabajo futuro que se tiene previsto realizar para completar el sistema. Finalmente, en la sección 5, se presentan las conclusiones.

2. Descripción general de SD2I

El proyecto SD2I se está desarrollando durante el curso académico actual, con la intención de tenerlo totalmente preparado para utilizarlo en el siguiente curso académico. El proyecto se inició tras la obtención de una ayuda para Proyectos de Innovación Docente, concedida por el Vicerrectorado de Innovación Educativa y Calidad Docente, y el Instituto de Ciencias de la Educación (ambos de la Universidad de Extremadura). Nuestro objetivo es facilitar y dotar a los alumnos de una plataforma de enseñanza, adaptada a sus necesidades y características individuales, que les permita aprender de manera progresiva los conceptos más importantes sobre sistemas digitales, aprovechando las posibilidades que ofrecen las nuevas tecnologías (Internet, software multimedia,...). En la actualidad, la plataforma combina técnicas propias en lenguaje HTML [3], Java [1] y CGI [9], junto con software comercial (Macromedia Authorware [2]). Además, se encuentra estructurada en tres grandes secciones: teoría, ejercicios y control. En los siguientes subapartados se describirán todas ellas.

2.1. Sección de teoría

En la sección de teoría se dan todos los conceptos teóricos que el alumno debe aprender sobre sistemas digitales. Esta sección se estructura en diversos temas, donde cada tema a su vez se subdivide en varias lecciones. Las lecciones son accesibles desde cualquier computador conectado a Internet, desde el que se puede seleccionar fácilmente la materia a estudiar (tema-lección), interactuando con ella. De esta forma los alumnos pueden acceder al sistema tanto desde su centro universitario como desde su propio domicilio, con lo que se potencia el trabajo en casa y el autoaprendizaje, redundando en una mejora de la calidad de la enseñanza.

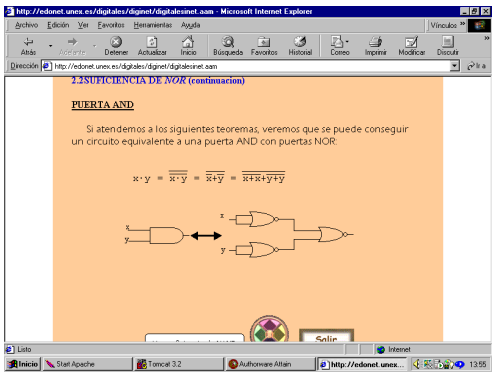


Figura 1. Una de las páginas web de la sección de teoría

En conclusión, el alumno puede seleccionar el tema y lección concreta para comenzar con el estudio de los conceptos asociados, o continuar el trabajo de días anteriores. Como regla general, cada lección está formada por varias páginas web que contienen los conceptos correspondientes, que se muestran mediante textos, imágenes, tablas, animaciones, etc. El alumno tiene libertad en todo momento para avanzar o retroceder dentro de una lección, así como para salir de la misma o incluso entrar en otra distinta. Es decir, junto a la facilidad

de uso del sistema también destacamos el hecho de que el alumno puede autocontrolar su ritmo de trabajo. En la figura 1 se presenta una de las páginas web asociadas con la sección de teoría, en este caso dentro del tema 2 y la lección "2.2. Suficiencia del Operador NOR".

2.2. Sección de ejercicios

En la sección de ejercicios el alumno puede responder a colecciones de preguntas de autoevaluación (de 5 a 20 preguntas por colección), siendo en todo momento tutorizado y corregido por el sistema. Las preguntas de autoevaluación están relacionadas con el tema y lección en la que se encuentra en cada instante el alumno. Los ejercicios de autoevaluación pueden ser de muchos tipos. Como ejemplo, a continuación enumeramos algunos de ellos:

- Se realiza una pregunta tipo test en la que el estudiante debe marcar la opción correcta.
- Se muestra una expresión algebraica y el alumno debe seleccionar la equivalente entre varias opciones.
- Se muestra el esquema de varios circuitos y el estudiante debe seleccionar el correcto según el enunciado.
- En ambos lados de pantalla aparecen imágenes que el alumno debe emparejar (p.e. la representación europea de las puertas lógicas con la representación norteamericana). El emparejamiento se realiza haciendo clic sobre las imágenes y arrastrándolas hacia su supuesta pareja.
- En un lado de pantalla aparecen expresiones algebraicas (u otro tipo de texto) y en el otro los circuitos correspondientes. El estudiante debe emparejarlos, al igual que en el caso anterior.
- Se muestran en pantalla mapas de Karnaugh que el alumno debe resolver, bien para indicar su solución o bien para indicar cuál de todos ellos es la respuesta correcta.
- Se muestra un circuito por pantalla que el estudiante debe analizar de varias formas según el caso: indicando el valor de la señal de salida, de las señales intermedias en varios puntos de circuito, etc.
- Se realiza cierta pregunta apoyada con la imagen de un circuito, su tabla de verdad, etc. y el alumno debe teclear la respuesta correcta: expresión algebraica asociada, tipo de circuito, …
- Etc.

Entre las funcionalidades ofrecidas en la sección de ejercicios destacan: el poder avanzar libremente de un ejercicio a otro, o salir incluso de la colección de ejercicios, el poder hacer clic sobre un botón para obtener directamente la solución del ejercicio, o sobre otro que permite obtener ayuda ("pistas") para solucionarlo. Los resultados de esta interactividad (tiempo de aprendizaje, respuestas a las cuestiones, evaluaciones, pidió ayuda o no, necesitó consultar la solución, etc.) son gestionados y almacenados por la sección de control de SD2I. Así conoceremos el rendimiento de los estudiantes y la calidad de las lecciones desarrolladas (control docente y evaluación de los alumnos, generación de estadísticas, etc.). La figura 2 presenta, como ejemplo, uno de los múltiples ejercicios de autoevaluación del sistema.

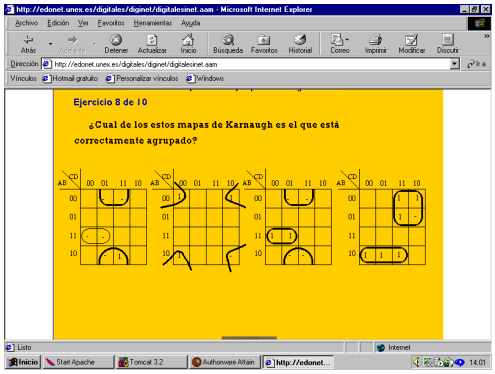


Figura 2. Ejemplo de ejercicio de autoevaluación

2.3. Sección de control

La sección control se apoya en el hecho de que, según hemos explicado, la información generada como consecuencia de la interactividad del usuario con el sistema se registra internamente. Es decir, el servidor monitoriza y almacena, en un formato dado y dentro de una base de datos, los resultados del proceso de aprendizaje. De manera que esta información puede ser utilizada por los educadores como elemento de juicio para poder evaluar el rendimiento del alumno. Esta sección sólo es accesible por los profesores, que entran en

la misma de forma local o remota, mediante clave de acceso. Los alumnos sólo pueden conocer algunas parcelas de la información del resultado de su interacción (calificación de la prueba, porcentajes de éxito, etc.), que obtienen a través de la sección de ejercicios. Dentro de esta sección lo primero que se le muestra al profesor es una página con el listado de alumnos, de los cuales el profesor seleccionará el alumno a evaluar. Posteriormente, en otra pantalla se mostrarán las distintas sesiones de trabajo realizadas por ese alumno, según fecha-hora y tema, y de nuevo el profesor deberá indicar la sesión deseada. De esta forma obtendrá los datos asociados al alumno y sesión indicados. La figura 3 muestra una pantalla con el seguimiento de la evolución de un alumno tras realizar una colección de 10 ejercicios del tema 2. En esta figura podemos observar que se registran datos como: alumno que realizó el test, fecha y hora de realización, el tipo de ejercicios realizados, el tiempo que se ha tardado en responder a cada pregunta, el número de intentos o fallos, si pidió solución, etc.

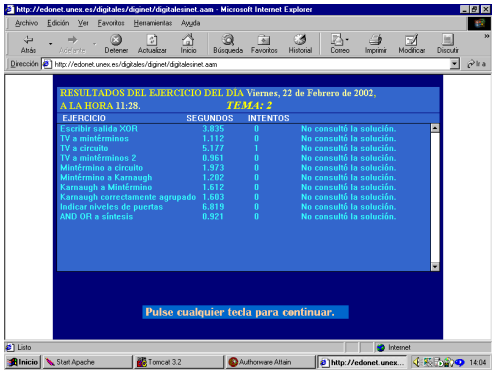


Figura 3. Página web con información sobre los resultados obtenidos tras la realización de una prueba

3. Herramientas y métodos

Para el desarrollo de SD2I se dispone de un servidor que sirve de soporte al sistema docente a través de Internet. El servidor, bajo Windows NT Server 4.0, ofrece servicios de administración y publicación de páginas web (WWW), de transferencia de ficheros (FTP), Gopher, correo electrónico, listas de distribución de correo compatible con *Majordomo*, así como diversos dispositivos multimedia. Así mismo, se dispone de software comercial (Macromedia Authorware [2]), para el diseño de páginas web, sistemas de preguntas y respuestas, y elaboración de contenidos docentes. En particular, se están construyendo un conjunto de estructuras genéricas y bibliotecas con una serie de elementos Authorware que posteriormente se utilizan para el desarrollo de los contenidos docentes, ejercicios de autoevaluación y el sistema de control del alumnado. En la figura 4 se muestra un ejemplo de las estructuras generadas con Authorware en el diseño del sistema SD2I.

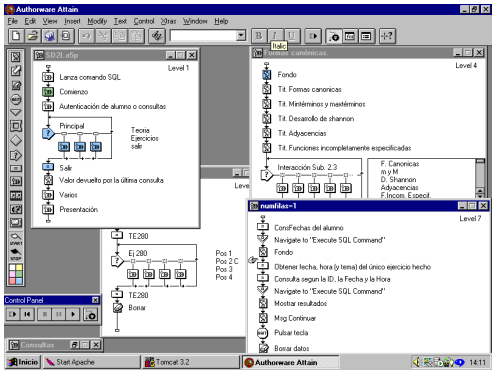


Figura 4. Estructuras y bibliotecas generadas mediante Authorware

Respecto a los contenidos docentes, indicar que éstos se almacenan con una estructura y formato estándar, de manera que es posible generarlos en PCs no conectados en red y ajenos al servidor, para posteriormente ser administrados por el servidor.

Por su parte, toda la información obtenida de la realización de los ejercicios de autoevaluación por parte del alumnado es almacenada internamente, sin que el usuario tenga conocimiento de ello. Esta información generada automáticamente por el sistema se almacena en una base de datos que contiene las tablas necesarias para la gestión de toda la información: información de alumnos, información de profesores (para el acceso restringido al sistema), información de ejercicios, información de interacción con el sistema, etc. La conexión interna entre nuestro sistema y la base de datos se realiza a través de ODBC (*Open DataBase Connectivity*), por lo que es fácil cambiar el

sistema de gestión de base de datos. Esta conexión nos permite, entre otras cosas, modificar los datos de las diferentes tablas u obtenerlos para mostrarlos en pantalla cuando un profesor desee consultarlos. Las confidencialidad de los datos generados está asegurada por dos motivos. Por un lado, al residir éstos en el servidor, no cabe la posibilidad de "retocarlos" desde fuera. Por otro lado, el acceso a los mismos se encuentra restringido mediante clave. Clave que sólo posee el profesor correspondiente, y que se gestiona mediante dos niveles de autentificación. En primer lugar, existe un nivel de autentificación bajo Apache [5], para permitir el acceso al sitio web donde se encuentra el sistema SD2I. En segundo lugar, SD2I restringe el acceso a la base de datos mediante clave gestionada a través de ODBC. La figura 3 presentaba un ejemplo de supervisión por parte del sistema.

Respecto al sistema de claves incluido, no sólo cada profesor posee su propia clave, sino que cada alumno también posee la suya. De esta forma, cada alumno debe introducir su clave antes de entrar en el sistema, y así el sistema lo identifica de forma unívoca evitando: que un alumno suplante a otro y desvirtúe sus resultados académicos en la interacción del sistema, que un alumno conozca datos de interés sobre otro distinto, etc. Como ya hemos comentado, las claves de los profesores poseen mayores privilegios, puesto que permiten acceder a toda la información del sistema (también a la de supervisión de alumnado), siendo éstos los que dan de alta o baja a los alumnos.

4. Trabajo futuro

Aunque ya se pueden observar los primeros resultados del sistema SD2I (visitar la dirección [8]), lo cierto es que aún no se ha finalizado su construcción. De hecho, se ha pensado en la incorporación de nuevas capacidades de simulación, animación e interacción complejas, que reduzcan aún más el tiempo de aprendizaje, y aumenten más si cabe la motivación de los alumnos. También se pretenden añadir otras secciones para permitir que los alumnos puedan descargar material didáctico, ver la bibliografía más adecuada, informarse de las últimas novedades, consultar sus dudas (tutorías virtuales), encontrar respuestas a las preguntas más habituales (FAQs), etc.

Una vez se concluya la fase de construcción, parte del equipo de trabajo se dedicará a evaluar el sistema en profundidad, contemplando todas las posibilidades y casos que se puedan presentar. El listado de posibles errores se irá pasando, conforme se vayan detectando, al resto de componentes del equipo, que se dedicarán a la depuración de errores en paralelo.

Más aún, el sistema será evaluado por parte de los alumnos. Es decir, una vez concluido, se pedirá a los alumnos matriculados en la asignatura Fundamentos de Informática que evalúen el sistema desarrollado. De esta forma, se detectarán los defectos y virtudes del sistema SD2I frente al modelo de enseñanza tradicional. Esta evaluación se llevará a cabo mediante la utilización del sistema de manera masiva por parte de los alumnos, y la realización de encuestas a los mismos. Los errores y deficiencias que se detecten por parte de los alumnos, serán depurados, buscando que el sistema esté totalmente preparado para el curso académico siguiente. Finalmente, se realizará la documentación del sistema para de esta forma permitir su fácil mantenimiento y uso (tanto por profesores como por alumnos).

También es importante resaltar que Authorware permite tanto la edición de contenidos para su implantación en un servidor WWW y visible desde un navegador, como su preparación para la distribución en formato CD. Por este motivo, se prevé que la utilización final del sistema se hará mediante ambas alternativas: uso a través de Internet y distribución del sistema en soporte CD.

5. Conclusiones

Como parte de nuestros esfuerzos en la mejora de la enseñanza, debemos aplicar todos los recursos que nos proporcionan los avances técnicos, como las tecnologías multimedia e Internet, para desarrollar de la mejor manera posible la capacidad y conocimientos de nuestros alumnos. En este sentido, estas tecnologías también deben aplicarse en la enseñanza de la Informática, y en particular de los sistemas digitales.

En este trabajo hemos presentado el sistema SD2I, que se está desarrollando para la enseñanza,

control docente y evaluación del aprendizaje a través de Internet de la materia "Sistemas Digitales", impartida en la asignatura Fundamentos de Informática de primero de la ITTSI de la UEX. El sistema comenzó a desarrollarse a principios del curso académico actual (2001/2002), y se prevé su finalización antes de que acabe el mismo.

Una vez el sistema SD2I se encuentre totalmente desarrollado podrá ser utilizado no sólo dentro de esta asignatura, sino también dentro de otras muchas (Sistemas Digitales, Electrónica Digital, Sistemas Electrónicos Digitales, etc.), dedicadas a la temática de los sistemas digitales, y que se imparten en los primeros cursos de muchas de las Ingenierías en la mayoría de Universidades.

Referencias

[1] Campione, M.; Alrath, K.W. *The Java Tutorial. Object Oriented Programming for the INTERNET*. Addison-Wesley, 1998.

[2] Kellog, O.; Ziajka, J. *Authorware 5 Attain Authorized*. Peachpit Press, 1998.

[3] Ray, D.S.; Ray, E.J. *Mastering HTML 4.0*. Sybex, 1997.

[4] Sánchez, J.C.; Sánchez, J.M.; Gómez, J.A. *EDONET: Sistema Piloto para la Docencia a través de Internet*. VIII Congreso de Innovación Educativa en Enseñanzas Técnicas / I International Congress in Quality and in Technical Education Innovation, Donostia-San Sebastián, vol. 2, pp. 25-33, Septiembre 2000.

[5] The Apache Software Foundation. http://www.apache.org, 2002.

[6] Vega, M.A. http://atc.unex.es/mavega/FI.htm, 2002.

[7] Vega, M.A.; Sánchez, J.M.; Gómez, J.A.; Chávez, F.; Fernández, F. *Tele-Enseñanza: Investigación y Proyectos*. III Jornadas Multimedia Educativo: Nuevos Aprendizajes Virtuales, Barcelona, Junio 2001.

[8] Vega, M.A.; Sánchez, J.M.; Rubio, M.; Chávez, F. http://edonet.unex.es/Digitales/Digitales.htm, 2002.

[9] Weinman, W.E. *The CGI Book*. New Riders Publishing, 1996.

Tema estratégico

Fomento de habilidades de trabajo en grupo

Propuesta metodológica para la mejora de la calidad y la excelencia de la educación superior en informática mediante el fomento del trabajo en equipo

José María Gutiérrez, Javier Macías, José Ramón Hilera, José Antonio Gutiérrez
Dept. Ciencias de la Computación
Universidad de Alcalá
28871 Alcalá de Henares MADRID
e-mail: {josem.gutierrez, javier.macias, jose.hilera, jantonio.gutierrez}@uah.es

Resumen

Se presentan las ideas exploradas y aplicadas y los resultados obtenidos durante el curso 2001/02 con la intención de fomentar el trabajo en equipo entre los estudiantes de primero de Ingeniería en Informática de la Universidad de Alcalá (UA). El campo de trabajo elegido ha sido el de las asignaturas de Laboratorio de Programación, dado su carácter eminentemente práctico que permite el trabajo en equipo como forma natural de desarrollo.

1. Motivación

Tradicionalmente se han observado deficiencias en el rendimiento, comportamiento y actitud de los alumnos de Ingeniería en general, y de Informática en particular, en cuanto al trabajo en equipo.

Es posible clasificar los problemas observados en dos grupos: por un lado tendríamos problemas puramente *logísticos* y, por otro, problemas de actitud. En concreto, se han detectado los siguientes problemas causantes de las deficiencias en el trabajo en equipo del alumnado:

- Problemas de distanciamiento físico del lugar de residencia durante el curso de los estudiantes y el centro. En nuestro caso, los alumnos proceden de distintos lugares de residencia, alejados entre sí, y estos alumnos se desplazan diariamente a la Universidad realizando trayectos que superan la hora de duración en algunos casos. Esto es debido a que la Universidad se encuentra en la Comunidad de Madrid, donde es relativamente frecuente que las personas se desplacen a sus centros de trabajo o estudio a través de largas distancias. Además, la situación geográfica, periférica respecto a la Comunidad de Madrid, y casi en la de Castilla-La Mancha, la convierte en la Universidad para los alumnos de Guadalajara. Esto hace que alumnos que comparten clase puedan vivir a 100 Km de distancia.
- Problemas de falta de espacio destinado al trabajo en equipo de los alumnos. En el Edificio Politécnico de la UA, los lugares destinados al trabajo de los alumnos son la biblioteca [2], los laboratorios [3] y los corredores [4]. En la biblioteca no se puede realizar trabajo en común, ya que las normas necesarias de silencio no lo permiten, estando destinada más bien al estudio personal y la consulta de bibliografía. Los laboratorios están ocupados permanentemente por clases y no tienen horario de apertura libre para que los alumnos desarrollen su trabajo. Por último tenemos los corredores que, en el Edificio Politécnico, ocupan gran cantidad de espacio y están concebidos como una zona de encuentro para los alumnos, con bancos y mesas grandes y móviles, pero sin ningún material adicional de tipo técnico. Este es el único lugar del edificio en el que se puede realizar trabajo en equipo pero, por motivos obvios, no es el más recomendable y su uso no está muy extendido, aunque en ocasiones se pue-

den encontrar, en las zonas más tranquilas, grupos de alumnos con ordenadores portátiles propios realizando trabajos.

- Otra circunstancia que dificulta el trabajo en equipo es que las prácticas que se plantean en los laboratorios son suficientemente cortas, en el primer curso, para poder ser realizadas por un solo alumno. De hecho, es común que los alumnos formen equipos estables para los laboratorios de todas las asignaturas y se repartan el conjunto de las prácticas para realizarlas de forma individual. Con esto se consigue que los alumnos realicen trabajo en colaboración pero, por otro lado, este tipo de reparto es indeseable en la enseñanza puesto que es equivalente a que cursen sólo la mitad de las asignaturas cada uno. Esto no es lo que se pretende, dado que la unidad de desarrollo de trabajo en común debe ser la asignatura y no el conjunto de las asignaturas del curso.
- Por otro lado, tenemos el hecho de que las prácticas no se diseñan pensando en la realización en equipo, de forma que resultara evidente la posibilidad de repartir el trabajo. Tan sólo se crean prácticas de tamaño medio para que parezca necesario realizarlas por más de una persona y se deja en manos del alumno la decisión del reparto o colaboración. Esto es contraproducente, especialmente en los primeros cursos en los que los alumnos no están acostumbrados a tomar ningún tipo de decisión y al final el método de trabajo que aplican es sentarse los dos a la vez y que uno trabaje y el otro observe.
- Por último, tenemos la actitud que presentan los alumnos desde hace años en relación a sus compañeros. Esta relación se basa en una competición continua en lugar de una colaboración [1], lo que es totalmente contrario al trabajo en equipo.

Ante estas dificultades, debemos dedicar una atención especial al trabajo en equipo, ya que es uno de los objetivos generales que se persigue en estas titulaciones. Un Ingeniero en Informática debe trabajar en equipo, y es misión de la Universidad prepararlo para ello. Esto redundará en un aumento de la calidad y la excelencia de la educación del alumno, que es uno de los objetivos perseguidos por la LOU [6].

2. Objetivos

Dada esta situación, se plantea el objetivo de fomentar el trabajo en equipo entre los alumnos. Este objetivo general, particularizado al contexto en el que nos encontramos, se concreta en los siguientes objetivos más detallados:

- Iniciar desde el primer cuatrimestre del primer curso la transmisión a los alumnos de las técnicas del trabajo en equipo. Se usará para ello la asignatura de Laboratorio de Fundamentos de la Programación [7] por ser la que está más directamente relacionada con el trabajo que en su futuro profesional desarrollará en equipo.
- Conseguir que el alumno experimente la necesidad y descubra la utilidad del trabajo en equipo. Esto incluye la toma de conciencia de que en el trabajo en equipo es posible obtener un resultado mayor que la propia suma de las partes implicadas (*sinergia*). Para lograrlo, será imprescindible que el alumno aprenda, utilice y perfeccione las técnicas básicas del trabajo en equipo [5].
- Fomentar explícitamente el trabajo en equipo durante las horas de clase de laboratorio mediante ejercicios especialmente diseñados para ello.
- Facilitar el acceso de los alumnos a nuevas tecnologías basadas en Internet que permitan la colaboración y el trabajo en equipo a distancia.

Figura 1. Diagrama de iniciativas

3. Líneas de Trabajo

Con los objetivos planteados, se han estudiado dos líneas de trabajo. Por un lado se han desarrollado ejercicios para descubrir en el alumno la necesidad y la utilidad del trabajo en equipo y, por otro, se ha trabajado en posibles herramientas de colaboración a través de Internet como pueden ser los servicios de directorio avanzados o los sistemas de control de versiones distribuidos (Figura 1). También se ha considerado la posibilidad de utilizar herramientas de e-learning como tecnología de soporte que permitiera atacar el problema de la falta de espacio físico dedicado específicamente para el trabajo de colaboración, mediante la creación de un espacio de trabajo *virtual*.

De estas dos líneas de trabajo, en el momento actual, se ha empezado a trabajar en la primera (ejercicios), y se han planteado dos posibilidades:

a) Desarrollar un ejercicio común para la globalidad del grupo de laboratorio en el cual cada alumno o equipo tenga una misión concreta (Figura 2).

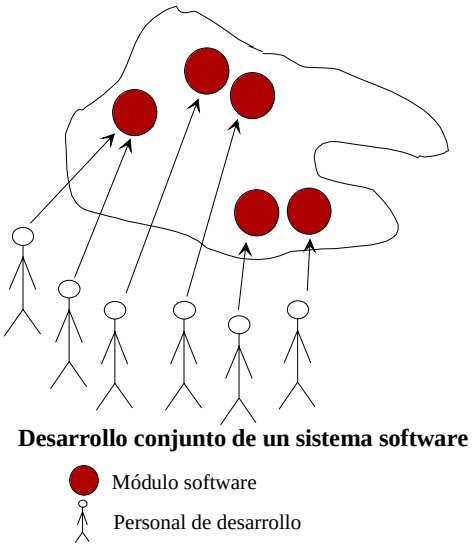


Figura 2. Esquema de un desarrollo en equipo

b) Crear una serie de ejercicios sencillos, distribuibles en varias sesiones, que requieran la colaboración (Figura 3).

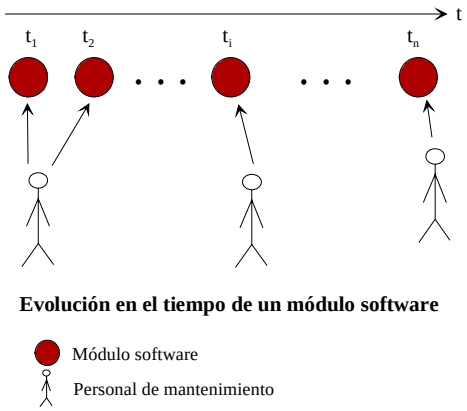


Figura 3. Esquema del mantenimiento de un módulo

La primera de las opciones plantea una serie importante de inconvenientes en el contexto en el que nos encontramos. Uno de ellos es la dificultad de implicar a los alumnos en un proyecto de esta envergadura en un laboratorio de primer curso debido a que acusan una importante falta de iniciativa, además de no estar asentada aún la relación de compañerismo, que con el tiempo se deberá desarrollar entre ellos, dado el poco tiempo que han trabajado aún juntos. Esta falta de relación hace difícil que tomen responsabilidades compartidas en un trabajo conjunto de esta magnitud. Otro obstáculo se debe a que completar el desarrollo de la aplicación precisará de la participación y el compromiso de todos los alumnos, cuestión que no es fácil de lograr teniendo en cuenta que aquellos alumnos que no tengan suficiente interés en la asignatura o que por alguna razón tengan que abandonarla, no aportarán su parte del trabajo, por lo que no se podría completar el sistema en su totalidad. Nótese que el hecho de no terminar el sistema sería una causa importante de insatisfacción entre los alumnos del grupo de laboratorio que sí hayan trabajado y entregado su parte. Estos inconvenientes, junto con el hecho de que el alumno en primero aún no posee, en general, suficientes conocimientos en programación, hacen casi inviable un proyecto de gran envergadura de desarrollo conjunto.

Debido a todo lo dicho, nos hemos decidido por la segunda opción (ejercicios de colaboración) y la hemos desarrollado de la forma que se explica en el siguiente apartado.

4. Ejercicios de colaboración

Tal y como se ha dicho, el objetivo de esta actividad consisten en lograr que los alumnos descubran la necesidad y ventajas del trabajo en equipo. Para ello, se utilizarán ejercicios en los cuales se enfrenten a un problema en el que deben trabajar juntos como requisito solicitado en el propio enunciado de los mismos. Conseguir que surja en los alumnos la necesidad de trabajar juntos, cuando no están acostumbrados a ello, puede ser muy difícil, por lo que es necesario que el enunciado les obligue expresamente a utilizar el trabajo en equipo.

Para conseguir el objetivo marcado, se plantean una serie de metas, un orden temporal para su consecución y un conjunto de ejercicios como medio para alcanzarlas, tal y como se desarrolla a continuación:

- Primera meta: mostrar a los alumnos los beneficios de rendimiento que se obtienen al utilizar el trabajo en equipo. Se planteará un ejercicio en el que se estipulará una serie de apartados de realización común y otros de realización separada. El tiempo de desarrollo estará limitado de forma que no sea posible terminar el ejercicio por separado; la colaboración será imprescindible. El ejercicio consistirá en un programa en Pascal, se deberá realizar en equipo, y será de sencilla realización, aportando el enunciado del mismo el diseño donde quedará patente los subprogramas a realizar y los datos a manipular, y encauzando totalmente las acciones de los alumnos. El programa debe ser sencillo para que los alumnos no tengan ningún problema en su desarrollo y así lo perciban en primera instancia, pero también que les permita darse cuenta de que no les será posible realizarlo por separado. Sin el trabajo de todos, el ejercicio no se podrá completar a tiempo. El tiempo para este ejercicio debe ser de 2 horas, el equivalente a una sesión completa de laboratorio.
- Segunda meta: hacer patente a los alumnos la necesidad de realizar un desarrollo de calidad cuando se trabaja en equipo, de forma que el cliente de nuestro producto (que en este caso será un compañero del equipo, esto es, un cliente interno) pueda continuar el desarrollo de dicho producto de una forma satisfactoria. Se establece un ejercicio de dos partes que obliguen a colaborar a los alumnos. Esto se conseguirá haciendo que la segunda parte precise de los resultados de la primera para su realización. Los alumnos realizarán este ejercicio agrupados según los mismos equipos que hayan formado para la realización de la práctica de evaluación de la asignatura, para que tengan una cierta confianza entre ellos. Los alumnos podrán producir, como resultado de su trabajo, tanto el código propiamente dicho como la documentación interna y externa que deseen. El enunciado de la segunda parte será desconocido hasta que se dé por terminada la primera, de manera que los alumnos trabajen tal y como lo hagan normalmente. Todos los alumnos recibirán el enunciado de la primera parte y tendrán un tiempo para realizarlo en torno a los 30 minutos, tiempo que se concretará en el enunciado y que dependerá del grado de dificultad del mismo. Terminado el primer ejercicio, los alumnos intercambiarán sus puestos de trabajo de forma que, en el nuevo puesto, sólo dispongan del código del compañero y la documentación que éste le haya dejado. Para la realización del segundo ejercicio se establecerá un tiempo superior al primero sin llegar al doble. Este tiempo será de unos 50 minutos. Todo el desarrollo del ejercicio debe poder completarse durante una sesión de laboratorio de dos horas. El tamaño de los dos ejercicios será similar, pero el tiempo para su realización distinto, debido a que en el segundo ejercicio los alumnos deben analizar primero lo que su compañero ha escrito.
- Tercera meta: asentar los conocimientos adquiridos sobre el trabajo en equipo. Para ello se repetirá el primer ejercicio, habiendo permitido a los alumnos pensar estrategias con las que afrontar un ejercicio de estas características. Para facilitar esto, el profesor debe realizar comentarios de técnicas e ideas que orienten a los alumnos pero dejando a su iniciativa la forma de afrontar el problema basándose en la experiencia de los ejercicios anteriores.

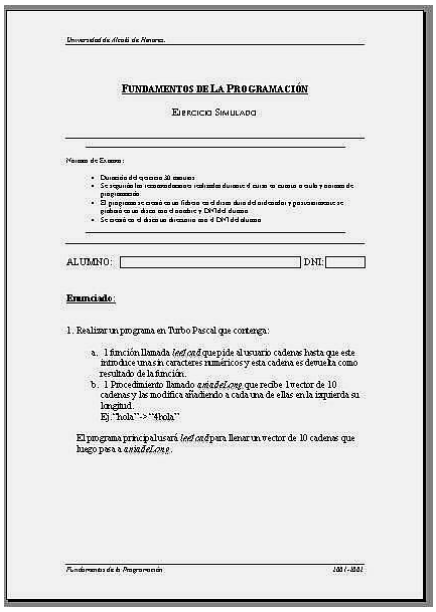
Es conveniente realizar los ejercicios en tres sesiones de laboratorio cercanas a la finalización

del cuatrimestre, para no entorpecer el desarrollo habitual de contenidos de la asignatura y para permitir que los alumnos tengan un nivel de conocimiento aceptable. También existe la posibilidad de utilizar un horario fuera del horario de laboratorio, e incluso realizar los ejercicios como actividad no obligatoria, pero esta última posibilidad parece poco recomendable según la actitud que muestran últimamente los alumnos.

4.1. Ejercicios realizados.

En el presente año lectivo sólo se ha efectuado el segundo de los ejercicios planteados (el correspondiente a la segunda meta), ya que su preparación y realización es más sencilla. Su desarrollo tuvo lugar en la asignatura de Laboratorio Fundamentos de Programación de primero de Ingeniería Técnica de Informática.

Tal y como se dijo, este ejercicio tiene dos partes, que denominaremos primer (Figura 4) y segundo ejercicio (Figura 5). En cada uno de ellos se encuentran varias secciones separadas, dedicadas a las normas del ejercicio, los datos de identificación del alumno y el enunciado del ejercicio propiamente dicho.



Universidad de Alcalá de Henares.

FUNDAMENTOS DE LA PROGRAMACIÓN

EJERCICIO SIMULADO

Normas de Examen:

- Duración del ejercicio 30 minutos
- Se seguirán las recomendaciones realizadas durante el curso en cuanto a estilo y normas de programación
- El programa ya creado en un fichero con el disco duro del ordenador y posteriormente se grabará en un disco con el nombre y DNI del alumno
- Se creará en el disco un directorio con el DNI del alumno

ALUMNO: DNI:

Enunciado:

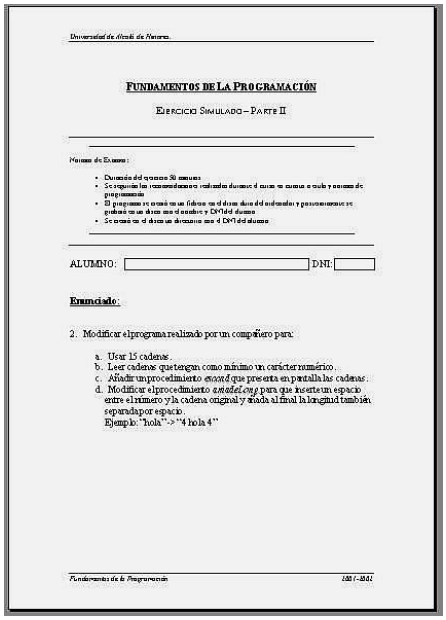
1. Realizar un programa en Turbo Pascal que contenga:

   a. 1 función llamada *leeCad* que pide al usuario cadenas hasta que este introduce una sin caracteres numéricos y esta cadena es devuelta como resultado de la función.
   b. 1 Procedimiento llamado *anadeLong* que recibe 1 vector de 10 cadenas y las modifica añadiendo a cada una de ellas en la izquierda su longitud.
   Ej: "hola" -> "4hola"

   El programa principal usará *leeCad* para llenar un vector de 10 cadenas que luego pasa a *anadeLong*.

Fundamentos de la Programación *1ª Ed. - 2001*

Figura 4. Primer ejercicio



Universidad de Alcalá de Henares.

FUNDAMENTOS DE LA PROGRAMACIÓN

EJERCICIO SIMULADO – PARTE II

Normas de Examen:

- Duración del ejercicio 30 minutos
- Se seguirán las recomendaciones realizadas durante el curso en cuanto a estilo y normas de programación
- El programa ya creado en un fichero con el disco duro del ordenador y posteriormente se grabará en un disco con el nombre y DNI del alumno
- Se creará en el disco un directorio con el DNI del alumno

ALUMNO: DNI:

Enunciado:

2. Modificar el programa realizado por un compañero para:

   a. Usar 15 cadenas.
   b. Leer cadenas que tengan como máximo un carácter numérico.
   c. Añadir un procedimiento *escribeCad* que presenta en pantalla las cadenas.
   d. Modificar el procedimiento *anadeLong* para que inserte un espacio entre el primero y la cadena original y añada al final la longitud también separada por espacio.
   Ejemplo: "hola" -> "4 hola 4"

Fundamentos de la Programación *1ª Ed. - 2001*

Figura 5. Segundo ejercicio

En las normas de los ejercicios se especifican todas las instrucciones y recomendaciones para su realización, tales como el tiempo del que disponen para su elaboración. El primer ejercicio tiene la apariencia de un ejercicio simple y breve, y es en el segundo donde se especifica la necesidad de utilizar los resultados del primero.

En las normas también se hace hincapié en la necesidad de seguir las recomendaciones de programación establecidas en clase, entre las cuales se encuentran todas las referidas a la necesidad de calidad en la documentación que acompaña a los ficheros fuente de toda implementación.

4.2. La organización y el grupo

El grupo de laboratorio típico sobre el que se va a trabajar está formado por un número de alumnos de 20 a 25 como norma. Debido a las fechas sobre el calendario lectivo en las que se planea el ejercicio, ya están formados los equipos de realización de prácticas, con dos alumnos por equipo, lo que da un total de 12 equipos. De estos equipos, por diversos motivos, hemos encontrado que de media asisten unos 8 a la realización de las pruebas.

Suponiendo un grupo de laboratorio compuesto de 8 equipos de dos alumnos y la disposición típica de los laboratorios en la Escuela Politécnica de la UA, los distribuimos como se muestra en la Figura 6. Dado que hay dos filas con ordenadores, los componentes de cada equipo se situarán de forma que queden en la misma columna, es decir, uno detrás de otro. Una vez colocados según esta disposición, resulta muy sencillo el intercambio de ejercicios entre ellos, ya que basta con que intercambien las posiciones en las que se encuentran sentados.

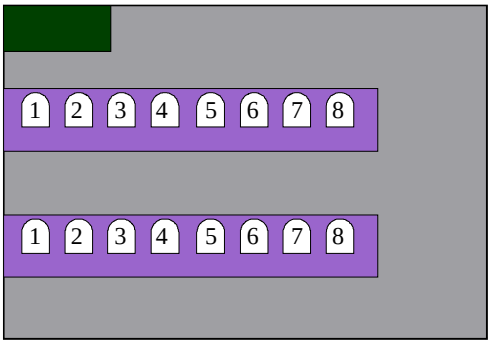


Figura 6. Disposición de los equipos en el laboratorio

4.2. El tiempo para los ejercicios

El tiempo para cada uno de estos ejercicios va a ser función del tiempo total disponible, que en nuestro caso es de 2 horas.

En primer lugar debemos tener en cuenta los 5 minutos necesarios para organizar a los alumnos según la disposición mostrada en la figura 3, que es necesaria para facilitar los pasos posteriores.

También tenemos que añadir 5 minutos más para cada vez que tengamos que recoger los resultados de los alumnos en los discos (a efectos de realizar su análisis posteriormente), lo cual nos da un total de otros 10 minutos.

Además habrá que contar con 10 minutos adicionales para realizar el cambio de puestos según está establecido y para que los alumnos reaccionen ante la novedosa situación planteada.

Estos tiempos suman un total de 25 minutos, y aunque puedan parecer a primera vista excesivos para un grupo tan pequeño, no lo son, puesto que siempre surgen todo tipo de pequeños problemas con un disco que falla, un alumno que no acaba de comprender las normas, etc.

El tiempo neto que nos resta es de 1 hora y 35 minutos para realizar el resto de las tareas. Como ya planteamos antes, es necesario más tiempo para el segundo enunciado. En concreto, se planteó tiempos de 30 y 50 minutos. Si sumamos ambos tiempos, vemos que el resultado, 1 hora y 20 minutos, nos deja aún 15 minutos de *colchón* para enfrentarnos a posibles problemas adicionales.

4.3. Los enunciados

Los enunciados (figuras 7 y 8) que se han utilizado plantean ejercicios sencillos utilizando el lenguaje Pascal, que es el lenguaje usado en los laboratorios de programación en el primer año de estudios de Informática en la UA.

1. Realizar un programa en Pascal que contenga:
   a. 1 función llamada ***lee1cad*** que pide al usuario cadenas hasta que este introduce una sin caracteres numéricos y esta cadena es devuelta como resultado de la función.
   b. 1 Procedimiento llamado ***aniadeLong*** que recibe 1 vector de 10 cadenas y las modifica añadiendo a cada una de ellas en la izquierda su longitud.
      Ej. “hola” -> “4hola”

   El programa principal usará ***lee1cad*** para llenar un vector de 10 cadenas que luego pasa a ***aniadeLong***.

Figura 7. Enunciado del primer ejercicio

2. Modificar el programa realizado por un compañero para:
   c. Usar 15 cadenas.
   d. Leer cadenas que tengan como mínimo un carácter numérico.
   e. Añadir un procedimiento ***esccad*** que presenta en pantalla las cadenas.
   f. Modificar el procedimiento ***aniadeLong*** para que inserte un espacio entre el número y la cadena original y añada al final la longitud también separada por espacio.
      Ejemplo: “hola” -> “4 hola 4”

Figura 8. Enunciado del segundo ejercicio

Como se puede apreciar en los enunciados, los ejercicios a resolver son muy simples, no pudiendo ser de otra forma debido al tiempo limitado existente para la realización de la actividad.

4.4. Tendencias observadas

Como se esperaba, los alumnos se encontraron en el segundo ejercicio con una ardua tarea, debido fundamentalmente a que:

- En ningún caso el primer ejercicio tenía comentarios, ni tan siquiera uno, lo cual es significativo puesto que éste es un asunto que casi desde el primer día recibe una atención muy especial en las asignaturas de programación en su conjunto.
- Tampoco había ni una sola línea de documentación en papel, ni siquiera unos garabatos con las intenciones de implementación.

En cambio sí había aspectos positivos como que:

- Se pudo observar un correcto sangrado del código y unos programas bien estructurados, debido en parte a lo detallado del enunciado en cuanto a la descomposición en subprogramas.
- Se utilizaron líneas de separación entre partes diferentes del programa que aportaban claridad al código.
- Se usaron nombres significativos para los identificadores de los diversos elementos del programa, destacando un caso extremo con un identificador de más de veinte letras.
- Las definiciones de los tipos de datos se realizaron correctamente.
- No se utilizaron variables globales, tal y como se había indicado durante las clases.

5. Conclusión

Aunque los resultados obtenidos no son suficientes para realizar valoraciones estadísticas, apoyan la necesidad de utilizar prácticas que enseñen y fomenten el trabajo en equipo entre los alumnos. Esto supone un cambio en la orientación de las prácticas tradicionalmente propuestas, que debe ir acompañado de la enseñanza de los métodos y técnicas básicas del trabajo en equipo.

Es destacable que los alumnos demostraron gran interés ante esta iniciativa una vez realizado el ejercicio y se interesaron por la forma de mejorar sus resultados en pruebas similares. Esto nos plantea la necesidad de preparar un conjunto de recomendaciones de acciones correctivas que se suministraría al alumno una vez finalizada la prueba. En estas recomendaciones se enumerarían los problemas que se habrá encontrado en la prueba y las posibles acciones para mejorar sus resultados. Dichas recomendaciones, que son ignoradas frecuentemente por el alumno cuando se exponen de una forma teórica, serán atendidas y aprendidas cuando el alumno se ha enfrentado a experiencias como las aquí presentadas, ya que ha podido comprobar por sí mismo la necesidad real de su utilización.

Además, hemos hallado un beneficio adicional no previsto de estas actividades, y es que permite a los alumnos enfrentarse a un ejercicio similar (aunque más sencillo) y en unas condiciones parecidas a las que se encontrará en el examen real de la asignatura. Esto hace que el alumno pueda autoevaluar sus conocimientos y tomar, si lo creyera necesario, las medidas que estime convenientes para poder afrontar con garantías de éxito el examen real.

6. Futuras líneas de trabajo

Dado lo positivo de los resultados obtenidos, se tiene previsto preparar los ejercicios de colaboración restantes, de forma que se puedan alcanzar todas las metas aquí presentadas.

Se desea también extender estos ejercicios de colaboración a todos los grupos de primero en los dos cuatrimestres y realizar valoraciones estadísticas, para lo cuál se necesitará apoyo de varios profesores de otras asignaturas. Con estas valoraciones, se podría profundizar más en el estudio y planear acciones de mayor alcance.

Si los resultados fueran satisfactorios, se realizarían también, mediante trabajo en equipo, sistemas software completos. Éstos serían realizados por alumnos de segundo o tercer curso, una vez que dichos alumnos hayan adquirido y perfeccionado las destrezas y conocimientos necesarios para este tipo de trabajo.

Referencias

[1] Kris Bosworth. *Developing Collaborative Skills in College Students*. New Directions for Teaching and Learning, no. 59, Jossey-Bass, 1994.

[2] Fotografía de la Biblioteca de la Escuela Politécnica de la UA: http://www2.uah.es /escuelapolitecnica/Album /foto29.htm.

[3] Fotografía de un laboratorio de la Escuela Politécnica de la UA: http://www2.uah.es /escuelapolitecnica/Album/foto26.htm.

[4] Fotografía de un corredor de la Escuela Politécnica de la UA: http://www2.uah.es /escuelapolitecnica/Album /foto12.htm.

[5] Jon Katzenbach y Douglas Smith. *The wisdom of teams*. Harvard Busines School Press. Boston, 1993.

[6] LOU. BOE núm. 309, de 26 de diciembre de 2001, págs. 49400-49425.

[7] Tabla de asignaturas de Ingeniería Técnica Informática de la UA: http://www2.uah.es/escuelapolitecnica /Titulaciones/pdf/TablaInfGestion.pdf

Un modelo para aplicación Sistemática de Aprendizaje Cooperativo

Antoni Pérez-Poch, Ferran Virgós Bel
Dept. Lenguajes y Sistemas Informáticos
Universidad Politécnica de Cataluña
EUETIB
C. Urgell 187 ; 08036 Barcelona
E-mail: Antoni.Perez-Poch@upc.es ; Ferran.Virgos@upc.es

Resumen

La evolución del mundo de la educación no se orienta sólo a los nuevos medios. El "e-learning" es un elemento importante pero no el único. El paso de la enseñanza al aprendizaje debe cimentarse, sobre todo, en innovación metodológica. Uno de los conceptos más relevantes al respecto es, sin duda, el "aprendizaje cooperativo".

Pero no basta con decirlo, ¡hay que saber aplicarlo!, ... y no es fácil. En el presente trabajo se parte de los fundamentos teóricos del concepto para proponer un modelo que cree luz en la aplicación sistemática de la metodología del Aprendizaje Cooperativo (AC).

Se describe la aplicación de este mismo modelo durante el cuatrimestre de Primavera 2001 en dos asignaturas, relacionadas con las TIC, impartidas en la EUETIB: Redes de Ordenadores (Optativa, Ingeniería Técnica en Electrónica Industrial) y Fundamentos de Informática (Troncal, Ingeniería Técnica Mecánica), adaptando en cada caso la variante de aplicación a las particularidades de la asignatura

En términos generales se valora positivamente su efecto motivador en los alumnos y se ha estudiado la influencia que ha tenido en la mejora de las calificaciones, así como en los resultados de la encuesta de calidad de la docencia que se pasa regularmente a los alumnos de la Escuela.

La realimentación obtenida permite la validación de algunas propuestas concretas y mejoras introducidas en el modelo. Entre ellas, la relativa a la implicación de un asesor externo en todos los equipos y el nombramiento de un portavoz en cada uno de ellos, encargado de la interacción con él, fijando, además, unas reglas mínimas para valorar la utilización de este recurso.

1. Introducción

El mundo de la educación está de moda. La rápida evolución del ciclo de vida del aprendizaje que nos lleva al concepto de *"life-long learning*" no es extraña a esta situación. Pero es más que eso: las empresas y los países se dan cuenta que su competitividad y, por tanto, su futuro, depende en gran medida de su capital humano. Además, la elevada competitividad de todos los mercados y, en particular, la propia maduración y globalización de la enseñanza, que va llegando, aunque sea lentamente, nos impulsa a buscar nuevos caminos.

La evolución del mundo de la educación mira a la virtualización y la semi- presencialidad. El "e-learning" es un elemento relevante, pero no es de fácil implantación generalizada en un plazo corto. Muchas universidades intentan incorporarlo sin contar con medios suficientes y sin entrar a considerar seriamente la necesidad de formación e incentivación del profesorado. Lo más probable para éstas es el más estrepitoso fracaso.

Otro importante aspecto a considerar es que, en la actualidad, una demanda frecuente de las

empresas es que sus directivos y empleados técnicos sean capaces de desenvolverse con éxito en equipos multidisciplinares e incluso multiculturales. La Universidad debe responder a esta demanda con la enseñanza de habilidades sociales y cooperativas para sus alumnos y no sólo con la mera transmisión de conocimientos técnicos.

En definitiva, el Aprendizaje Cooperativo se muestra como un extraordinario compañero metodológico para la mejora del proceso educativo. Pero no basta con decirlo, hay que aplicarlo. La cuestión es ¿cómo?, o ¿por dónde empezar?.

En consecuencia, nuestro objetivo en el trabajo se centraba en partir de los fundamentos teóricos del concepto para proponer un modelo que creara luz (práctica) en la aplicación sistemática de la metodología del Aprendizaje Cooperativo (AC) en el aula (sea física , virtual o mixta).

Subsidiáriamente, por motivos interesados, deseábamos aplicarlo y validarlo en la enseñanza universitaria de la informática.

2. Fundamentación teórica

El Aprendizaje Cooperativo (AC) es una técnica metodológica que pone el centro del trabajo que comporta un aprendizaje en el propio alumno [1]. Hasta ahora ha sido utilizada principalmente en niveles pre-universitarios, aunque cada vez se están recogiendo más experiencias en el nivel universitario [2]. En la Universitat Politécnica de Catalunya existe un Grupo de Interés en AC , con una importante presencia en la Escuela de Ingeniería Técnica Industrial de Barcelona, EUETIB [3].

(I) Grupos formales e informales

En AC se distinguen tres tipologías de grupo: grupos informales, que duran el tiempo de una clase, grupos formales, que existen por un plazo de tiempo superior al de una hora lectiva y que tienen como objetivo realizar alguna tarea determinada, y finalmente grupos de base, con los que los alumnos avanzan progresivamente en sus estudios.

En nuestros trabajos hablaremos de grupos formales cuya composición puede o no ser decidida por el propio profesor de la asignatura, siendo ésta una decisión que debe determinarse antes del inicio de la actividad.

(II) Condiciones para el verdadero AC

Hay que resaltar que con una simple propuesta de trabajo en grupo no se produce un verdadero AC. Por el contrario deben existir ciertos elementos imprescindibles para que el aprendizaje entre iguales exista: a) interdependencia positiva, b) interacción directa cara a cara, c) responsabilidad personal d) habilidades grupales, y e) proceso de desarrollo del grupo.

a) *Interdependencia positiva*

Empecemos describiendo la interdependencia positiva. Esta se da cuando cada individuo del grupo es consciente de que el éxito de cada uno depende ineludiblemente del éxito de los demás. Se trata de que todos asuman que no hay ninguna posibilidad de alcanzar el éxito si no se implican todos los componentes del grupo. Si se hunde uno, se hundiran todos. Para ello las tareas deberán diseñarse de forma que todos comprendan que si uno no cumple con su parte, todos recibirán las consecuencias negativas. La búsqueda del éxito por parte de todos los componentes del grupo es fundamental.

b) Interacción directa

La interacción (cara a cara, o no), es otro elemento esencial en AC. La interacción constante y la compartición de ideas y recursos es una herramienta fundamental para el éxito. Para ello hay que crear las condiciones físicas para favorecerlo. De todos es conocido, por ejemplo, que las mesas de reuniones circulares favorecen la interacción entre los interlocutores.

c) Responsabilidad personal

La responsabilidad individual, la implicación de cada miembro del grupo mediante su actitud y su tarea , debe llevar a la responsabilidad de grupo, es decir a que éste consiga todos sus objetivos. Con que sólo un miembro del grupo no asuma su propia implicación hará fracasar el objetivo global.

d) Habilidades grupales

El cuarto elemento, el desarrollo de habilidades sociales y cooperativas es uno de los objetivos del AC a la vez que un elemento indispensable para su éxito. Cada vez los proyectos técnicos se llevan a cabo por equipos multiculturales y multidisciplinares, con lo que este tipo de habilidades son fundamentales para el futuro éxito profesional del alumno de enseñanzas técnicas.

e) Proceso de desarrollo del grupo

Finalmente, un elemento esencial para que los alumnos puedan evaluar su progreso es la reflexión sobre su trabajo en grupo. Los alumnos que forman parte del grupo han de saber qué acciones y tareas son útiles para poder alcanzar sus objetivos y cuáles no, con el objeto de determinar qué acciones deben mantenerse y cuáles hay que potenciar. Este tipo de decisiones es común a la gestión de cualquier tipo de proyecto, pero aquí el énfasis se hace en el autocontrol por parte del propio grupo de trabajo.

Así, se pueden incluir en el proceso del aprendizaje cooperativo algunas técnicas de control de calidad: cuestionarios de autoevaluación, establecimiento de fechas límite con entregas parciales, cuestionarios de valoración del trabajo realizado, reuniones de autoevaluación periódicas, etcétera [4].

(III) Rol del profesor como motivador

El rol de profesor pasa de mero transmisor de conocimientos en una clase magistral, y por tanto, protagonista absoluto de la clase, a coordinador. El papel del profesor pasa a un segundo plano y es el alumno, o mejor dicho el grupo de alumnos el que asume el papel activo en el proceso de enseñanza-aprendizaje. El profesor dirige las actividades, asigna los recursos necesarios para conseguir los objetivos del proceso, observa el desarrollo de cada grupo y efectúa las correcciones necesarias para asegurar la eficacia del proceso.

Sin embargo, el profesor debe evitar la tentación de controlar de forma exhaustiva al grupo. Por el contrario, ha de efectuar su tarea de coordinación en un discreto segundo plano.

3. Modelo de aplicación

A partir de los principios teóricos de la metodología del Aprendizaje Cooperativo se propone el modelo de aplicación práctica recogido en la figura. En ella, puede observarse

- Debe decidirse con qué criterios se realizarán los grupos. El modelo puede contener un repositorio de estos criterios. En nuestro caso, para asegurar que no se formaran grupos cerrados, optamos por una agrupación aleatoria que luego podría ser corregida en caso de aparecer algún problema, pero ésto debería ser excepcional.

- Los diferentes contenidos del trabajo se deberán establecer teniendo en cuento la materia y el nivel de la asignatura. También aquí, el modelo contendría un repositorio de criterios que pueden incluir la conveniencia de favorecer la existencia de un eje de actividad común.

- También debe cuidarse la clarificación de la normativa de “desarrollo” que podría basarse, asimismo, en un repositorio.

- Para el inicio del proyecto se dedica una sesión, normalmente de prácticas para explicar qué es lo que se pretende y dar las directrices básicas. Es de un lado “tutorial” y de otra, impulso de motivación.

- Se crean los grupos y éstos empiezan a trabajar planteando los primeros objetivos y haciendo una búsqueda inicial de información. Esta parte es ideal hacerla en biblioteca y/o con conexión a Internet y/o bases de datos.

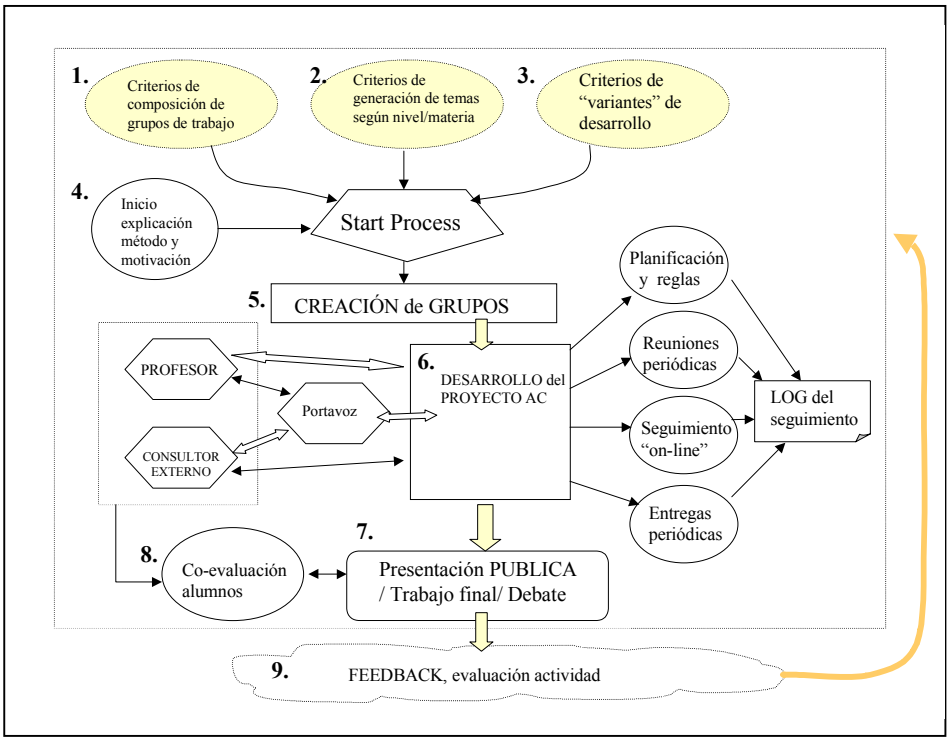


Figura 1: Modelo de aplicación de AC

- Se establecen las reglas del juego, con las normas para la interacción del grupo, vía portavoz con el asesor externo, y se establecen algunas fechas importantes para realizar entregas parciales y la presentación final.

- Se desarrolla el trabajo durante el cuatrimestre en forma semipresencial, dedicándose alguna(s) sesión(es) más de prácticas a trabajar el proyecto. Durante esta fase el profesor debe coordinar el funcionamiento de cada uno de los grupos mediante reuniones periódicas y/o de seguimiento "on-line".

- En la presentación final del trabajo se pueden establecer normas para asegurar que todos los miembros del grupo han trabajado en su realización. Por ejemplo se puede anunciar previamente que se harán preguntas aleatoriamente a uno de los miembros del grupo y que la nota será la misma para todos ellos. Esto se complementa con la autoevaluación de los miembros del grupo y la coevaluación del profesor con el asesor externo y los propios compañeros de otros grupos.

- Se puede plantear una coevaluación en la presentación, es decir, que los alumnos se evalúen entre ellos mismos , o decidir que sea el profesor el que puntúe el proyecto. También puede ser interesante entablar un debate al final de cada presentación pública.

- Es esencial, finalmente, incorporar algún método efectivo de realimentación del funcionamiento de la actividad. Una encuesta breve y específica (de orientación creativa, no descriptiva) realizada de forma anónima permite conocer en qué medida la actividad ha sido provechosa para los alumnos y cuáles pueden ser los puntos de mejora del método.

En los siguientes apartados podemos ver los resultados de la aplicación de este modelo práctico en la EUETIB en dos cuatrimestres alternos y sobre dos asignaturas del área informática pero de nivel y características bien diferenciadas.

4. Descripción de nuestra experiencia de aplicación del modelo

Una aplicación de este modelo se realizó para dos asignaturas del área de tecnologías y sistemas de información, en la EUETIB: Redes de ordenadores y Fundamentos de Informática.

AC como método de enseñanza de informática

En efecto, la Universidad debe ser capaz de favorecer las actitudes positivas ante el trabajo en equipo y las habilidades sociales requeridas para ello. Aunque no está establecido en qué materias deben favorecerse estas habilidades. En principio cualquier asignatura es válida y la informática en forma singular.

El diseño algorítmico es un tipo de problema abierto, que no tiene una única solución válida. Este tipo de problema es frecuentemente el más importante en cualquier texto de informática básica, y en particular uno de los objetivos básicos de la asignatura troncal Fundamentos de Informática en el plan de estudios de EUETIB para todas sus especialidades (Electrónica industrial, electricidad, mecánica y química industrial). Se pide además que el diseño que haga el alumno sea eficiente y elegante además de efectivo y sintácticamente correcto. Parece pues, un tipo de problema no convergente apropiado para la discusión y el trabajo en equipo.

En el campo de la Telemática y Redes de Ordenadores, hay una gran diversidad de tecnologías y protocolos de conectividad que evolucionan de forma muy rápida. Su trabajo en detalle es prácticamente imposible para una asignatura de iniciación, optativa de sólo 3 créditos. Se pretende con la actividad que profundicen en una de ellas de cara a su aplicación práctica a la implementación a un escenario real. Se hace hincapié en la viabilidad de la aplicación de esta tecnología, y se discutirá posteriormente en clase si las opiniones del grupo son asumidas por el resto de la clase. Creemos que para una titulación de ingeniería técnica esto es de mayor importancia que incidir en la planificación teórica de la red.

Aún cuando por limitaciones de tiempo y número de alumnos no se puedan realizar este tipo de experiencias del modo que sería idealmente deseable, es interesante promover este tipo de actividades por el valor añadido que suponen sobre la utilización exclusiva de métodos de enseñanza puramente tradicionales [5].

Redes de Ordenadores es una asignatura optativa de tres créditos de quinto cuatrimestre dentro de la titulación de Ingeniería Técnica en Electrónica Industrial. El número de alumnos matriculados está entre 30 y 40, formándose dos grupos para las sesiones prácticas.

Sus objetivos generales son proporcionar unos conceptos básicos en cuanto a las redes de ordenadores, sus protocolos, elementos físicos de interconexión y conectividad de área local y de banda ancha. Se hace énfasis en su aplicación a la mejora de la productividad de las empresas.

Entre las sesiones prácticas hay una parte de hardware (montaje y configuración de redes de área local), una parte de simulación, una visita guiada al punto neutro de acceso a Internet de Catalunya, CESCA-CATNIX [6] y el estudio de un sistema operativo de red.

En cuanto a la asignatura de Fundamentos de Informática su contexto es totalmente distinto a pesar de ser impartida en la misma escuela. Se trata de una asignatura de carácter troncal de primer cuatrimestre dentro de fase selectiva. Tiene un número de alumnos alrededor de 140 en el cuatrimestre de otoño y unos 50 en el cuatrimestre de primavera. En éste último se repite el cuatrimestre de fase selectiva habiendo no más de un 10 % de alumnos de nueva incorporación. Sus objetivos son ofrecer unos conocimientos básicos de informática, estructura de ordenadores, metodología de la programación y sistemas operativos. Es una asignatura de 6 créditos con 2 horas teóricas y 2 de prácticas a la semana. La parte central del curso la constituye la programación estructurada y el diseño

algorítmico. Las prácticas de programación con lenguaje Pascal son una parte esencial de la asignatura y favorecen el aprendizaje progresivo del alumno con propuestas de realización de algoritmos de dificultad creciente.

5. Metodología de trabajo

En la asignatura Redes de Ordenadores se propone una evaluación continua a lo largo del cuatrimestre formada por un examen parcial (40% de la nota), la evaluación de los informes de prácticas (30%) y un trabajo de intensificación (30%) que constituye nuestra experiencia de aprendizaje cooperativo. A los alumnos que superan satisfactoriamente esta evaluación quedan eximidos de la obligatoriedad del examen final y pueden presentarse para mejorar su calificación.

El trabajo de intensificación consiste en una investigación sobre una determinada tecnología de conectividad (cable, inalámbrica, DSL, GPRS, UTMS, etc.) y aplicarla a un escenario real propuesto por el profesor y común a todos los grupos.

Inicialmente los alumnos quedaron divididos en grupos de tres alumnos y a cada grupo se le asignó una o unas tecnologías que debían estudiar. La primera sesión de trabajo se realizó en la biblioteca de la Euetib donde los grupos se dedicaron a buscar información tanto en la bibliografía como por Internet con el apoyo del profesor. Al final de ésta, cada grupo debía entregar un informe en Word con la descripción del trabajo realizado, los objetivos de su trabajo y la repartición de tareas en el grupo.

Se entregó a cada grupo un documento con las próximas tareas a realizar. La siguiente actividad que debían realizar eran los dos primeros puntos del trabajo: introducción, objetivos y bibliografía recogida. En esta sesión se dejó un tiempo para trabajo autónoma y se entregó el escenario real que debían resolver. Este escenario es una mediana empresa con unas necesidades concretas de conectividad. Cada grupo debía estudiar si su tecnología era aplicable y en caso afirmativo, presentar un presupuesto concreto.

En las últimas sesiones de clase del cuatrimestre se hizo la presentación pública del trabajo por parte de cada uno de los grupos. El trabajo se evaluó mediante la entrega de una memoria y de la presentación en clase, con mayor peso para la presentación. Una de las transparencias debía ser un mapa conceptual de su presentación.

Al principio de la actividad se recogieron las direcciones de correo electrónico de todos los alumnos de cada grupo para que lo utilizaran a lo largo del trabajo. Además , se contactó con dos expertos externos a la UPC que accedieron a dar soporte via e-mail a los grupos que lo requiriesen. En concreto, estos expertos fueron Josep Lluís Guallar, técnico de comercio electrónico de IBM, North Carolina y Xavier Larrosa, vicepresidente de la Unió de Radioaficionats de Catalunya.

En cuanto a la asignatura de Fundamentos de Informática, hay que constatar que la mayoría de alumnos son repetidores en fase selectiva y el principal problema con que nos encontramos es su falta de motivación. Además, la asignatura dentro de la especialidad de Mecánica es terminal.

La mayor dificultad que acostumbran a tener los alumnos es aplicar sus conocimientos de programación al diseño algorítmico. En clase la limitación de tiempo hace que el número de ejercicios resueltos no sea muy elevado.

Se propuso la realización en grupos de tres de un problema de diseño algorítmico de nivel parecido al exigido en el examen final. Este ejercicio se empezó a resolver en clase dedicándose una media hora a plantear el problema con ayuda del profesor. Al cabo de una semana debían entregar el problema terminado en pseudocódigo. El profesor lo comentó con cada grupo, y en la última clase del cuatrimestre cada grupo presentó el problema en clase como ejercicio de síntesis del curso.

En esta primera experiencia, la entrega final del problema se dejó como ejercicio optativo y se valoró dentro de la evaluación continua de la asignatura (25%). Se coordinó con los profesores

de prácticas para que pudieran trabajar el ejercicio en las clases prácticas desarrollándolo también en Pascal.

En los trabajos propuestos en el siguiente curso académico 2001-02 se realizaron algunas modificaciones, como hacer obligatorio el trabajo de Fundamentos de Informática, nombrar un portavoz para cada grupo y extender la interacción con un profesional externo a todos los grupos.

Se anunció a los alumnos que la calificación se realizaría en parte con preguntas sobre el trabajo a un miembro del grupo que sería asignado aleatoriamente. Sin embargo, la nota sería la misma para todos los componentes del grupo. Esto les obliga por mutuo acuerdo a intentar que nadie del grupo se quede fuera del papel activo que debe desempeñar dentro del propio grupo.

Se puede incluir a criterio del profesor una autoevaluación y/o co-evaluación del trabajo que tenga un peso significativo en la calificación final del trabajo cooperativo.

Asimismo como continuación de la experiencia se recogen en este curso también los resultados de la experiencia activa de aprendizaje.

6. Resultados

En ambas experiencias se ha valorado la efectividad de estos trabajos con una encuesta anónima, específica para ello y la evaluación de las notas finales.

En la asignatura de Redes de Ordenadores respondieron la encuesta el 92% de los alumnos que realizaron el trabajo cooperativo. La encuesta estaba formada por 11 items que valoran diversos aspectos del trabajo siendo la media de respuestas de 3.6 sobre 5 (5 máxima valoración, 1 mínima).

Los aspectos mejor valorados fueron el hecho de hacer el trabajo en grupo y no individualmente (4.0) y especialmente la experiencia de interactuar via e-mail con un experto externo a la Universidad (4,7) así como la utilidad de esta ayuda (4,4). Los aspectos peor valorados fueron la utilidad de la primera entrega parcial del trabajo y el uso de la biblioteca, aún así con un 2,8 sobre 5.

El 100% de los alumnos que siguieron la evaluación continuada han aprobado la asignatura, siendo significativo que la realización del trabajo ayudó a subir la nota del examen parcial. En contraposición, sólo 1 de los 4 que no realizaron la evaluación contínua en la que se incluye el trabajo cooperativo aprobaron el examen final y por tanto la asignatura. El porcentaje de aprobados en este cuatrimestre fue superior al porcentaje del pasado año en el que se realizó también un trabajo pero sin la metodología cooperativa.

También se preguntó por correo electrónico a los dos técnicos que participaron en la experiencia su opinión. Ambos manifestaron estar muy satisfechos con la experiencia, y se muestran dispuestos a continuarla y si es posible a ampliarla por ejemplo, con el uso de videoconferencia.

En cuanto a la experiencia en la asignatura de Fundamentos de Informática el número de alumnos que realizaron el trabajo hasta el final fue reducido (8 personas sobre un total de 20 que realizaron la primera sesión) Nos proponemos para próximos cuatrimestres plantear este ejercicio como obligatorio en las clases teóricas dado que los resultados han sido positivos para la mayoría de los que lo han realizado.

La encuesta de evaluación de este trabajo se planteó también a partir de una serie de ítems, diferentes para los que realizaron el ejercicio de síntesis hasta el final de los que no lo realizaron.

Entre los que sí lo realizaron recibimos 7 encuestas (88%) obteniendo una valoración media de la actividad de 3.65 sobre 5. Los aspectos mejor valorados fueron la coordinación con las prácticas (4.3), el que el trabajo se realizara cooperativamente (4,1) y el hecho de que el hecho de presentarlo en clase obligaba a prepararselo mejor (4.0).

Significativamente, tres alumnos manifestaron que de no haber sido por esta

actividad no habrían hecho ningún ejercicio de este tipo hasta el mismo día del examen final.

A los que no realizaron el ejercicio hasta el final se les preguntó por qué razón. De los resultados de las encuestas contestadas (8) se deduce que no es la falta de motivación la razón principal para no acabar y presentar el ejercicio final sino la falta de tiempo dado que no era una actividad obligatoria.

En cuanto a los resultados finales, el 75% de los alumnos que además de realizar la evaluación contínua de la asignatura realizaron este ejercicio superaron la asignatura, porcentaje que se reduce al 53% para el total de alumnos de la asignatura.

En la exposición final de esta experiencia se anotarán los resultados recogidos al final del cuatrimestre de primavera 2001-02.

7. Conclusiones

Se ha realizado una experiencia de aprendizaje cooperativo en el área informática de la EUETIB en el cuatrimestre de Primavera del curso 2000/01 con resultados positivos. Esta experiencia tiene continuación en siguientes cuatrimestres y se realiza también en diversas asignaturas de otras áreas de conocimiento de EUETIB como Automática y Experimentación Química.

La experiencia se ha realizado en una asignatura optativa de la titulación de Ingeniería Técnica en Electrónica Industrial y en una troncal de Ingeniería Técnica en Mecánica valorándola con una encuesta específica y el análisis de resultados de las asignaturas.

En la asignatura optativa de Redes de Ordenadores nos planteamos a la vista de los resultados, ampliar la experiencia en próximos cuatrimestres. Para ello, proponemos dar un mayor peso en la nota final a este trabajo y contactar con otras personas del mundo empresarial y de Radioaficionados para ofrecer soporte interactivo a los grupos de trabajo.

Creemos que el hecho de especificar por escrito al máximo lo que se pide de cada grupo, y los plazos de entrega ha ayudado a que la experiencia llegara a buen puerto. Posiblemente, la primera entrega parcial debería formar parte del trabajo definitivo para centrar más inicialmente los objetivos del trabajo.

También es importante que el profesor esté disponible de forma continua a lo largo de la realización del trabajo y que se enseñe a hacer mapas conceptuales como modelo de esquema del trabajo, ya que muchos alumnos no conocían en qué consistía un mapa de este tipo.

En la asignatura troncal de Fundamentos de Informática, se da el hecho de que hay una mayor masificación de alumnado en el cuatrimestre de otoño por lo que la aplicación de la metodología cooperativa puede ser más problemática. En el cuatrimestre de primavera parece ser una actividad que puede ser motivadora en especial para suplir las carencias de falta de práctica de los alumnos en el diseño algorítmico. Nos planteamos también darle un mayor peso en la nota de evaluación continua de la asignatura así como presentarla como obligatoria. Especialmente interesente es el que esta actividad esté coordinada con las clases prácticas.

El feed-back al final del cuatrimestre indica los puntos fuertes del desarrollo de la actividad y favorece la mejora de la actividad en cuatrimestres sucesivos.

Referencias

[1] Rué i Domigo, Joan (1991) 'El trabajo cooperativo: La organización de la enseñanza y el aprendizaje. *Barcanova Educación.*

[2] Jornadas de Aprendizaje Cooperativo. JAC-01 Instituto de Ciencias de Educación, Universitat Politècnica de Catalunya. Julio 2001.

[3] http://www-ice.upc.es apartado GIAC

[4] Johnson, David W. Johnson, Roger T. Smith, Karl A. (1991): "Active Learning: Cooperation in the College Classroom". *Interaction Book Company.*

[5] L.Villardón, Univ. Deusto. 'Una experiencia de aprendizaje cooperativo en la Universidad. Organización y reflexión'. JAC-01. ICE Univ. Politécnica de Catalunya. Julio 2001.

[6] http://www.catnix.es ; http://www.cesca.es

Integración del aprendizaje individual y del colaborativo en un sistema hipermedia adaptativo

Carlos Arteaga, Ramón Fabregat
Institut d'Informàtica i Aplicacions (IIiA)
Universitat de Girona (UdG)
e-mail: carteaga@eia.udg.es, ramon.fabregat@udg.es

Resumen

En este artículo se define un Sistema Hipermedia de Aprendizaje Colaborativo Adaptativo (SHACA) en el que se relaciona el Aprendizaje Individual y el Aprendizaje Colaborativo a través de un Modelo Adaptativo. El Modelo Adaptativo permite al Ambiente de Aprendizaje adaptar su comportamiento al estudiante tanto cuando interactúa sólo, como cuando está involucrado en tareas colaborativas. La arquitectura del sistema propuesto se basa en la arquitectura definida para los Sistemas Hipermedia Adaptativos (SHA) agregándole el Modelo de la Colaboración en el que se definen las reglas para el comportamiento adaptativo durante el Aprendizaje Colaborativo. La información y las reglas contenidas en estos modelos son la base para establecer el comportamiento adaptativo del ambiente de aprendizaje integrado.

1. Introducción

En el mundo actual, la tecnología empieza a jugar un papel importante en los procesos de aprendizaje. Esto nos obliga a reflexionar sobre los elementos involucrados en su uso y a buscar nuevas formas de enseñar y de aprender eficientemente.

La tecnología informática y de comunicaciones puede cambiar radicalmente la forma de relacionarnos y conseguir información, pero en los sistemas educativos esto no es suficiente. Tenemos que ser capaces no sólo de transmitir información sino también de lograr la asimilación efectiva de conocimiento. Por esto debemos contar con mecanismos de medición y evaluación de los resultados y ser capaces de soportar la demanda creciente sin perder calidad.

Hay varios trabajos de investigación relacionados con el impacto de la tecnología y los nuevos problemas que pueden surgir de su uso en los procesos de aprendizaje [25]. Pero para entender el impacto y los problemas asociados es importante distinguir entre *"el efecto de"* la tecnología y *"el efecto con"* la tecnología. De acuerdo a la perspectiva que se tome para analizar el problema, los resultados pueden cambiar. Se entiende por *"efectos de la tecnología"* a: qué se ha aprendido y que puede transferirse de aquéllas situaciones en las que se trabaja con la computadora" y por "efectos con la tecnología" a: qué puede lograr uno en sinergia con una computadora [18].

En [2] se analiza una serie de flaquezas asociadas al uso del Internet en los cursos a distancia, y a porque muchas de las expectativas se ven frustradas. Algunas de estas flaquezas no están directamente relacionadas con la tecnología sino con el diseño del material instruccional debido al desarrollo de estrategias de aprendizaje inadecuadas más que a los problemas de Internet por sí mismo.

En [25] se comenta que hay que ser escépticos a cualquier predicción en el campo de las tecnologías de la información y se resalta la necesidad de ser cautos a la hora de aplicar la tecnología. Sin embargo también considera que estas nuevas herramientas facilitarán la adquisición de las habilidades de aprendizaje, la elaboración de estrategias propias de aprendizaje, el reconocimiento de los estilos de aprendizaje típicos, la selección del recurso de aprendizaje apropiado, así como el refuerzo del valor de las habilidades de reflexión. Se puede prever que los recursos digitales serán más interactivos, adaptables y amistosos, promoviendo la creatividad del estudiante y la integración de la experiencia.

La propuesta que exhibimos plantea el uso de la tecnología para integrar el aprendizaje

individual y el aprendizaje colaborativo a través de un modelo adaptativo.

Actualmente se han desarrollado muchos ambientes de aprendizaje basados en tecnologías de la información y de comunicaciones. Unos hacen más énfasis en el aprendizaje individual [25] y [8], otros en el aprendizaje colaborativo [13] y existen algunos que integran ambos aspectos [23], [21] y [11].

La colaboración adaptativa es un campo de investigación reciente [6] y su tecnología se desarrolló junto con los sistemas educativos conectados a una red de computadoras. La meta del apoyo a la colaboración adaptativa es conformar grupos de colaboración, sin la intervención directa del estudiante, basados en el Modelo de la Colaboración y el Modelo del Estudiante.

Teniendo en cuenta la importancia de estudiar el tema del aprendizaje en un marco colaborativo y la importancia del aprendizaje individual como un elemento indivisible de la generación y asimilación de conocimiento, nos planteamos las siguientes preguntas que son base para la definición de la propuesta de este trabajo:

- ¿Mejorarán el proceso de aprendizaje Los Sistemas Hipermedia Adaptativos que incorporen características Colaborativas Adaptativas?
- ¿Qué impacto tiene en el aprendizaje colaborativo la utilización de técnicas adaptativas?
- ¿Cuál es la mejor técnica para proveer de adaptación a un ambiente colaborativo?
- ¿Cómo cambian las reglas de colaboración y cuál es su impacto en el Aprendizaje Individual?
- ¿El efecto es distinto de acuerdo a la población objetivo?

Para poder dar respuesta a todas ellas proponemos un modelo para el desarrollo de ambientes integrados de Aprendizaje Individual y Aprendizaje Colaborativo con soporte para el comportamiento adaptativo.

La arquitectura del sistema propuesto, se basa en la arquitectura definida para los Sistemas Hipermedia Adaptativos (SHA) [3] y [26], que tiene como componentes el Modelo del Estudiante (ME), el Modelo del Dominio (MD) y el Modelo del Profesor (MP) o modelo pedagógico. Esta arquitectura es extendida agregándole un nuevo modelo el Modelo de la Colaboración (MC) en el que se definen las reglas para el comportamiento adaptativo durante el Aprendizaje Colaborativo.

La adaptatividad en el aprendizaje colaborativo afecta principalmente a la conformación de los grupos y al rol que los estudiantes pueden jugar en cada uno de los grupos.

La diferencia entre nuestra propuesta y los trabajos que integran el aprendizaje individual y el colaborativo esta en la forma de efectuar la integración y lograr el comportamiento adaptativo. Por ejemplo, los trabajos mencionados no utilizan un Sistema Hipermedia Adaptativo (SHA) para presentar el material instruccional y ni los avances del estudiante en un entorno de aprendizaje (colaborativo o individual) afectan directamente la adaptatividad del otro ambiente.

En los apartados siguientes describiremos brevemente los conceptos de la adaptatividad y la naturaleza de los modelos de aprendizaje que son base para el SHACA. Posteriormente se presentará el modelo para el Sistema Hipermedia de Aprendizaje Colaborativo Adaptativo (SHACA). Finalmente se enumeran las conclusiones y se mencionan algunos trabajos futuros.

2. Sistemas Hipermedia Adaptativos

Brusilovsky [3] define un Sistema Hipermedia Adaptativo como aquel que construye para cada usuario un modelo de objetivos, preferencias y conocimientos. Utiliza este modelo a través de la interacción para adaptarse a las necesidades del usuario.

Oppermann [22] tiene en cuenta las características del usuario y distingue entre:

- *Sistemas Adaptables*: permiten al usuario cambiar ciertos parámetros del sistema y adaptar de esta manera su comportamiento.
- *Sistemas Adaptativos*: Se adaptan al usuario automáticamente basándose en las suposiciones que el sistema realiza de las necesidades del usuario.

Según la formalización realizada en [3] y [26] los Sistemas Hipermedia Adaptativos utilizan tres

componentes básicos para lograr la adaptatividad: el Modelo del Usuario, el Modelo del Dominio de la aplicación y el Modelo de la adaptación (en el caso educativo es el Modelo del Profesor o Modelo Pedagógico). Esta división proporciona claridad al desarrollo de aplicaciones adaptativas y permite asignar responsabilidades específicas a cada modelo.

- El Modelo del Estudiante describe la información del usuario que un SHA se guarda en un registro permanente. Esta información incluye una representación del estado del conocimiento adquirido por el estudiante y un registro de los nodos que ha visitado.
- El Modelo del Dominio describe cómo se enlaza y estructura la información.
- El Modelo del Profesor está compuesto por reglas pedagógicas que definen cómo se combinan el Modelo del Dominio y el Modelo del Estudiante para proveer de adaptación al sistema.

Añadiendo a estos tres componentes mencionados se puede efectuar dos tipo de adaptación: Presentación y Navegación Adaptativa [4], [5], [20] y [16]. Diferentes investigaciones [7], [15] y [24] han demostrado que esta adaptación puede tener un efecto positivo en el proceso de aprendizaje. Sin embargo en [24] se comenta que algunas técnicas adaptativas descritas en [4] pueden causar confusión y problemas en el proceso de aprendizaje.

La Presentación Adaptativa tiene que ver con la forma en la que una página de información es presentada y se realiza mediante la adaptación o generación dinámica de las páginas de información. La Navegación Adaptativa tiene que ver con la manera de navegar en el hiper-espacio de información definido y se realiza mediante la adaptación de los enlaces a la página siguiente para guiar al estudiante de forma particular.

En ambos casos existen técnicas y métodos que permiten el desarrollo de sistemas con características adaptativas. Las técnicas definen como realizar la adaptación al nivel de implementación. Los métodos describen a nivel conceptual cómo debe realizarse la adaptación y se consideran cuatro aspectos: ¿Qué adaptar? (tecnologías para la adaptación), ¿Por qué? (objetivos de la adaptación), ¿Dónde? (áreas de aplicación) y ¿A qué? (características del usuario).

Los métodos empleados por la Presentación Adaptativa son la explicación adicional, la explicación basada en pre-requisitos, la explicación comparativa, variantes de explicación y ordenamiento según la importancia.

Los métodes empleados por la Navegación Adaptativa son la guía directa, el ordenamiento adaptativo de enlaces, el ocultamiento adaptativo de enlaces, la anotación adaptativa y los mapas adaptativos.

Es muy importante resaltar que la adaptatividad representa un cambio cuali-cuantitativo en los sistemas de aprendizaje. El estudio de su aplicación se ha orientado a lograr eficiencia y eficacia en los procesos de aprendizaje.

3. Modelos de Aprendizaje

En este apartado revisaremos brevemente el Aprendizaje Individual, y el Aprendizaje Colaborativo [2], [12] y [25].

El Aprendizaje Individual está orientado a satisfacer necesidades del estudiante que pueden variar en el tiempo, la forma, el contenido y el volumen. Esto determina la necesidad de que los ambientes desarrollados para apoyar el Aprendizaje Individual sean flexibles, amigables y tengan incorporado los conceptos de adaptación. La valoración que un estudiante particular tendrá de un sistema está determinado por la habilidad del sistema para facilitarle su aprendizaje [25].

En las teorías de aprendizaje queda claro que la interacción es un factor catalizador del proceso de aprendizaje [17]. Pero la creación de conocimiento y la asimilación del mismo es siempre un proceso individual [12] y [19]. Por lo tanto es importante que los sistemas de educación a distancia proporcionen un soporte adecuado para este tipo de aprendizaje.

La necesidad de adaptación en un sistema hipermedia de soporte al aprendizaje individual surge por

Debido a la confusión ocasionada en el estudiante (principalmente en aquel que es novicio en un tema) por el gran volumen de información que puede ser presentado es necesario que los sistemas hipermedia de soporte al aprendizaje

individual incorporen los tipos de adaptación presentados en el apartado anterior.

El Aprendizaje Colaborativo está orientado a la generación de conocimiento y lo podemos definir como: "Co construcción de conocimiento, y mutuo compromiso de los participantes". En términos genéricos, se define colaboración, como: cualquier actividad que un par o más de individuos efectúa juntos [18].

El Aprendizaje Colaborativo intenta eliminar los problemas de aislamiento y de soledad que tienen los estudiantes al interactuar con ambientes de aprendizaje individual sin la presencia del docente y/o grupos de estudio. La colaboración es un catalizador de conocimientos y muchos de los avances están orientados a la socialización del proceso de aprendizaje [1], [9] y [10].

Varias de las metodologías y herramientas que usan la tecnología (hardware y software) para apoyar el Aprendizaje Colaborativo han surgido bajo el nombre de "Aprendizaje Colaborativo Soportado por Computador" (CSCL-Computer Supported Colaborative Learnig) [13]. Pero en este campo no sólo se ha investigado la tecnología sino también los aspectos sociales, psicológicos, organizacionales y los efectos en el aprendizaje. Por los resultados de las investigaciones realizadas se puede deducir que el sólo hecho de utilizar tecnología no soluciona los problemas asociados al proceso de aprendizaje. Es necesario incluir otros elementos: objetivos pedagógicos, seguimiento, análisis y medición de resultados [13].

Hay que tener claros los objetivos que se persiguen con el aprendizaje colaborativo pues pueden generar diferencias importantes en el momento de implementar soluciones y buscar resultados. Cuestiones como ¿Qué es colaboración? y ¿Qué estudiamos cuando aplicamos colaboración apoyada en la tecnología? son elementos a considerar al momento de aplicar la colaboración [18].

Según [14] el proceso de aprendizaje donde intervienen grupos de estudiantes tiene dos secuencias: Secuencias de Transferencia de Conocimiento y Secuencias de Colaboración. En la primera es donde el instructor transfiere conocimiento a los estudiantes y en la segunda es donde se sintetiza y aplica el conocimiento recibido.

4. Sistema Hipermedia de Aprendizaje Colaborativo Adaptativo (SHACA)

El SHACA es un sistema que integra el aprendizaje individual con el aprendizaje colaborativo, y en el que ambos aprendizajes son adaptativos. Una característica importante es que los modelos utilizados para la adaptación en el sistema de aprendizaje individual son los mismos que se utilizarán para conseguir la adaptación durante el aprendizaje colaborativo. Otro aspecto relevante es que cualquier cambio en el modelo del estudiante afecta el comportamiento adaptativo tanto del ambiente de aprendizaje individual como el del ambiente de aprendizaje colaborativo.

Como ya se ha comentado en la introducción, la arquitectura propuesta extiende la arquitectura de los Sistemas Hipermedia Adaptativos ([3] y [26]) incorporando el Modelo de la Colaboración en el que se definen las reglas para el proceso de selección y formación de los grupos de colaboración.

La arquitectura del SHACA está formada por los siguientes módulos (figura 1):

- **Diseño y Creación de Material Instruccional.** Herramienta de autor que permite al docente crear el material instruccional. Incluye soporte para crear el material basado en los estilos de aprendizaje definidos en [14].
- **Interfaz de usuario.** Módulo que permite la interacción entre el estudiante y el Ambiente de aprendizaje integrado.
- **Aprendizaje Individual.** Módulo para la presentación, estudio y exploración del material instruccional. Este mecanismo es un Sistema Hipermedia Adaptativo, que incorpora Presentación Adaptativa y Navegación Adaptativa.
- **Diagnóstico.** Sirve para determinar el estado inicial del conocimiento del estudiante sobre el dominio de la aplicación. , y preferencias del estudiante y el o los estilos de aprendizaje de su preferencia. La información obtenida es utilizada para aplicar las reglas de adaptación del ambiente de aprendizaje individual y colaborativo.

- **Aprendizaje Colaborativo.** Este componente incorpora el concepto de Colaboradores Potenciales y el de Espacio de Colaboración. Los Colaboradores Potenciales son todos los estudiantes que participan en el proceso de aprendizaje y el Espacio de Colaboración son subconjuntos de los Colaboradores Potenciales formados de acuerdo a las reglas del modelo de adaptación que se definan. Incluye herramientas de comunicación, intercambio de ideas y otros. (E-mail, Chat, Pizarras electrónicas, Noticias, Foros, y otros).
- **Motor Adaptativo (Engine).** Ejecuta las reglas de adaptación para todo el ambiente de aprendizaje utiliza los cuatro modelo considerados: Modelo del Dominio, Modelo del Estudiante, Modelo del Profesor y el Modelo de la Colaboración. Es una extensión al mecanismo de adaptación empleado en los Sistemas Hipermedia Adaptativos. A través de este mecanismo cualquier cambio en el Modelo del Estudiante se refleja tanto en el ambiente de aprendizaje individual como en el ambiente de aprendizaje colaborativo.
- **Evaluación, Monitoreo y Seguimiento.** Recolecta información del Ambiente de Aprendizaje Individual como del Colaborativo, procesa la información recolectada y emite indicadores de la actividad del estudiante en el sistema. Los resultados obtenidos ayudan a afinar el proceso de aprendizaje

Un proceso típico de utilización del sistema SHACA sería el siguiente:

- Diseño y creación del material instruccional.
- Definición del Modelo del Profesor, del Modelo del Dominio y del Modelo de la Colaboración.
- Publicación del material a través del ambiente de aprendizaje individual (que fundamentalmente es un Sistema Hipermedia Adaptativo).
- Diagnóstico para el estado inicial del estudiante y su estilo de aprendizaje.
- El estudiante inicia su aprendizaje tanto individual como colaborativamente. Este proceso es integrado sin línea divisoria entre ellos.

El avance del estudiante en el estudio del material instruccional afecta la actividad en el Ambiente de Aprendizaje Colaborativo, y a su vez el avance en el Aprendizaje Colaborativo afecta al ambiente de aprendizaje individual. Por ejemplo los roles que juega el estudiante pueden cambiar, así como la presentación y navegación en el ambiente de aprendizaje individual.

La definición del Modelo de la Colaboración, no es trivial y de hecho es uno de los componentes más complejos y críticos para el éxito de esta propuesta. Las reglas sobre las cuales los Espacios de Colaboración se forman deben tener en cuenta los problemas que se asocian a la interacción en los grupos y al estilo de aprendizaje propio de cada estudiante [14]. Deben permitir que los participantes realmente se encuentren en ambientes donde se construya conocimiento colaborativamente y no meramente en un ambiente para la difusión superficial de conocimiento o intercambio de opiniones personales [18]. Este es el reto principal que tiene la implementación de esta propuesta.

El Modelo de la Colaboración incluye las reglas para la conformación de los Espacios de Colaboración y está orientado a resolver cuestiones como:

- ¿Qué condiciones debe cumplir un estudiante para pertenecer a un grupo particular?
- ¿Las características de cada grupo serán previamente definidas o serán dinámicas?
- ¿Los grupos permitirán el uso de roles de los estudiantes?
- ¿El cumplimiento de los objetivos del grupo cambia el estado del Modelo del Estudiante?
- ¿Cuáles son las condiciones para que un estudiante salga de un grupo e ingrese en otro?

5. Conclusiones

En este trabajo se presenta una arquitectura básica para desarrollar sistemas de aprendizaje que integren el Aprendizaje Individual y el Aprendizaje Colaborativo. Esta integración se hace de manera dinámica y de forma transparente al estudiante. Se definen los conceptos para lograr comportamiento adaptativo en la conformación y manejo de grupos de aprendizaje colaborativo.

Creemos que no sólo se deben desarrollar nuevas aplicaciones para soportar el proceso de aprendizaje, sino también se debe investigar el impacto que ellas tienen en el mismo. Hay mucho que investigar sobre la naturaleza de los sistemas adaptativos y el impacto de ellos en los estudiantes.

Consideramos que el SHACA (como un ambiente de aprendizaje integrado y adaptativo) puede ser un instrumento muy útil para el desarrollo de cursos a distancia.

Las cualidades adaptativas que incorpora se convierten en un componente clave para una entrega eficiente de material instruccional y para un desarrollo más dinámico de las actividades de aprendizaje tanto individual como colaborativo.

Finalmente el combinar ambientes de aprendizaje individual con ambientes de aprendizaje colaborativo y dotar a ambos de características adaptativas puede incrementar la eficiencia y eficacia durante el proceso de aprendizaje. Esta afirmación deberá confirmarse con el desarrollo de experimentos que den más luces sobre su validez.

Como continuación de este trabajo se creará un ambiente para un dominio específico y se pondrá a prueba para analizar su funcionamiento. Es necesario estudiar las poblaciones objetivo y a partir de ello definir nuevas estrategias y/o cambios en la filosofía del enseñar y aprender. Los paradigmas para la creación de material de estudio, la forma de monitorear el avance y el contacto con el estudiante son aspectos que también deben ser considerados.

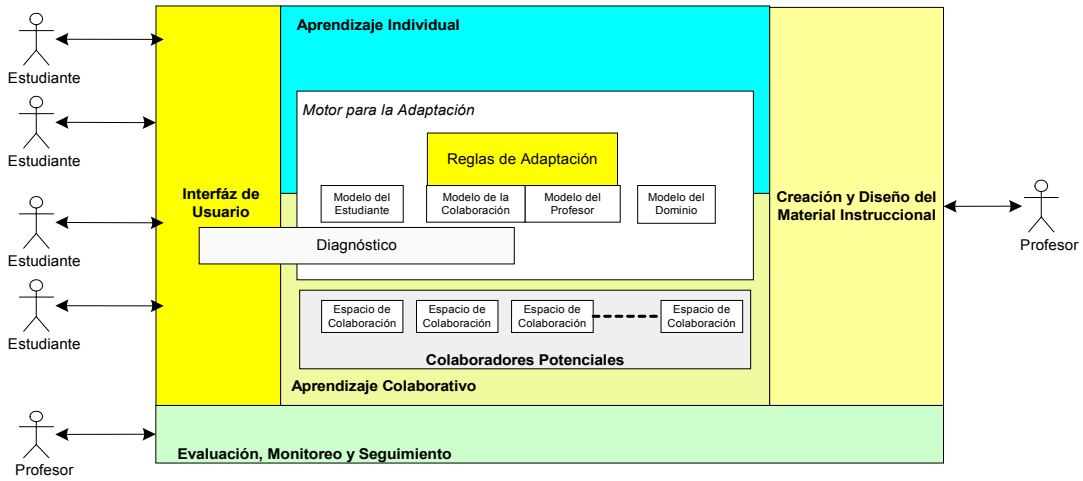


Figura 1: Arquitectura del SHACA

Referencias

[1] L. Anido, M. Caeiro, M. Llamas, M.J. Fernández.: "*Creating collaborative environments for web-based training scenarios"*. SIIE2000, 2do Simposio Internacional de Informática Educativa. 15-17 de Noviembre de 2000 Puertollano (Ciudad Real)

[2] A. Bork: *"What is needed for effective learning on the Internet"*. Educational Technology & Society 4(3)2001 SIN 1436-4522.

[3] P. De Bra, G.J. Houben, H. Wu: "*AHAM: A Dexter-based Reference Model for Adaptive Hypermedia"*.Proceedings of the ACM Conference on Hypertext and Hypermedia, pp. 147-156, Darmstadt, Germany, 1999. (Editors K. Tochtermann, J. Westbomke, U.K. Wiil, J. Leggett)

[4] P. Brusilovsky: *Methods and techniques of adaptive hypermedia.* User Modeling and User Adapted Interaction. v 6, n 2-3, pp 87-129. (Special issue on adaptive hypertext and hypermedia). 1996

[5] P. Brusilovsky: "*Adaptive Navigation Support: A Component for Information Exploration Interfaces"*. Proceedings, CHI 98, April 18-23, 1998, Los Angeles, CA USA

[6] P. Brusilovsky: *"Adaptive Educational Systems on the World-Wide-Web: A Review of Available Technologies"*. In: Proceedings of Workshop "WWW-Based Tutoring" at 4th International Conference on Intelligent Tutoring Systems (ITS'98). 1998. San Antonio, Texas.

[7] P. Brusilovsky, J. Eklundb J., E. Schwarzc: "*Web-based education for all: a tool for development adaptive courseware"*. Seventh International World-Wide Web Conference, April 14-18, 1998, Brisbane, Australia

[8] P. Brusilovsky: *"Adaptive hypermedia"*. User Modeling and User Adapted Interaction, Ten Year. Anniversary Issue, (Alfred Kobsa, ed.) 11 (1/2), 87-110. 2001

[9] A. J. Cañas, K. M. Ford, J. W. Coffey, T. Reichherzer, N. Suri, R. Carff, D. Shamma, G. Hill, M. Breedy.: "*Herramientas para Construir y Compartir Modelos de Conocimiento Basados en Mapas Conceptuales"*. Revista de Informática Educativa, Vol. 13, No. 2 (2000), pp. 145-158.

[10] A. J. Cañas, K. M. Ford, P. H. Hayes, T. Reichherzer, N. Suri, J. W. Coffey, R. Carff, G. Hill,: "*Colaboracion en la Construccion de Conocimiento Mediante Mapas Conceptuales"* Invited Plenary Talk, VIII Congreso Internacional sobre Tecnología y Educación a Distancia, San José, Costa Rica, (Nov. 1997). Available in the Proceedings of the Conference, pp. XXV-XLII.

[11] M. Constantino-Gonzalez, D. Suthers: *A Coached Collaborative Learning Environment for Entity-Relationship Modeling In Intelligent Tutoring Systems*, Proceedings of the 5th International Conference (ITS 2000).

[12] J. Ewing, D. Miller: "*A framework for evaluating computer supported collaborative learning"*. Educational Technology & Society 5(1) 2002. ISSN 1436-4522

[13] S. Gerry (Editor 2002) *"Proceedings of CSCL 2002, Boulder, Colorado, USA"*. January 7 - 11, 2002, distributed by Lawrence Erlbaum Associates, Inc. Hillsdale, New Jersey, USA © 2002

[14] C. R. Haller, V. J. Gallagher, L. T. Weldon, R. M. Felder*,: "Dynamics of peer education in cooperative learning workgroups*." J. Engr. Education, 89(3), 285-293 (2000).

[15] K. Hook*: "Evaluating the Utility and Usability of an Adaptive Hypermedia System"*. in Journal of Knowledge-Based Systems, vol. 10, no. 5, 1998.

[16] A. Jameson*:"User-Adaptive Systems: An Integrative Overview"*.UM99, the Seventh International Conference on User Modeling, Banff, June 1999; and at IJCAI99, the Sixteenth International Joint Conference on Artificial Intelligence, August 1999.

[17] G. Kearsley: *"Explorations in learning & instruction: the theory into practice database"*. Copyright 1994-2001 Greg Kearsley (gkearsley@sprynet.com). http://home.sprynet.com/~gkearsley.

[18] L. Lipponen*: "Exploring foundations for computer-supported collaborative learning"*. Proceedings of CSCL 2002, Boulder, Colorado, USA. January 7 - 11, 2002 Edited by Gerry Stahl, distributed by Lawrence Erlbaum Associates, Inc.

[19] J. A. Macías y P. Castells. *"Diseño interactivo de cursos adaptativos"*. SIIE2000, 2do Simposio Internacional de Informática Educativa. 15-17 de Noviembre de 2000 Puertollano (Ciudad Real)

[20] J.A. Macías; P. Castells: *"Un sistema de presentación dinámica hipermedia para representaciones personalizadas del conocimiento"*. II Congreso Internacional de Interacción Persona-Ordenador (Interacción 2001). Salamanca (España), 2001.

[21] C. J. M, Olguín, A. L. N. Delgado, I. L. M. Ricarte*: "An Agent infrastructure to set collaborative environments"*. Educational Technology & Society 3(3) 2000. ISSN 1436-4522

[22] R. Oppermann, R. Rashev, Kinshuk: "*Adaptability and Adaptivity in Learning Systems"*. Knowledge Transfer (volume II) (Ed. A. Behrooz), pAce, London, pp173-179. 1997. (ISBN 1-900427-015-X).

[23] A. Rachida, M. Amine Benkiran, M. Ato*.: "A Framework for Adaptative and Cooperative Learning for the Internet; SMART Learning"*. INET 2000 Proceedings. Japan. 18-21 July 2000

[24] P. Da Silva, R. Van Durm, E. Duval, H. Olivié.: "*A Simple Model for Adaptive Courseware Navigation"*. Conferentie Informatiewetenschap 1997. Thema: "Let your Browser do the Walking" Technische Universiteit Eindhoven. 27 november 1997

[25] K. Sinitsa. *"Learning Individually: a Life-Long Perspective Introduction to the Special Issue"*. Educational Technology & Society 3(1)2000. SIN 1436-4522

[26] H. Wu (Position Paper*): "A Reference Architecture for Adaptive Hypermedia Systems"*. Third Workshop on Adaptive Hypertext and Hypermedia. Hypertext'01 . Århus, Denmark, August 14-18, 2001

Arquitectura de ordenadores

Un computador didáctico elemental (CODE-2)

A.Prieto F.J. Pelayo F.Gómez Mula J.Ortega

A.Cañas A.Martínez F.J.Fernández

Departamento de Arquitectura y Tecnología de Computadores
E.T.S.I. Informática
Universidad de Granada
E-18071 Granada. e-mail: aprieto@ugr.es

Resumen

Se describe en los niveles de lenguaje máquina, micromáquina y usuario un computador didáctico elemental (CODE-2) que reúne las características deseables para la enseñanza de las asignaturas de introducción a los computadores. Este computador ha sido diseñado completamente, con unidades de control cableada y microprogramada. También se dispone de un entorno completo de simulación que incluye emulador, ensamblador, desensamblador y visualización del comportamiento dinámico de todos los elementos internos del computador.

1. Introducción

La arquitectura de computadores es una disciplina típica de ingeniería y para preparar material didáctico para ella se debe realizar un laborioso trabajo de generalización de las diversas técnicas utilizadas en computadores concretos, y no limitarse a recopilar información detallada sobre ellos. Este material debe presentar al alumno abstracciones de equipos reales, de forma que le capaciten no sólo a entender los computadores actuales sino también los futuros, cuando éstos vean la luz. Este concepto es especialmente relevante en un ámbito tan cambiante y en expansión como el de los computadores. Esta idea es de gran importancia en los cursos de iniciación, donde hay que lograr que el estudiante fije su atención en las cuestiones esenciales, y no se pierda en los detalles, la mayoría de ellos fugaces, de las máquinas reales.

Dentro de este contexto la mayoría de libros de texto que tratan de la estructura o arquitectura de computadores suelen presentar máquinas ficticias que muestran cruda y claramente los conceptos más básicos y generales; sin perjuicio de que el conocimiento de éstos posteriormente se reafirme y particularice considerando casos de máquinas concretas y reales. Chu [1] fue uno de los primeros autores que describió con detalle la estructura y diseño de un computador didáctico; con posterioridad Gorsline [2] presenta el "G-1", Karam y Bryant [3] el "CUSP" *(Carlenton's Utterly Simple Processor),* Mano [4,5] el "Basic Computer" y posteriormente [6] otros dos prototipos (uno RISC y el otro CISC), Foster [7] el "Blue" (nombre coincidente con el color de su chasis), Knuth [8] el "MIX", Lee [9] el "SDC" *(Simple Didactic Computer),* Tanenbaum [10] el "Mac-1" *(Macroarchitecture-1),* Hennessy y Patterson [11] una máquina RISC genérica, el "DLX" (*DeLuXe*), Carpinelli [12] el *Relatively Simple Microprocessor* y Patt [13] el "LC-2" *(Little Computer-2).*

Otros autores utilizan para apoyar sus explicaciones subconjuntos o modelos simplificados de máquinas reales. Así, Tannebaum [14] presenta el "IJVM" *(Integer Java Virtual Machine),* que contiene el subconjunto de instrucciones para manejar enteros de la JVM, Murdocca [15] el "ARC" *(A RISC Computer)* que es un subconjunto del SPARC, y Patterson y Hennesy [16] utilizan como modelo didáctico una versión simplificada del MIPS R2000/R3000.

Los autores españoles de textos de introducción a la estructura de computadores también suelen considerar máquinas ideales. Este es el caso del procesador descrito por de Miguel [17], el "ODE" (Ordenador Didáctico Elemental) de Prieto, Lloris, Pelayo y Gómez-Mula [18-21], la "MS" (Máquina Sencilla) de Ayguadé, Navarro y Valero-García [22], y la Máquina Rudimentaria de Hermida, del Corral, Pastor y Sánchez [23,24], también descrita por Ruz [25]. Gregorio Fernández, con una gran sentido pedagógico, presenta su curso de arquitectura y sistemas operativos [26] partiendo de una máquina muy sencilla "Simplez(+i4)", que va ampliando en máquinas virtuales sucesivas conforme va introduciendo conceptos más complejos: "Algorítmez", "Regístrez", "Monoalgorítmez" y "Multialgorítmez".

Ante esta pléyade de máquinas didácticas, Fermín Sánchez presentó en la edición anterior de estas jornadas [27] un interesante trabajo en el que definió las características deseables en un procesador pedagógico para la enseñanza básica de arquitectura de computadores. En él, aparte de describir las características indicadas, se comparaban tres de los procesadores citados anteriormente: el MS, el DLX y la Máquina Rudimentaria. Estas tres máquinas se acercaban en mayor o menor grado al modelo preconizado en [27], pero ninguna de ellas lo seguía plenamente.

En el presente trabajo presentamos un Computador Didáctico Elemental (CODE-2) que consideramos satisface plenamente las especificaciones y prestaciones propuestas en [27]. De este computador en la actualidad se dispone del siguiente material:

- Definición de la estructura accesible por el programador y del lenguaje máquina.
- Definición de un lenguaje ensamblador
- Implementación de un emulador para un entorno MS-Windows
- Implementación de un ensamblador cruzado, para un entorno MS-Windows.
- Implementación de herramientas integradas para trabajar con CODE-2 (contiene además del ensamblador y emulador, depurador, desensamblador, visualización del funcionamiento dinámico de cada una de los elementos del procesador, etc.)
- Diseño completo de CODE-2 con unidad de control cableda.
- Diseño completo de CODE-2 con unidad de control microprogramada.
- Diseño de CODE en VHDL
- Construcción de un entrenador CODE-2 con circuitos lógicos programables (FPGA).

El resto del trabajo se estructura de la siguiente forma: la Sección 2 presenta las características generales de CODE-2 cotejándolas con las indicadas en [27] para un procesador pedagógico ideal; la Sección 3 presenta someramente el repertorio de instrucciones máquina. El computador CODE-2, para ser utilizado por los alumnos como entrenador, se ha montado en un bastidor y su panel frontal contiene visualizadores de los principales elementos internos, en la forma que se describe en la Sección 4. La Sección 5 considera las pautas seguidas para el diseño de la máquina; por último, la Sección 6 presenta las conclusiones.

2. Características de CODE-2

Obviamente un procesador didáctico ideado para introducir la estructura y el funcionamiento de un computador debe tener los elementos indispensables para que el alumno adquiera los conocimientos que se pretenden. En un curso básico estos conocimientos se centran en conceptos tales como: memoria principal, puertos de entrada/salida (E/S), unidad de procesamiento de datos, unidad de control, buses, etc., así como los relacionados con la estructura de dichos elementos a nivel de micromáquina (RTL): contador de programa (PC), puntero de pila (SP), banco de registros (RF), unidad aritmético-lógica, biestables indicadores de la ALU, etc. También deben introducirse los conceptos básicos inherentes al funcionamiento dinámico del procesador: fases y estados en la ejecución de una instrucción, activación de los biestables indicadores, circulación de la información por los buses, sincronización, etc.

En la Figura 1 se muestra un esquema de los elementos de CODE-2 a los que se tiene acceso desde el lenguaje máquina; es decir, esta figura presenta los elementos visibles al programador. Las principales características de CODE-2 son:

1. Su arquitectura es sencilla y ortogonal. En efecto, dispone de tan sólo 16 instrucciones

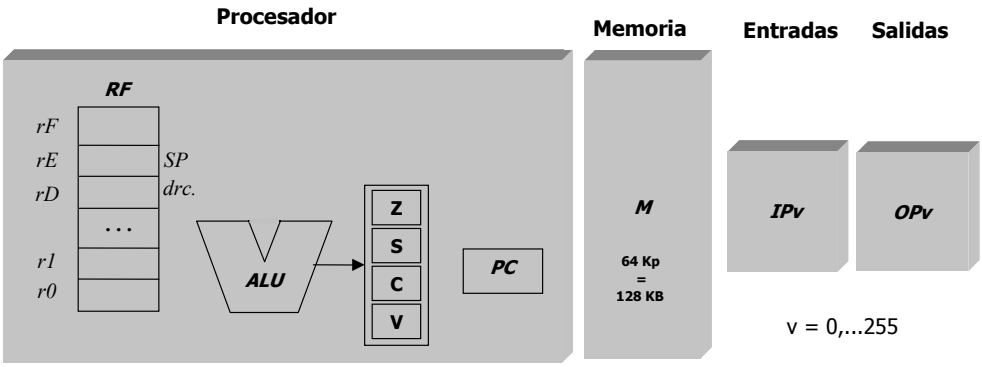


Figura 1. Elementos de CODE-2 accesibles al programador

máquina, coincide el tamaño de los datos y las instrucciones (16 bits) con lo que se evitan problemas de alineamiento y se facilita el diseño del procesador, la memoria es de 128 Kbytes haciéndose su direccionamiento por palabras y no por bytes. La unidad de procesamiento de datos dispone de un banco de 16 registros (*RF*) de 16 bits. El formato de las instrucciones es completamente ortogonal (Figura 2), estando ubicado el código de operación (*codop*) siempre en los cuatro primeros bits. En la Figura 2, *rx, rs* y *ra* representan registros de RF, *cnd* la condición de salto, y *v*, un valor inmediato de 8 bits.

2. También sigue la tendencia RISC, en cuanto es una arquitectura de carga-almacenamiento. En efecto, todas las operaciones de la ALU se hacen entre registros de RF, cuyos contenidos pueden cargarse o memorizarse con dos instrucciones específicas, una de carga desde memoria (*LD*) y otra de almacenamiento en memoria (*ST*), respectivamente. Hay cuatro espacios de direcciones (Figura 1): direcciones de registros (16), direcciones de memoria (64 Kp), direcciones de puertos de entrada (256), y direcciones de puertos de salida (256). El registro *rE* del banco de registros (Figura 1) actúa como puntero de pila (*SP*), y, al igual que el contador de programa (*PC*), es de 16 bits. El registro *rD* se utiliza para almacenar direcciones (direccionamiento indirecto).
3. La ALU realiza operaciones muy sencillas: suma y resta de enteros en complemento a 2, operación lógica NAND, y desplazamientos a derecha e izquierda. Se dispone de cuatro biestables indicadores: cero (*Z*), signo (*S*), acarreo (*C*), y desbordamiento (*V*).

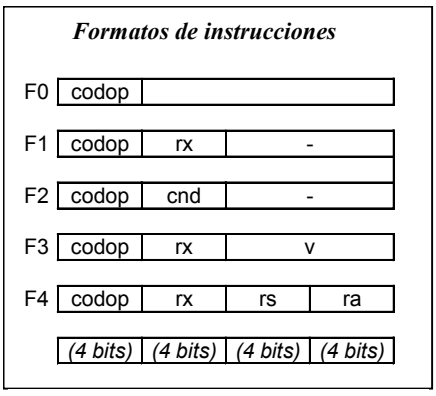


Figura 2: Formatos de las instrucciones de CODE-2

Especial cuidado se ha tenido en la planificación de los direccionamientos a memoria, que intervienen sólo en las instrucciones de carga (*LD*) y de almacenamiento (*ST*). En efecto, los direccionamientos se realizan sumando el contenido del registro *rD* y el campo *v* de la instrucción; siendo, por tanto, la dirección efectiva *d=rD+v*. De esta forma se puede tener, como casos particulares: direccionamiento directo (si *rD=H'0000*), direccionamiento indirecto a través de registro (si *v=H'00*), direccionamiento indexado (si *rD* hace las funciones de índice) y direccionamiento relativo a base (si *rD* hace las funciones de base y *v* las de desplazamiento).

En consecuencia consideramos que CODE-2 reúne todos los requisitos establecidos en [27] para un procesador pedagógico.

3. Repertorio de instrucciones máquina de CODE-2

El repertorio de instrucciones máquina de CODE-2 se muestra en la Tabla 1. Los nemotécnicos que se utilizan son los propuestos en el estándar IEEE 694 [28]. Aparte de las instrucciones de carga y almacenamiento (comentadas en la sección anterior) hay dos instrucciones de carga inmediata de registros, que utilizan el formato F3 (Figura 2): una de ellas (*LLI*) carga la parte baja del registro especificado en la instrucción y otra (*LHI*) la parte alta del registro. También se han incluido instrucciones específicas de entrada y salida, lo cual no impide que el profesor explique la posibilidad de realizar entradas y salidas utilizando direcciones del mapa de memoria. Las instrucciones de salto y de llamada a subrutina utilizan el formato F2, de forma que el segundo campo de 4 bits especifica si la bifurcación es incondicional *(cnd=0000)* o, en caso contrario la condición de salto (especificada con otros valores de *cnd*). La dirección de salto hay que almacenarla previamente en el registro *rD* del banco de registros.

Debido a la ortogonalidad de CODE-2, que divide todas las instrucciones en campos de 4 bits (1 cifra hexadecimal) es inmediato pasar de nemónicos a código hexadecimal; en efecto, la primera cifra hexadecimal identifica el *codop*, la segunda uno de los 16 registros de RF (0, 1,..., E,F) o la condición de salto, y los otros dos campos dos registros o un valor inmediato (*v*), en hexadecimal

Cuando se realiza un programa en lenguaje máquina para CODE-2, se recomienda al alumno cargar en los registros r0 y r1 los valores 0 y 1, respectivamente, con dos instrucciones *LLI*. Una vez almacenados estos valores, resulta inmediato llevar el contenido de un registro a otro (con la instrucción de suma, sumando el contenido de r0 al registro origen, y llevando el resultado al registro destino), o incrementar o decrementar el contenido de un registro (sumándole o restándole el contenido de r1, respectivamente).

Además del lenguaje máquina, se ha definido un lenguaje ensamblador, siguiendo el estándar IEEE694 [28] tanto para la definición de los nemónicos (que coinciden con los de la cuarta columna de la Tabla 1) como de las directivas que se han implementado (*ORG, EQU, DW, DR* e *INCLUDE*).

4. Montaje de CODE-2

El CODE-2, para ser utilizado por los alumnos como entrenador, se encuentra montado en un pequeño bastidor cuyo panel frontal se muestra en la Figura 2, y que contiene los siguientes elementos:

Tabla 1. Repertorio de instrucciones de CODE-2

Codop		**Nombre**	**Nemó-nico**	**Pára-metros**	**For.**	**Explicación**
binario	*Hex*					
0000	0	Cargar	LD	rx,[v]	F3	rx←M(rD+v)
0001	1	Almacenar	ST	[v],rx	F3	M(rD+v)←rx
0010	2	Carga inme.baja	LLI	rx,v	F3	rx(15:8)←H'00; rx(7:0)←v
0011	3	Carga inme. alta	LHI	rx,v	F3	rx(15:8)←v
0100	4	Entrada	IN	rx,IPv	F3	rx←IPv
0101	5	Salida	OUT	OPv,rx	F3	OPv←rx
0110	6	Suma	ADDS	rx,rs,ra	F4	rx←rs+ra
0111	7	Resta	SUBS	rx,rs,ra	F4	rx←rs-ra
1000	8	NAND	NAND	rx,rs,ra	F4	rx←(rs·ra)'
1001	9	Desplaza izquierda	SHL	rx	F1	C←rx(15), rx(i)←rx(i-1), i=15,...,1; rx(0)←0
1010	A	Desplaza derecha	SHR	rx	F1	C←rx(0), rx(i)←rx(i+1), i=0,...,14; rx(15)←0
1011	B	Desplaza arit. dch.	SHRA	rx	F1	C←rx(0), rx(i)←rx(i+1), i=0,...,14
1100	C	Salto	B-	cnd	F2	Si cnd se cumple, PC←rD
1101	D	Subrutina	CALL-	cnd	F2	Si cnd se cumple, rE←rE-1, M(rE)←PC, PC←rD
1110	E	Retorno	RET	-	F0	PC← M(rE); rE←rE+1
1111	F	Parar	HALT	-	F0	Parar

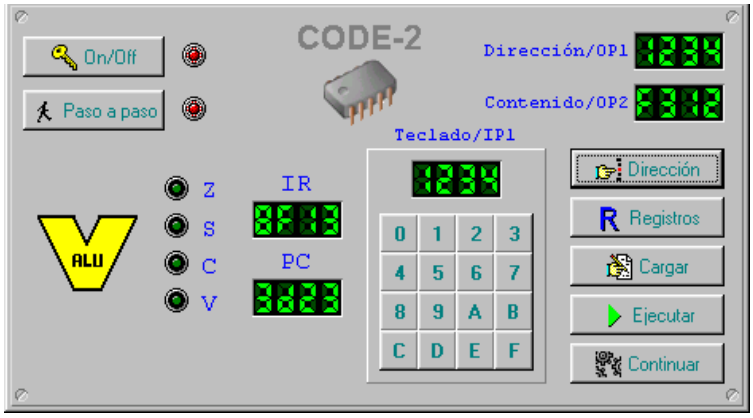


Figura 3. Panel de control de CODE-2

- *Interruptores* ON/OFF, para funcionar en Modo Paso a Paso, del que se puede salir con el pulsador CONTINUAR.
- *Teclado hexadecimal*, para introducción de programas y datos.
- *Puertos de salida 1 (OP1) y 2 (OP2).* La información se visualiza en hexadecimal.
- *Teclas de órdenes*: Dirección, Registros, Cargar, Ejecutar, y Continuar.
- *Pilotos* indicando el valor de los biestables indicadores de la ALU (*Z, S, C* y *V*) y visualizadores de los contenidos de los registros *IR* y *PC*.

Una vez activado el interruptor ON/OFF, bajo el control de las teclas de órdenes, pueden realizarse las siguientes tareas:

- *Seleccionar una posición de memoria previamente tecleada.* Con lo que en *OP1* aparece la dirección y en *OP2* su contenido.
- *Cargar un valor en una posición de memoria.*
- *Seleccionar los registros.*
- *Cargar un valor en un registro.*
- *Ejecutar un programa*.

Para gestionar el funcionamiento de las teclas de órdenes, la memoria de CODE-2 contiene un pequeño monitor, que se ubica en la zona ROM de la memoria principal.

5. Estructura y diseño de CODE-2

La Figura 4 muestra la estructura del procesador de CODE-2, cuyo funcionamiento se determina por medio de 29 señales de control. Se han diseñado dos unidades de control, una cableada y otra microprogramada.

El diseño de la unidad de control cableada se realiza partiendo de un diagrama de flujo del repertorio de instrucciones. Por otra parte realizamos una tabla con una lista de todas las microoperaciones (40) que hay que generar, y que se muestra parcialmente en la Tabla 2. A partir de las columnas segunda y tercera de la tabla resulta inmediato obtener las funciones de conmutación correspondientes a cada señal de control. Por ejemplo, para implementar la señal de control *b* se suman todos los términos que aparecen en la columna tercera correspondientes a las filas donde aparezca *b=1* en la segunda columna. A título de ejemplo, algunas de las 29 funciones de conmutación obtenidas se muestran en la Tabla 3.

La unidad de control se implementa según el modelo que se incluye en la parte derecha de la Figura 4. Un reloj actúa sobre un contador de 4 bits ("estado", en la figura) que va generando los distintos ciclos de cada instrucción.

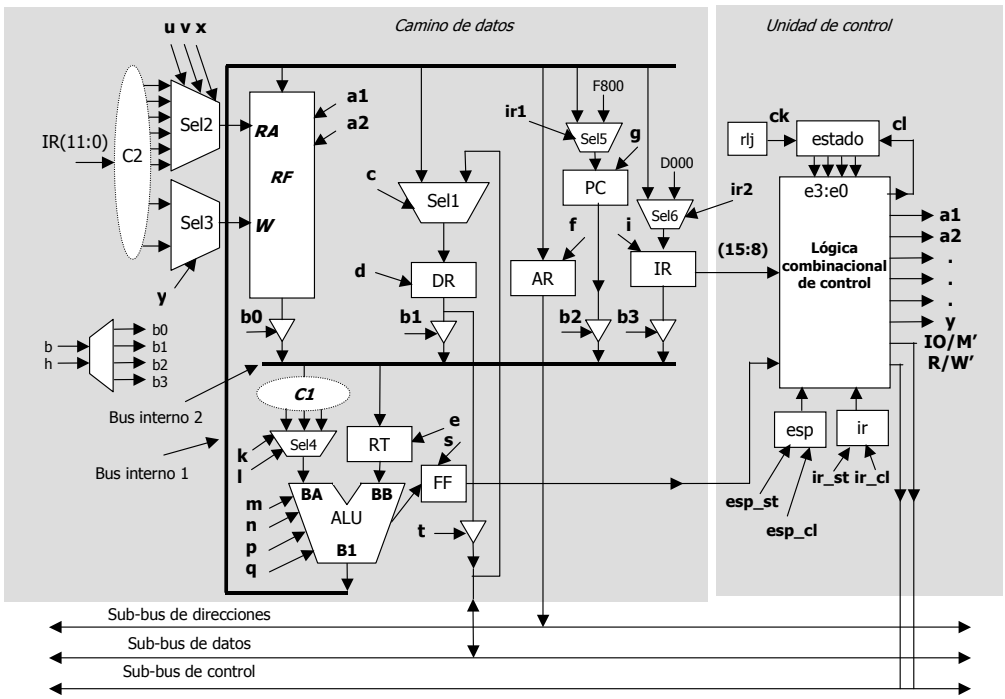


Figura 4. Esquema completo del procesador de CODE-2

En el ciclo en que finaliza una instrucción se hace *cl=1*, con lo que el contador pasa a generar otra secuencia de ciclos: c0, c1, c2, ... La Figura 5 muestra un esquema de la lógica combinacional de control, que contiene un decodificador de instrucciones, un decodificador de estado y un selector de condición, que, en las instrucciones de bifurcación, proporciona a su salida (FFc) un valor que es 1 si efectivamente hay que generar el salto. Con ayuda de este esquema, y la Tabla 2 es inmediato sintetizar las distintas señales de control. En la Figura 5, a título de ejemplo, se muestran los esquemas de las señales *k, c, R/W'* e *y*.

Más fácil de diseñar y describir resulta la unidad de control microprogramada, que sigue un esquema completamente clásico, y cuyos detalles pueden verse en [29].

7. Conclusiones

Un computador es un sistema de gran complejidad y cuya estructura y funcionamiento requiere el conocimiento de un gran número de conceptos. Debido a ello la mayoría de profesores de esta materia optan por utilizar máquinas ficticias donde explicar con claridad los conceptos más generales y básicos que se aplican a cualquier máquina real. El problema fundamental a la hora de seleccionar una máquina ideal estriba en buscar un compromiso adecuado entre cualidades didácticas (presentar sólo cuestiones esenciales) y complejidad (entrar en excesivos detalles).

En el presente trabajo proponemos un Computador Didáctico Elemental (CODE-2), que consideramos (Sección 2) reúne todas las características deseables en un procesador pedagógico para la enseñanza básica de la arquitectura de computadores [27]. Detalles adicionales sobre CODE-2 se encuentran en [29], y en la dirección web

http://atc.ugr.es/intro_info_mcgraw.html

se incluye un entorno didáctico completo (emulador, ensamblador, desensamblador, etc.), así como su diseño VHDL. En la actualidad se

Tabla 2. Lista parcial de las microoperaciones a implementar

Microoperaciones	Señales a generar	Situación en que debe generarse la microoperación
AR←H'00##IR(7:0)	b=h=1,k=0,l=1,m=n=p=q=0,f=1	c5·I4+c5·I5
AR←00'H##IR(7:0) +RT	b=h=1,k=m=0, n=p=1, q=0,f=1	c6·I0+c6·I1
AR←alu←PC	b=1,h=k=j=m=n=p=q=0, f=1	c1·ir'
AR←RF	a1=1,b=h=0, k=0,I=0,f=1	c6·ID·FFc+c5·IE
DR← alu ←PC	b=1,h=0, m=n=p=q=0, c=0, d=1	c7·ID·FFc+c7·IE
.....	.....	.....
RF←RT+1	m=p=q=0, n=1, a2=1	c6·IE
RT←PC	b=1,h=0,e=1	c2·ir'
RT←RF	a1=1,b=h=0,e=1	c5·(I0+I1+I6+I7+I8+IE)+c6·I3
WA←E'H	y=1	c5·ID·FFc+c6·IE
WA←rx	y=0	c5·(I2+I3+I9+IA+IB)+c6(I6+I7+I8) +c7·(I3+I4)+ c8·I0

esta procediendo al montaje de varios prototipos usando circuitos lógicos programables (FPGA).

CODE-2 se presenta en cursos de Ingeniería Informática y de Ingeniería Electrónica de la Universidad de Granada, según distintos niveles de complejidad y abordándose su diseño en profundidad de forma completa y utilizando diversas metodologías de implementación. Concretamente, en la primera de las titulaciones citadas, se considera a CODE-2 en las siguientes asignaturas:

- *"Introducción a los computadores"* (1.1, primer curso, primer cuatrimestre): Programación en lenguajes máquina y ensamblador, y utilización de CODE-2.
- *"Estructura de computadores"* (2.1): diseño completo de CODE-2 al nivel de micromáquina. Hay que hacer notar que en un cuatrimestre previo (1.2) el alumno ha cursado la asignatura *"Tecnología de Computadores I",* donde ha estudiado todos los elementos que intervienen en el diseño, que son los siguientes: decodificador binario, selector de datos, adaptador triestado, ALU, registro, biestable asíncrono, banco de registros, reloj y contador binario de 4 bits.
- *"Diseño de circuitos microelectrónicos"* (3.2): diseño VHDL de CODE-2
- *"Síntesis automática de arquitecturas VLSI"* (4.2): implementación de CODE-2 utilizando circuitos lógicos programables.

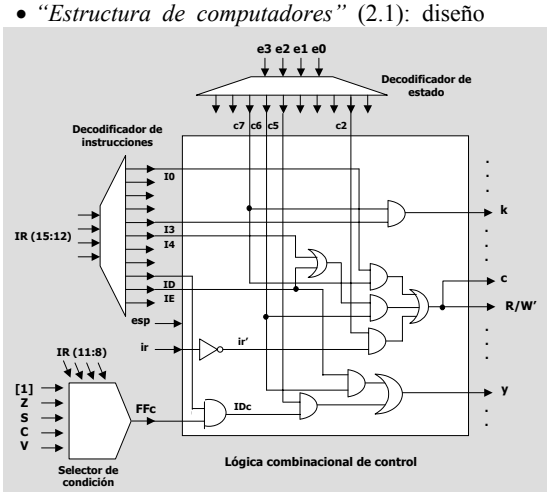


Figura 5. Esquema simplificado de la lógica combinacional de la unidad de control

En consecuencia, consideramos a CODE-2 como una potente herramienta didáctica, utilizable como caso práctico para presentar conceptos de distinto nivel de complejidad.

Referencias

[1] Y.Chu. *Digital Computer Design Fundamentals*. McGraw-Hill, 1962.

[2] G.W.Gorsline. *Computer Organization: Hardware/Software.* Prentice Hall, 1986.

[3] G.M.Karam, J.C.Bryant. *Principles of Computer Systems.* Prentice Hall, 1992.

[4] M.M. Mano. *Computer System Architecture.* Prentice Hall, 1976.

[5] M.M.Mano. *Arquitectura de computadores.* 3ª Ed. Prentice Hall, 1994.

[6] M.M.Mano, C.Kime. *Logic and computer design fundamentals.* Prentice Hall, 1997.

Tabla 3. Algunas de las 29 funciones de conmutación que sintetizan las señales de control de CODE

```
b= c1·ir'c2·ir'+ c3·ir'+ c5·(I2+I4+I5) + c6(I0+I1)+ c7·(I3+IDc+IE)
c= c2·ir'+c6·(I4+IE) + c7·I0
cl= c5·(I2+I9+IA+IB+IC+IF)+c6·(I6+I7+I8)+c7·(I3+I4+I5+IE)+c8·(I0+I1)+c9·ID
d= c2·ir'+ c6·(I5+I4+IE) + c7·(I0+IDc+IE+c7·IE)
ir_cl= c9·IDc·ir
k= c7·I3
l= c5·(I2+I3+I4 +I5)
x= c5·(I0+I1+IC·FFc)+ c6·(I6+I7+I8)+c9·IDc·ir'
y= c5·IDc+c6·IE
IO/M'= c6·I4+ c7·I5
R/W'= c
```


[7] C.C.Foster, *Computer Architecture,* 2nd Ed. Van Nostrand, 1976.

[8] D.Knuth. *The art of computer programming,* Volume I, 3rd Ed. Addison Wesley, 1997.

[9] G.Lee. *From Hardware to Software. An Introduction to Computers*. McMillan, 1982.

[10] A.S.Tanenbaum. *Structured Computer Organization.* 3rd Ed., Prentice Hall, 1990.

[11] J.L.Hennessy, D.Patterson. *Computer Architecture: A Quantitative Approach,* 2nd Ed. McGraw-Hill, 1996.

[12] J.D.Carpinelli. *Computer Systems. Organization & Architecture,* Addison Wesley, 2001.

[13] Y.N.Patt, S.J.Patel. *Introduction to computing systems,* McGraw-Hill, 2001.

[14] A.S. Tanenbaum, *Organización de computadoras. Un enfoque estructurado,* 4ª Ed. Pearson Educación, 2000.

[15] M.J. Murdocca, V.P. Heuring. *Principles of Computer Architecture,* Prentice Hall, 2000.

[16] D.A. Patterson, J.L.Hennessy. *Estructura y diseño de computadores*, (3 volúmenes) Editorial Reverté, 2000.

[17] P.M.Miguel, *Fundamentos de los computadores*. Paraninfo, 1999.

[18] A.Prieto, A.Lloris, J.A.Romera. *Realización de un ordenador didáctico elemental,* V Congreso de Informática y Automática. A.E.I.A; pp. 611-615. Madrid. 1982.

[19] F.J.Pelayo, A.Prieto, A.Lloris, F.Gómez-Mula. *Emulador y ensamblador de un ordenador didáctico elemental,* Rev. de Informática y Automática. Vol. 17, Nº 161, pp. 7-17. 1984.

[20] A.Prieto, A.Lloris, J.C.Torres. *Introducción a la Informática*. 1ª Ed. McGraw-Hill, 1989.

[21] A.Prieto, F.J.Pelayo, A.Lloris, F.Gómez-Mula. *Description and use of a simple didactic computer,* EC Newsletter (Education in Computing Computers in Education). Vol.2; N1 1, Jan-Apr 19-90, pp.17-29. 1990.

[22] E.Ayguadé, J.J.Navarro, M.Valero-García. *La màquina senzilla. Introducció a l'estructura básica d'un computador*. Col·lecció Aula. CPET, 1992.

[23] R.Hermida, A.M. del Corral, E.Pastor, F.Sánchez, *Fundamentos de computadores,* Síntesis, 1998.

[24] E.Pastor, F.Sánchez. *La Máquina Rudimentaria: un procesador pedagógico.* III Jornadas de Enseñanza Universitaria sobre Informática (JENUI'97), págs. 394-402, Junio 1997.

[25] J.Ruz. *De la tecnología a la arquitectura de computadores,* Síntesis, 1997.

[26] G.Fernández. *Conceptos básicos de arquitectura y sistemas operativos*. Sistemas y Servicios de Comunicación, 1998.

[27] F.Sánchez. *Características deseables en un procesador pedagógico para la enseñanza básica de la Arquitectura de Computadores.* Actas de las VII Jornadas de Enseñanza Universitaria de la Informática, pp. 68-73, Palma de Mallorca, 16-18 Julio 2001.

[28] *IEEE Standard for Microprocessor Assembly Language.* IEEE Std. 694-1985. The Institute of Electrical and Electronics Engineers, Inc. New York., June 30, 1985.

[29] A.Prieto, A.Lloris, J.C.Torres. *Introducción a la Informática*. 3ª Ed. McGraw-Hill, 2002.

Enseñanza de la Unidad Aritmético-Lógica en la asignatura de Arquitectura de Computadores de I. T. Informática de Gestión

Antonio J. de Vicente, Manuel Prieto Abraham del Río
Departamento. de automática
Universidad de Alcalá
28871 Alcalá de Henares
e-mail: avicente@aut.uah.es

Resumen

Presentamos en este artículo una aplicación para la enseñanza a distancia del tema de Arquitectura de Computadores: la unidad Aritmético-Lógica. La asignatura pertenece al segundo curso de la titulación de I. T. Informática de Gestión. La herramienta ha sido desarrollada en Flash y se ha enfatizado el empleo de las animaciones para que los alumnos puedan comprender mejor el funcionamiento de los diferentes componentes de la ALU.

1. Introducción

Cada vez va cobrando una mayor relevancia la necesidad de formación continua y un fenómeno que conlleva: la educación a distancia. En la Escuela Politécnica de la Universidad de Alcalá hemos visto como cada año un gran número de alumnos simultanea su formación con su actividad laboral; y somos conscientes de que, por tal motivo, no pueden acceder a las clases con regularidad. Además, gracias al abaratamiento progresivo de las Nuevas Tecnologías de la Información y las Comunicaciones, la mayor parte de los alumnos tienen acceso a Internet, ya sea desde sus puestos laborales, ya sea desde sus domicilios. Por ello, hemos desarrollado una aplicación accesible desde Internet que explique de la forma más gráfica posible la estructura y comportamiento de la Unidad Aritmético-Lógica.

La estructura de la ponencia es la siguiente:

- El punto 2 explica la motivación que condujo al desarrollo del presente trabajo
- El punto 3 muestra el desarrollo que se ha seguido, así como las diferentes consideraciones de diseño que se barajaron.
- El punto 4 presenta los temas tratados en la aplicación: teoría, animaciones, ejercicios
- El punto 5 muestra la metodología didáctica seguida en la aplicación.
- El punto 6 presenta la interfaz de usuario de la aplicación.
- Finalmente, el punto 7 recoge las conclusiones del trabajo

2. Motivación

Es cada vez mayor el impacto que tienen las Nuevas Tecnologías de la Información y la Comunicaciones en todos los ámbitos de la vida social. Ya no resulta extraño que existan particulares que, al igual que puedan contratar los servicios de una televisión digital, adquieran con una compañía telefónica una tarifa plana para acceder a Internet. Debido a esto, Internet se ha convertido en un proveedor de información al que los propios estudiantes pueden acceder en busca de los datos que precisen para su estudio.

Una gran mayoría de los estudiantes, de la Escuela Politécnica de la Universidad de Alcalá simultanean su formación académica con su actividad laboral, no pudiendo asistir de una manera regular a las clases. Aunque la Universidad Nacional de Educación a Distancia (UNED) nació con la idea de formar a tales estudiantes, desgraciadamente, no se imparten todas las titulaciones que los alumnos pueden desear.

Una primera manera de ayudar a los estudiantes que no pueden asistir a las clases es con la creación de una página web de la asignatura que contenga, al menos, las normas, temario y método de evaluación, e ir activando los enlaces a dichas informaciones según se imparten en clase, de manera que el alumno virtual pueda conocer el ritmo con el que se imprimen las clases. No obstante, quedaría reducida a una descarga de apuntes, y la normativa de la asignatura, que no está nada mal y que requiere un gran esfuerzo por parte del profesor para tenerla actualizada y útil para los alumnos, pero solamente consistiría en un mero repositorio de información.

Es conocido, que una presentación visual, conjuntamente con una exposición oral hace que se recuerde cerca del 65% de la información a los tres días. Por ello, estamos desarrollando animaciones que puedan facilitar la comprensión de determinados temas por parte de los alumnos. Si se cuenta con un proyector en el aula podrían ser mostradas a los estudiantes como parte de la lección magistral, pero, ¿qué ocurre con los alumnos a distancia? Podrían descargarse la animación, pero entonces, ¿cómo poder recibir su retroalimentación? La solución fue crear una misma aplicación que sirviese como animación y como herramienta de educación a distancia. Las diferentes alternativas que tratamos se tratarán en el siguiente apartado.

3. Desarrollo

Este apartado recoge las diferentes alternativas de diseño que consideramos a la hora de desarrollar la aplicación, la cual, debía satisfacer los objetivos siguientes:

- Para el alumno: una herramienta sencilla de usar y que mediante animaciones y ejercicios complete los conocimientos adquiridos en las clases presenciales o en el estudio de la bibliografía propuesta.

- Para el profesor: una aplicación que permita realizar el estudio de resultados sobre una población relativamente numerosa (208 alumnos matriculados en el curso 2001-2002) identificando de forma clara los puntos en los que los alumnos han tenido más problemas y el tipo de errores cometidos.

En una primera aproximación pensamos realizar diferentes tipos de interfaces según el mayor o menor ancho de banda que pudiesen tener los alumnos al acceder a la aplicación vía Internet. Pero nos encontramos con el problema de que las animaciones solamente estaban desarrolladas con tecnología Flash y por lo tanto, una página en modo texto no tenía sentido con lo que el primero de los dos objetivos no se alcanzaba.

La segunda idea que contemplamos fue la de que los alumnos que deseasen trabajar con la aplicación pudiesen descargarse la misma y trabajar con ella de manera local. El inconveniente en este caso, es que si bien el requisito para los alumnos se cumplía, no ocurría así con el segundo, también fundamental para que el profesor reciba información de en qué puntos tienen más dificultades o comenten más errores para poder reforzarlos en las clases presenciales.

Finalmente, se adoptó la idea de generar una única aplicación a la que los alumnos pudiesen acceder desde Internet, con las animaciones realizadas en Flash y que fuese archivando y analizando el número y tipo de los errores para llevar una estadística que sirviese al profesor.

La aplicación se ha diseñado entonces de manera que interactúe con una base de datos en la que se vayan almacenando los datos de los alumnos: número de intentos, ejercicios resueltos con éxito y ejercicios con errores. Esta información servirá tanto al alumno que le llegará en forma de calificación, como al profesor que le servirá de guía para tener constancia de los puntos en los que los estudiantes tienen más problemas.

Al tratarse de una aplicación que funcionará en un entorno web necesitamos herramientas que nos den servicio web. Llegados a este punto existen en el mercado una gran cantidad de servidores web, cada uno con sus ventajas e inconvenientes. Nuestra elección es la siguiente: Servidor web utilizado Apache Web Server, el Apache es uno de los servidores web más utilizados en el mundo, debido a su gran rapidez

para servir HTML a los clientes, además es una herramienta totalmente gratuita y de código libre.

Por último también necesitaremos para completar el entorno de programación un servidor de bases de datos, dónde se almacenarán los datos de la aplicación. Existen una gran cantidad de productos de este tipo en el mercado, pero principalmente los más utilizados son los siguientes:

- **SQL Server:** tiene una alta velocidad de consultas, un entorno intuitivo. La pega es que solamente sirve para plantaformas Microsoft y no es gratuito.
- **Oracle:** Multiplataforma y permite la optimización de bases de datos, pero tampoco es gratuito.
- **MYSQL:** es gratuito y multiplataforma pero es difícil de configurar y presenta un entorno poco intutitivo

Dado el carácter del proyecto, las herramientas utilizadas deben ser de libre distribución. Por lo tanto la elección del gestor de base de datos se simplifica bastante, ya que el único de libre distribución (para cualquier plataforma) es MYSQL.

4. Temas tratados

La asignatura de Arquitectura de Computadores se encuentra ubicada en el primer cuatrimestre del segundo curso de la titulación de I. T. Informática de Gestión. Los alumnos, llegan a ella con conocimientos previos tanto de Electrónica como de Estructura de Computadores.

La aplicación muestra el tema 1 de la asignatura: "La Unidad Aritmético-Lógica" [1] [2] [3] [4] [5]

Cada apartado del tema es tratado de manera diferente debido a la importancia que creemos puede tener en la asignatura.

Así, se hace mucho hincapié en la operación de suma, ya que ésta está siendo empleada continuamente, no sólo como incrementos en programas, sino para poder actualizar el contador de programa para apuntar a la dirección de la instrucción siguiente, el acceder a una posición de memoria casi siempre lleva asociado un direccionamiento relativo lo que implica una operación de suma, etc. Por ello, a partir de un sumador elemental con propagación de acarreo se les mostrará las posibles mejoras que pueden incorporarse y sobre todo los diferentes tipos de sumadores: con anticipación, con selección o salto de acarreo.

El resto de las operaciones aritméticas, multiplicación y división se tratan más de una manera algorítmica, aunque sí se les presenta, en el caso de la multiplicación, un posible circuito para la implementación del algoritmo de suma-desplazamiento.

Todas las operaciones anteriores se realizan en coma fija, reservándose la coma flotante únicamente para la suma en la que se tratará también el tema de los dígitos de guarda y de las diferentes técnicas de redondeo: truncación, forzar el bit menos significativo a uno o redondeo al más próximo.

Como la Unidad Aritmético-Lógica requiere que los alumnos comprendan los diferentes sistemas de representación que existen, la aplicación realiza un repaso obligatorio de los mismos al comienzo.

5. Método didáctico empleado

De todos es sabido que el empleo del ordenador no constituye un sustituto eficaz del profesor, sino que es una herramienta para ayudar a la docencia y el aprendizaje.

Según expone Romero [6] con respecto a los hipertextos “la fragmentación de los contenidos y su consiguiente interconexión, además de una inclusión masiva de elementos gráficos, son los elementos básicos que proporcionan ventaja frente a los libros clásicos”.

Aunque, tal y como expone Rafael Tormo [7] “los contenidos de cualquier disciplina pueden ser desarrollados en formato hipertextual consiguiendo más facilidad de integración a través de las imágenes con el texto”, los contenidos de nuestra aplicación no son estrictamente

hipertextuales, aunque sí existen enlaces que el alumno puede elegir.

En nuestra aplicación, los temas se han secuenciado según un orden lógico de impartición. Por tal motivo, tanto el contenido teórico como el práctico guían el aprendizaje del alumno.

Para evitar la desorientación del estudiante, los botones de navegación guían al alumno tanto en la manera de aprender, como en la forma en que se le presentan lo contenidos. No permitiéndole pasar a un tema posterior si no se conoce el actual.

La forma de navegación proporciona los conocimientos progresivamente. Comienza la aplicación con la presentación de la ALU y posteriormente se presentan los diferentes operadores y operandos que suelen estar presentes en las ALUs.

El estudiante podrá ir avanzando en los contenidos según vaya adquiriendo y consolidando los conocimientos recibidos. A tal efecto existen una serie de ejercicios que el alumno deberá resolver con éxito para poder avanzar al siguiente tema. De esta forma evitamos que se navegue de forma errática por la aplicación. Es más, el estudiante deberá haber resuelto satisfactoriamente, al menos, un ejercicio de cada tema para poder pasar al siguiente. Eso no quiere decir que no pueda realizar más ejercicios del tema actual, sino que no se le obligará a repetir constantemente los ejercicios de un tema determinado.

Las numerosas animaciones de este tema pueden ser evitadas también, una vez que el alumno ya las ha visto por primera vez. Dándole la opción en todo momento de "*Ver animación*" o "*Saltar animación*"

Hemos desarrollado, básicamente, dos tipos de animaciones, las que por su sencillez de comprensión pueden presentar todos los detalles numéricos a los alumnos, y las que debido a su complejidad conceptual únicamente se les presenta cómo fluye la información a través de los elementos del diagrama, siendo trabajo posterior de los estudiantes el realizar ejercicios sobre el tema para consolidar su aprendizaje.

Finalmente, se quiere indicar que la resolución de los ejercicios, aciertos y fallos cometidos son analizados y tratados por la aplicación en dos sentidos diferentes: a) de cara al alumno, b) de cara al profesor.

Para el alumno los fallos cometidos le orientarán sobre los temas en los que debe hacer más hincapié mediante el empleo de mensajes de texto. En particular se tiene en cuenta si el alumno contesta al azar los ejercicios.

Para el profesor, la tasa de aciertos y de fallos le indicarán qué conceptos del tema son mejor o peor comprendidos por los alumnos y en dónde debe reforzar sus explicaciones en las clases.

6. Interfaz de usuario

Se ha prestado especial interés en diseñar una interfaz de usuario sencilla que permita una navegación ágil por el contenido de la aplicación. Aunque se trata de una asignatura de segundo curso de I.T. Informática de Gestión, la interfaz se ha desarrollado pensando en usuarios con poco conocimiento de informática o conocimientos a nivel de usuario.

Hemos distinguido la navegación entre pantallas y la navegación dentro de la pantalla, y de acuerdo con esta división se han desarrollado los diferentes elementos de la interfaz.

La figura 1 muestra la barra de navegación entre pantallas. Básicamente consiste en dos flechas avanzar o retroceder entre una pantalla y la siguiente (o la anterior) También presenta un botón para ir al comienzo del tema, lo que evita que la navegación sea errática y tener que volver a pasar por elementos ya visitados. Dichos botones se muestran en la parte inferior de la pantalla.

Figura 1: Botones de navegación entre pantallas

En cuanto a la navegación dentro de la pantalla, existen diferentes opciones: pantallas sin animaciones y con animaciones; pudiendo en estás últimas saltarse, en casi todas, la animación si el alumno no desea verla. El motivo por el que no todas las animaciones pueden saltarse es porque en algunos casos se ha considerado sumamente importante para la comprensión del tema tratado, tal y como es el caso del sumador con anticipación de acarreo por bloques mostrado en la figura 2.

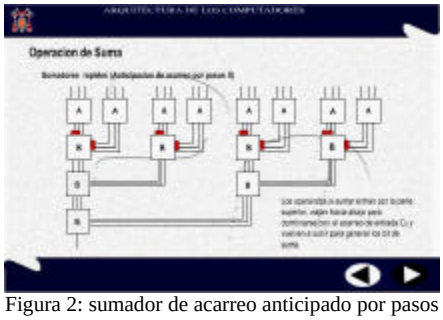


Figura 2: sumador de acarreo anticipado por pasos

La presentación de las animaciones en las pantallas puede mostrarse de manera explícita, tal y como muestra la figura 3, o implícita, pasando el ratón sobre el elemento lo que le dará al alumno la posibilidad de arrancar la animación o no tal y como muestra la figura 4.

Figura 3: Navegación dentro de la pantalla

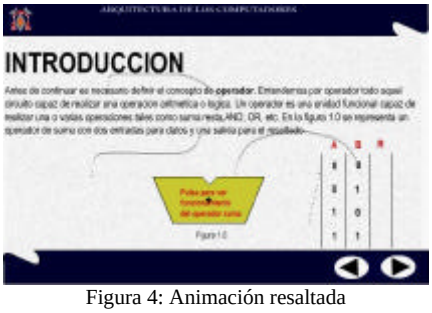


Figura 4: Animación resaltada

También, dentro del contenido de las pantallas existen hiperenlaces a otras pantallas, para lo cual, tal y como muestra la figura 5, se resalta en azul dicho hipervínculo.

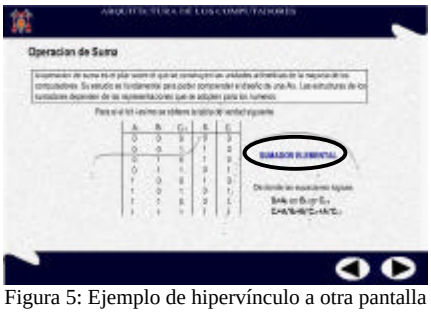


Figura 5: Ejemplo de hipervínculo a otra pantalla

También es importante mostrar la interfaz que se ha desarrollado para los ejercicios. Básicamente consisten en dos tipos de pantallas, la de enunciado y la de resolución del mismo, mostradas en las figuras 6 y 7.

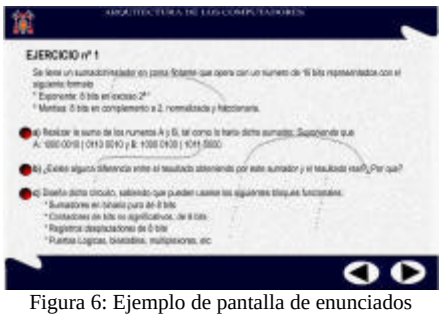


Figura 6: Ejemplo de pantalla de enunciados

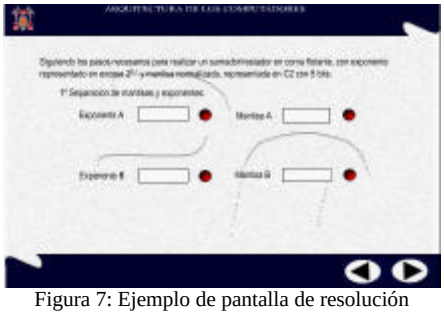


Figura 7: Ejemplo de pantalla de resolución

Por último, y como resumen del apartado, indicar que se ha empleado un código de colores para facilitar la nemotécnia de la interfaz. Así, la navegación entre pantallas será en negro, los hiperenlaces en azul y los botones de acción dentro de las pantallas en rojo, tal y como recoge la figura 8

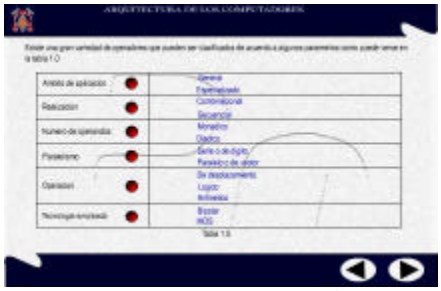


Figura 8: resumen de la interfaz

7. Conclusión

En este artículo hemos presentado una aplicación didáctica para la Web sobre la Unidad Aritmético Lógica del Computador dentro de la estructura clásica de un computador. El objetivo de la misma es presentarla a los alumnos durante el próximo curso y evaluar si existen mejoras en los rendimientos académicos con respecto a años anteriores, para lo cual diseñaremos un experimento que permita recoger datos estadísticos suficientes como para marcar una tendencia.

En caso de que existan mejoras sensibles, se empezarían a desarrollar nuevas aplicaciones que cubriesen el resto de los temas de la asignatura de Arquitectura de Computadores de la titulación I.T.Informática de Gestión y que pudiesen ver mejorada su impartición mediante una aplicación de estas características. No obstante, no todos los temas requerirán animaciones para su comprensión.

Asimismo, también se ha pensado distribuir la aplicación en otra serie de carreras afines, en las que el temario de Arquitectura de Computadores sea lo suficientemente similar como para poder ser empleadas. De momento, se ha pensado en las carreras e I. T. Informática de Sistemas en la asignatura de Arquitectura de Computadores I y la de I. T. Telecomunicación, especialidad Telemática en la asignatura de Arquitectura de Computadores.

Referencias

[1] John I. Hennesy, David A. Patterson. *Arquitectura de Computadores. Un enfoque cuantitativo*. Mc Graw-Hill. 1995.

[2] David A. Patterson, John I. Hennesy. Estructura y Diseño de Computadores. Reverté. 2000

[3] William Stallings. *Organización y Arquitectura de Computadores*. Prentice may. 1996

[4] Pedro de Miguel Anasagasti. Fundamentos de los Computadores. Paraninfo. 1999

[5] José Antonio de Frutos, Rafael Rico. Arquitectura de Computadores. Servicio de publicaciones de la Universidad de Alcalá. 1995

[6] R Romero. Diseño de páginas para una Red accessible.. Quaderns Digitals, 1998

[7] Rafael Tormo Molina. Uso de hipertextos para facilitar el aprendizaje en la Universidad. Comunicación Pedagógica, Septiembre 2000. Págs 22-27

[8] Giovanni Sartori. Homo Videns. LA sociedad teledirigida. Taurus. 1988.

[9] Nicholas Negroponte. El mundo digital. B.S.A. 1995.

[10]Luis Joyanes. Cibersociedad. McGraw-Hill. 1997

[11]Jesús González Boticario, Elena Gaudioso Vázquez. Aprender y formar en Internet. Paraninfo. 2000

Facilitando el aprendizaje de la Arquitectura del Juego de Instrucciones

José M. Claver Iborra, María Isabel Castillo Catalán
Depto. de Ingeniería y Ciencia de los Computadores
Universitat Jaume I
12071 Castellón
e-mail: {claver | castillo}@icc.uji.es

Resumen

En este artículo se presenta la metodología y estrategias llevadas a cabo para la enseñanza de la Arquitectura del Juego de Instrucciones en la asignatura de Estructura de Ordenadores de 2º curso de Ingeniería Informática y en particular dentro del desarrollo de sus prácticas. Éstas están dedicadas a la introducción del lenguaje ensamblador de un procesador de propósito general. La metodología utilizada tiene en cuenta, entre otros aspectos, la elección del tipo de procesador y la herramienta de trabajo, la personalización del ritmo de trabajo y la responsabilidad del alumno en la consecución de los objetivos como ejes fundamentales del aprendizaje. Así, se pretende conseguir un acercamiento mucho menos traumático del alumno a una de las asignaturas que mayor importancia tiene en la formación de base de los futuros ingenieros informáticos. Los resultados obtenidos hasta el momento nos indican que la acogida por parte del alumno es positiva, quedando ésta reflejada en la facilidad e interés con que resuelven los problemas propuestos en cada práctica. Por ello se produce una mejora en el aprendizaje de la asignatura, aspecto que queda reflejado en la evaluación de su rendimiento.

1. Introducción

La docencia de Arquitectura de Computadores en las titulaciones de informática aparece generalmente en el primer curso y se prolonga a lo largo de toda la carrera, variando en intensidad y profundidad según se trate de Ingeniería Informática (I.I.), Ingeniería Técnica en Informática de Sistemas (I.T.I.S.) o Ingeniería Técnica en Informática de Gestión (I.T.I.G.). Los contenidos de esta materia son en la actualidad, y previsiblemente lo serán en los próximos 10 a 20 años, un aspecto fundamental en la formación de los futuros informáticos [2], ya que el desarrollo de aplicaciones y sus prestaciones tienen un importante impacto en la arquitectura, estructura y organización de los computadores [10, 11, 14]. Por otra parte, no se prevé en este periodo una modificación importante en las bases de la construcción y funcionamiento de los computadores. Actualmente estamos utilizando algunas ideas que aparecieron hace más de 30 años, aunque habrá que estar a la expectativa de iniciativas que proponen modelos de computación muy diferentes a los que conocemos hoy en día, como la computación cuántica o molecular [4].

Si bien hay un acuerdo generalizado en cuanto a los contenidos que deben impartirse en los primeros cursos de arquitectura de computadores [1,2], existen diversas alternativas referentes al tipo de procesador y herramientas utilizadas para la descripción y estudio de su funcionamiento [6]. También puede variar la profundidad del estudio de los diferentes elementos del computador [7], dada la alta sofisticación de los computadores actuales y la variedad de equipos con los que nos podemos encontrar. De especial interés es el estudio del procesador y la arquitectura de su juego de instrucciones, y de manera muy especial la programación en ensamblador, el uso de la memoria y la gestión de la entrada/salida. Por ello, los cursos introductorios de Arquitectura de Computadores hacen hincapié en estos temas.

El complemento de esta materia lo constituiría el estudio de los aspectos más tecnológicos del diseño de computadores, pero estos caen fuera de los contenidos tratados en el presente artículo.

Como veremos a lo largo de este trabajo, la elección del procesador o las herramientas utilizadas para la enseñanza de esta materia son importantes, pero también influyen de forma decisiva el ritmo y la responsabilidad en el aprendizaje de los conceptos y técnicas que queremos trasladar. Esto determinará que los alumnos puedan, en su gran mayoría, alcanzar las habilidades que se pretenden al finalizar el curso, o que muchos de ellos se descuelguen del fino hilo conductor que les liga a la asignatura, y que a partir de un momento dado les parezca kafkiana y la abandonen.

El resto de este trabajo se organiza como sigue: En el apartado 2 describimos algunos de los aspectos más destacados relacionados con la docencia en las asignaturas introductorias de Arquitectura de Computadores. En el apartado 3 hacemos un pequeño repaso a la docencia de estas asignaturas en otras universidades y en el apartado 4 desarrollamos los aspectos más destacados de nuestra propuesta educativa. En el último apartado, presentamos las conclusiones derivadas de la experiencia de nuestra propuesta y planteamos las mejoras que abordaremos en próximos cursos.

2. Docencia introductoria a la arquitectura de computadores

Los contenidos de las asignaturas introductorias de arquitectura de computadores incluyen, entre otros, el estudio de los siguientes ítems:

- Procesador: Estructura y organización, ruta de datos, juego de instrucciones y lenguaje máquina, lenguaje ensamblador y su relación con lenguajes de alto nivel.
- Memoria: Organización, almacenamiento de instrucciones y datos y cache.
- Entrada/Salida: Gestión mediante consulta de estado e interrupciones.
- Rendimiento: Análisis de prestaciones y comparativas entre computadores.

Estos contenidos deben transmitirse al alumno, tanto a través de la docencia en el aula como de su trabajo en el laboratorio, de tal forma que adquiera los conocimientos y habilidades establecidos en los objetivos del curso. En el aula deben fijarse los aspectos más conceptuales de la materia, mientras que en el laboratorio estos deben reforzarse y ampliarse mediante su uso y la adquisición de habilidades de análisis y síntesis. Así, suele ser habitual en estas asignaturas potenciar el estudio en el laboratorio del lenguaje ensamblador (exoarquitectura) mediante el uso de una máquina real o un simulador que esté estrechamente relacionado con el modelo presentado en el aula (más relacionado con la endoarquitectura y la microarquitectura [8]). En estas prácticas deben manejarse todos los elementos conceptuales vistos en teoría de forma aplicada, lo que supone seguir en lo posible el ritmo y orden de los conceptos introducidos en el aula.

Como esta materia no puede verse de forma abstracta, existe un momento en el diseño de su proyecto docente en el que hay que decidir el procesador ejemplo y la herramienta o entorno de trabajo a utilizar en cada asignatura de entre varias alternativas.

2.1. Procesador

En el dilema entre procesador real (comercial) o procesador ficticio (hipotético) parece claro que el interés de un procesador real está en la posibilidad de utilizarlo como parte integrante de un dispositivo real como es un computador comercial o un sistema de entrenamiento especialmente diseñado para las prácticas, algo imposible de ver con procesadores ficticios. Pero los procesadores reales habitualmente introducen particularidades, que pueden ser poco generalizables. Esta característica, en el caso de los primeros cursos, puede introducir complejidades añadidas en el proceso de aprendizaje, así como introducir una visión poco acorde con la generalidad de los procesadores, por ello la importancia de una adecuada elección.

Como contrapartida los procesadores ficticios pueden adaptarse a las necesidades educativas en cada momento de la carrera, en función de la preparación del alumno para asimilar, y aplicar los conceptos y técnicas que estos integran. Estos procesadores suelen diseñarse eligiendo abstracciones de las características más generalizadas en los procesadores reales.

2.2. Herramienta de trabajo

La elección de la herramienta a utilizar varía entre

el uso directo de un computador comercial o un sistema de desarrollo, y el uso de un simulador. Sólo es posible utilizar directamente un computador si hemos optado por un procesador real, y entonces el ensamblador y el software utilizado deben adaptarse a las características de dicho procesador, y a las posibilidades y restricciones de la máquina en el que se encuentre integrado. En estos casos muchos de los análisis del comportamiento de procesador tienen que ser indirectos y están limitados por la estructura particular de la máquina (memoria, cache, bus, etc.).

Si se elige un simulador, éste puede estar diseñado para un procesador real o para un procesador ficticio (de origen o como abstracción de un procesador real, con lo que se transforma en un procesador ficticio). El uso de simuladores de procesadores ficticios permite una adecuación mayor a las necesidades de un determinado curso e ir añadiendo, en cursos posteriores, ampliaciones o modificaciones de las abstracciones iniciales. De esta manera se van introduciendo conceptos cada vez más avanzados sin necesidad de que el alumno tenga que aprender una nueva herramienta ni un nuevo lenguaje ensamblador.

En cualquiera de los dos casos es posible utilizar recursos del computador (directa o indirectamente), aunque estos siempre suelen ser mucho más limitados en el caso de los simuladores.

3. Docencia en otras universidades

En los últimos años, la experiencia docente en las asignaturas de introducción a la arquitectura de computadores se ha caracterizado por:

- Elección de un primer procesador simple o sencillo ficticio en vez del procesador real de 8 bits (8085, Z80, 6800 ó 6500) de hace unos años, para conseguir reducir el escalón entre los conocimientos del alumno y los conceptos introductorios [3, 13] en los primeros cursos de introducción a los computadores.
- Continuación en el segundo curso con un procesador real, normalmente un CISC de los años 80, de 16 bits (Intel i8086 o Motorota MC68000) y en algunos caso un procesador RISC de 32 bits (como el MIPS R2000 simplificado, el ARM, Motorota MC88010 u otros), para obtener una mayor aproximación a la complejidad de los procesadores reales [6].
- Uso de simuladores o combinación de estos con sistemas de desarrollo o computadores comerciales, que permiten ver con comodidad la visualización del estado y comportamiento del procesador, y la posibilidad de realizar aplicaciones sobre hardware real, respectivamente.

En las universidades españolas suelen darse las características anteriores, aunque con algunas variaciones. Hemos elegido para mostrar este hecho las experiencias llevadas a cabo por cuatro de éstas en las asignaturas de los dos primeros cursos de Ingeniería Informática (Universidad Complutense de Madrid (UCM), Universidad Politécnica de Cataluña (UPC), Universidad Politécnica de Madrid (UPM) y Universidad Politécnica de Valencia (UPV)). La elección se ha basado en la antigüedad de sus estudios, la experiencia de su profesorado y el número de alumnos, sin menoscabo de aquellas aquí omitidas.

En el primer curso se aborda, en todas ellas, el estudio de un procesador simple: en la UPM se utiliza el picocomputador, y en la UPC y la UCM la máquina sencilla [3]. Estos procesadores son completamente ficticios, mientras que en la UPV se estudia el procesador R2000 en su abstracción simplificada no segmentada [14]. En todos los casos se hace uso de un simulador como herramienta de trabajo.

Ya en el segundo curso se utilizan procesadores reales de tipo CISC en la UCM (MC68000, sobre un sistema de desarrollo) y en la UPC (i8086, sobre un PC), combinando, en ambas, el uso del ensamblador y del lenguaje C. Por el contrario, se utilizan simuladores de procesadores RISC en la UPM (MC88010) y la UPV (abstracción del MIPS R2000).

4. Nuestra propuesta

La asignatura de Estructura de Ordenadores (EO) de 2º curso de la Ingeniería Informática es la segunda de las asignaturas dedicadas a la Arquitectura de Computadores, después de la

asignatura Introducción a la Informática, de primer curso, donde se introducen conceptos muy elementales acerca del funcionamiento de los computadores. Por tanto, la enseñanza del juego de instrucciones del procesador y, por ende, el aprendizaje del lenguaje ensamblador forma parte importante de los contenidos de EO.

Hasta el curso pasado el microprocesador elegido (desde hace 10 años) para las prácticas era el MC68000, uno de los exponentes más elegantes de la arquitectura CISC. Por muchas razones éste es uno de los procesadores más utilizados en la enseñanza de la Arquitectura de Computadores y sobre el que se han escrito más libros, aunque está muy alejado de los procesadores seminales de las actuales arquitecturas superescalares. Las prácticas se realizaban sobre un sistema de desarrollo que sólo podía ser utilizado en el laboratorio, con un entorno de trabajo bastante tedioso, y sin la posibilidad de que el alumno pudiera utilizarlo en su casa.

Cada sesión de prácticas se organizaba de la siguiente forma: comenzaba con una larga explicación por parte del profesor, donde se introducían muchos conceptos nuevos y se fijaban los objetivos de la práctica. Estos objetivos eran claros, pero muy ambiciosos, y requerían que el alumno, por un lado, prestara mucha atención durante la explicación realizada por el profesor, y por otro, antes de empezar con el desarrollo de la práctica, realizara un concienzudo estudio del enunciado de ésta, donde se incluía una breve descripción de los conceptos anteriormente explicados. Ambas tareas ocupaban una parte importante de la sesión, dando lugar a que el alumno dispusiese de muy poco tiempo para desarrollar los problemas propuestos en la práctica. Además, y aunque las prácticas estaban diseñadas en orden creciente de complejidad, desde la primera práctica el alumno tenía que enfrentarse al análisis y diseño de un programa en ensamblador más o menos complejo, con declaración de datos en memoria, instrucciones de distinto tipo y diversos modos de direccionamiento. Esta última circunstancia se da habitualmente en la enseñanza del lenguaje ensamblador en las universidades antes analizadas.

Todo lo anterior nos llevó a plantearnos una alternativa que hiciera más sencillo y cómodo para el alumno conseguir los objetivos que pretendíamos para estas prácticas. Esta debía basarse en un conjunto de condiciones iniciales que facilitaran la obtención de nuestros objetivos y que resumimos a continuación:

1. Sencillez de la arquitectura del procesador.
2. Abstracción de un procesador real más avanzado.
3. Sencillez de uso del simulador.
4. Posibilidad de seguir las prácticas en casa.
5. Posibilidad de autoaprendizaje.
6. No obligatoriedad de asistencia con horario fijo.
7. Ritmo de aprendizaje personalizado.

Para alcanzar la primera condición se ha elegido el procesador MIPS R2000, y en particular una abstracción no segmentada de éste que evita las dificultades asociadas a su estructura y los problemas de su programación real [14]. Este procesador es de tipo RISC, por lo tanto con un conjunto de instrucciones sencillo y modos de direccionamiento reducidos. Sin embargo, esta elección permitirá en cursos más avanzados el uso del mismo procesador, o similar, sin las actuales simplificaciones [9,11], con lo que alcanzamos la segunda condición.

Las condiciones 3 y 4 se consiguen mediante el uso del simulador SPIM, en particular su versión gráfica XSPIM, desarrollado por James R. Larus de la Universidad de Wisconsin, que funciona tanto en plataformas Linux como DOS/Windows [12]. Se trata de un simulador integrado, donde toda la información está visible en todo momento, y muy fácil de utilizar. En las últimas versiones (a partir de la 6.3) tiene la posibilidad de mostrar las dificultades de programación del MPIS R2000 segmentado con la introducción de cargas y saltos retardados (delayed load y delayed branch). Además, gracias a que se trata de un simulador de libre distribución y multiplataforma, el alumno puede utilizarlo en casa.

Las tres últimas condiciones se consiguen gracias a la planificación y desarrollo del material de prácticas. El contenido de las sesiones de laboratorio debe diseñarse para que introduzcan en cada una de ellas nuevos conceptos, cada vez más complejos a medida que avanza el curso, y en las que deben utilizarse los conceptos vistos en las sesiones anteriores. La Figura 1 muestra de dentro hacia fuera el contenido de las sesiones prácticas programadas en la asignatura.

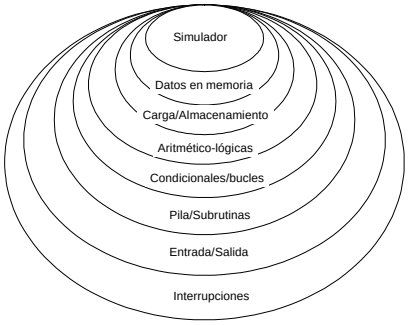


Figura 1. Contenidos de las sucesivas sesiones prácticas.

Dentro de cada sesión se debe tener especial cuidado con el ritmo del aprendizaje del alumno y con no violar la máxima de que primero hay que analizar antes de pasar a la síntesis. Por ello se comienza siempre con una breve introducción y unos programas ejemplo en los que el alumno debe analizar su comportamiento. Con posterioridad se le proponen diversos cambios sobre el programa ejemplo que vuelven a analizar y en los que se pretende aumentar su participación. Finalmente, y antes de seguir adelante, se le propone un pequeño problema de síntesis para comprobar su comprensión de la técnica o concepto introducido. Estos pasos se resumen el la Figura 2.

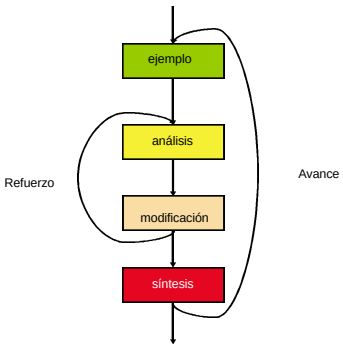


Figura 2. Flujo de aprendizaje de las prácticas.

Así, las prácticas se estructuran en forma de sucesivas cuestiones que demandan del alumno acciones de análisis de programas ejemplo, modificaciones de estos y diseños completos a partir de lo aprendido hasta ese momento (como puede verse en la Figura 3). Cada práctica concluye con problemas más complejos que adquieren la estructura de pequeños proyectos. Todas estas prácticas se encuentran recogidas en un libro de prácticas publicado a tal efecto que puede conseguirse también electrónicamente a través de la página Web de la asignatura [5].

Cuando el alumno tiene dudas acerca de algún concepto o técnica, sólo tiene que revisar las sesiones anteriores y consultar o repetir alguna parte de las prácticas ya realizadas. De esta forma, la necesidad de un profesor todo el tiempo al lado del alumno se reduce, el alumno va, paso a paso, resolviendo las cuestiones que se le plantean y aprendiendo de forma activa, por lo que la asistencia a las sesiones de prácticas no es obligatoria. Como es lógico, tampoco se exige al alumno un ritmo de aprendizaje determinado, y éste puede llevar su propio *tempo*. Lo que se le recuerda, en el laboratorio, en las tutorías y en la página Web de la asignatura, es el ritmo general. Así, el alumno al que le pueda costar más alguna práctica sabe que debe trabajar fuera del horario de las sesiones programadas, en casa o en el horario de acceso libre al laboratorio, para llegar al final del curso con todas las prácticas realizadas. De ésta forma, podrá afrontar con mayores garantías la superación de la evaluación de esta parte de la asignatura.

Ejemplo	Crea un fichero con el siguiente código: … Descripción: …
Análisis	**Cuestión 1.** ¿Qué hace? ¿Cuál es el valor de...? **Cuestión 2.** ¿Indica qué instrucciones hacen...? **Cuestión 3**. ¿Si el dato... tiene el valor 5 qué ocurre....?
Modificación	**Cuestión 4.** Modifica el código para que....
Síntesis	**Cuestión 5.** Implementa un programa que....

Figura 3. Estructura general de los apartados de las prácticas.

El manejo del simulador no es un objetivo finalista de las prácticas, y por ello el alumno no deberá demostrar la destreza en su uso en la evaluación. Sí que se le exige, por el contrario,

que sepa analizar, modificar y diseñar pequeños programas utilizando los conceptos y técnicas vistos en las clases prácticas.

En la Figura 4 se puede constatar el incremento del número de aprobados y las notas obtenidas en la evaluación del ensamblador en el curso 00/01, en el que hemos introducido los cambios propuestos, respecto de los cursos anteriores (98/99 y 99/00). Este hecho queda claramente reflejado en las líneas de tendencia de los dos últimos años.

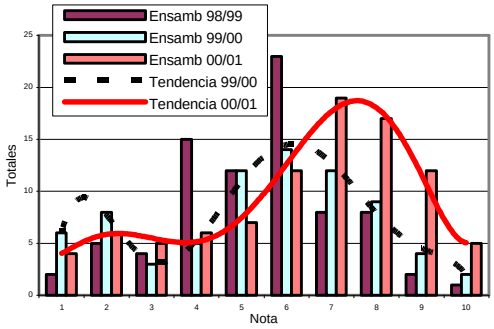


Figura 4. Resultados en la evaluación del ensamblador en los últimos 3 cursos.

También se ha incrementado el número de alumnos presentados a los exámenes en un 20%, por lo que se ha reducido el número de abandonos de la asignatura (y que en mayor número se producen en el primer año en que se cursa ésta). A pesar de ello aún existen notas muy bajas (entre 0 y 3) que suponen el 16 % del total de alumnos presentados, aunque los porcentajes son menores a los del curso 99/00 en el que éstos alcanzaban el 23 %.

5. Conclusiones y trabajos futuros

Las impresiones obtenidas en el laboratorio, a través de la actitud activa de los alumnos y de las apreciaciones personales de aquellos que han repetido la asignatura con el nuevo método, indican que las prácticas se siguen mucho mejor y por tanto el aprovechamiento por parte del alumno es mayor. Este extremo ha sido corroborado por los resultados de la evaluación del ensamblador en el último curso. Además, en las nuevas prácticas se trata la entrada/salida de datos y el manejo de las interrupciones de forma práctica, cosa que no se llegaba a ver en las prácticas de años anteriores.

La metodología descrita en este artículo, parte de la cual queda plasmada en el libro de prácticas elaborado [5], se ha adoptado como guía para el desarrollo de las prácticas de las asignaturas introductorias a la programación en ensamblador de todas las titulaciones de Informática en nuestra Universidad. En el caso de la I.T.I.G. se han seleccionado solo los contenidos que consideramos más importantes para la formación del alumno.

Por último, comentar que otras universidades nos han pedido permiso para adoptar este material como guía para sus prácticas en cursos similares. Para el próximo curso tenemos previsto ampliar este material con la inclusión de las operaciones aritméticas en coma flotante y el uso de las interrupciones para la gestión de tareas.

Referencias

[1] ACM/IEEE-CS Joint Curriculum Task Force. *Computing Curricula 1991*. http://computer.or/education/cc1991.

[2] ACM/IEEE-CS Joint Curriculum Task Force. *Computing Curricula 2001*. http://computer.or/education/cc2001.

[3] E. Ayguadé, J.J. Navarro, M. Valero García. *La màquina senzilla. Introducció a l'estructura bàsica d'un computador*. Col•lecció Aula, CPET, 1992.

[4] A. Barenco, A. Ekert, A. Sanpera, C. Machiavello. *Un saut d'échelle pour les calculateurs*. La Recherche, Nov 1996, http://www.qubit.org/intros/comp/comp.html.

[5] M. Castillo, J. Claver. P*rácticas guiadas para el ensamblador del R2000*. Ediciones Universitat Jaume I. 2001, http://yan.act.uji.es/E38.

[6] A. Clements. *Selecting a Processor for Teaching Computer Architecture*. Microprocessors and Microsystems, mayo 1999.

[7] A. Clements. *The Undergraduate Curriculum in Computer Architecture*. IEEE Micro, pp. 13-22, mayo - junio 2000.

[8] S. Dasgupta. *Computer Architecture - A Modern Synthesis*. John Wiley & Sons, 1989.

[9] E. Farquhar, P.J. Bunce. *The MIPS Programmer's Handbook.*. Morgan Kaufmann, 1993.

[10] J.P. Hayes. *Computer Architecture and Organization*. McGraw-Hill, 1998.
[11] J.L. Hennessy, D.A. Patterson. *Computer Architecture: A Quantitative Approach*. Morgan Kaufmann, 2ª edición,1996.
[12] J.R. Larus. *MIPS. A R2000/3000 Simulator*. University of Wisconsin, http://www.cs.wisc.edu/~larus/spim.html.
[13] E. Pastor, F. Sanchez. *La máquina rudimentaria: Un procesador Pedagógico*. III Jornadas de Enseñanza Universitaria sobre Informática JEUNI'97, pp. 395-402, junio 1997.
[14] D.A. Patterson, J.L. Hennessy. *Computer Organization and Design. The Hardware/Software Interface*. Morgan Kaufmann, 2ª edición, 1997.

Impacto del nuevo software de simulación en las prácticas de estructuras de computadores

Juan-Carlos Cano, Salvador Petit, Julio Sahuquillo
Departamento de Informática de Sistemas y Computadores
E. U. I.-Universidad Politécnica de Valencia
e-mail: {jucano, spetit, jsahuqui}@disca.upv.es

Resumen

Hasta hace pocos años las asignaturas de estructuras de computadores no contemplaban prácticas de laboratorio. De hecho, en la Universidad Politécnica de Valencia (UPV) los planes del 94 fueron los primeros que las incorporaron. La mayoría de las universidades elaboraron estas prácticas en unas condiciones precarias en cuanto a infraestructura (ordenadores, entornos de simulación, entrenadores lógicos, puestos de trabajo, etc.). A medida que los equipos han ganado en potencia y velocidad, la mayoría de estos puestos de trabajo han quedado desfasados y obsoletos para utilizar los entornos de simulación casi profesionales que existen hoy en día. Por otra parte, los nuevos planes de estudio que se están poniendo en marcha en la mayoría de las universidades españolas ofrecen un excelente escenario para la preparación de nuevas prácticas y/o adaptación de las ya existentes a estas nuevas herramientas.
En el presente trabajo se presenta la evolución de una práctica que ha mantenido los conceptos teóricos pero que se ha adaptado a los nuevos avances tecnológicos, que en principio, no diferirán mucho, de los que utilizarán los futuros ingenieros.

1. Introducción

En las enseñanzas de Ingeniería Informática, la realización de prácticas que impliquen el estudio de componentes de un computador requiere, en general, la utilización de una herramienta de simulación digital. El Software de simulación se ha convertido una herramienta de importancia creciente ya que su carácter visual ayuda y refuerza la comprensión de los conceptos teóricos previamente tratados en clases magistrales y seminarios, y como consecuencia ejerce de elemento catalizador despertando en el alumno el interés por la materia.
Estas características se hacen especialmente visibles cuando dichas herramientas han sido diseñadas con carácter didáctico. Sin embargo, las herramientas didácticas tienen como mayor inconveniente el distanciamiento de la realidad con respecto a otras herramientas con características profesionales de elevado coste utilizadas en las empresas.
Con al avance tecnológico y la incorporación masiva del PC como elemento indispensable en la realización de las prácticas de laboratorio en carreras de Ingeniería, las empresas de Software profesional ofrecen versiones, que además de compartir características con sus "hermanas profesionales", tienen características educativas a un bajo coste que las hacen especialmente atractivas para su utilización docente.
La selección de la herramienta de simulación no suele ser tarea fácil, ya que se tienen que dar una serie de circunstancias para que su uso reporte los beneficios que de ellas se esperan. Asimismo dicho Software debe ser adecuado al nivel del conocimiento y preparación del alumno, así como a los equipos disponibles. Por otra parte, si se considera que el ciclo de preparación y ajuste de prácticas de laboratorio suele ser de varios años, una elección equivocada nos retrasaría de forma importante.
En este trabajo, los autores presentan la evolución en el uso de dos herramientas de simulación diferentes en las asignaturas de Estructuras de Computadores, de las titulaciones de Ingeniería Informática de la Universidad Politécnica de

Valencia. En particular se contextualiza una práctica de laboratorio en el entorno de dos planes de estudio diferentes. Los planes del 94 y los planes de 2001.

En los planes del 94 se seleccionó como herramienta de simulación una herramienta educativa denominada *Computer Assited Simulation for Circuits Análisis & Design* (CASCAD) [1], con características distantes de herramientas profesionales, pero que se adaptaba bastante bien no solamente a la formación de los alumnos sino también a la potencia de los equipos disponibles.

En los planes de 2001 se ha optado por una versión educativa de una herramienta profesional denominada XILINX [2] que presenta características interesantes no solamente para asignaturas de primer curso sino también para asignaturas específicas de cursos avanzados. Aunque dicha herramienta se encuentra actualmente en fase de implantación, hay un compromiso entre los profesores por ponerla en marcha durante este curso e ir utilizándola en las sucesivas asignaturas.

El presente trabajo se organiza como sigue. En el apartado 2 se presentan los contenidos teóricos de la práctica ejemplo dedicada a la implementación de un banco de registros. A continuación en los apartados 3 y 4 se presenta brevemente la implementación de dicha práctica utilizando el CASCAD y el XILINX respectivamente. El trabajo termina comparando ambas herramientas y presentando, en base a la experiencia de los autores, algunas de las conclusiones del trabajo.

2. Práctica ejemplo: el banco de registros

La práctica sobre la que nos centraremos en este trabajo es la implementación de un banco de registros. Hemos seleccionado esta práctica no solamente por que su diseño requiere de conceptos previamente tratados en temas de diseño digital tales como los Multiplexores y Decodificadores, sino además porque será un elemento de especial importancia en el diseño de ruta de datos y unidad de control del procesador que se utiliza en todas las enseñanzas relacionadas con la Estructura del Computador.

Un banco de registros es un circuito que proporciona un conjunto de registros, a los cuales se puede acceder para modificar el contenido de cualquiera de ellos, operación que recibe el nombre de escritura, o para averiguar el estado de los biestables de un determinado registro. Esta última operación recibe el nombre de lectura.

Estas dos operaciones se pueden implementar de distintas formas, a continuación se detalla una posible implementación, Para más información se puede consultar [4].

2.1. Operación de escritura

La Figura 1 ilustra un puerto de escritura en un banco de registros, utilizando un decodificador. Para escribir en un registro, se envía el dato de *n* bits que se desea escribir, a todos los registros que constituyen el banco, por un conjunto de *n* líneas que constituyen el bus de datos.

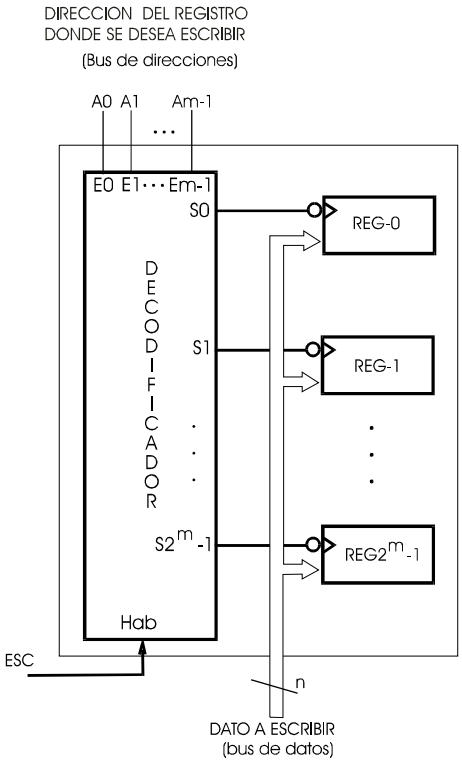


Figura 1. Implementación de un circuito de escritura en un banco de registros mediante un decodificador

Para seleccionar en qué registro se desea escribir, se utiliza un decodificador binario de *m* a 2^m. Se utilizan *m* líneas para indicar la dirección del registro *A=(Am-1Am-2...A1A0)* (número de registro) sobre el que se desea escribir. Estas *m* líneas constituyen el bus de direcciones, y se conectarán a las *m* entradas del decodificador. Estas líneas de entrada activarán únicamente una de las 2^m salidas del decodificador.
Como cada una de las salidas está conectada a un registro, la salida activa del decodificador habilitará el registro sobre el que se desea escribir. En estos momentos, los biestables del registro cambiarán su contenido si las líneas del bus de datos conectadas a sus entradas así lo indican. Obsérvese que para que esto sea factible, el decodificador debe estar activo; por lo tanto la señal de habilitación actúa como una señal de control que cuando está activa habilita la escritura.

2.1. Operación de lectura

En la Figura 2 se ha añadido un multiplexor que actúa como puerto de lectura sobre el banco de registros.
Además, se han representado las *m* líneas que constituyen el bus de direcciones mediante una sola línea, relativamente ancha, que representa el bus. Las salidas de los registros se encuentran permanentemente enviando su estado al multiplexor. Para leer el contenido de un determinado registro, se enviará el número de registro que se desea leer por el bus de direcciones, cuyas líneas se conectarán a las entradas de selección del multiplexor. En estos momentos, si el multiplexor se encuentra habilitado, las *n* salidas del multiplexor de *n* bits (*n* multiplexores de un bit), contendrán el dato que se encontraba en el registro.

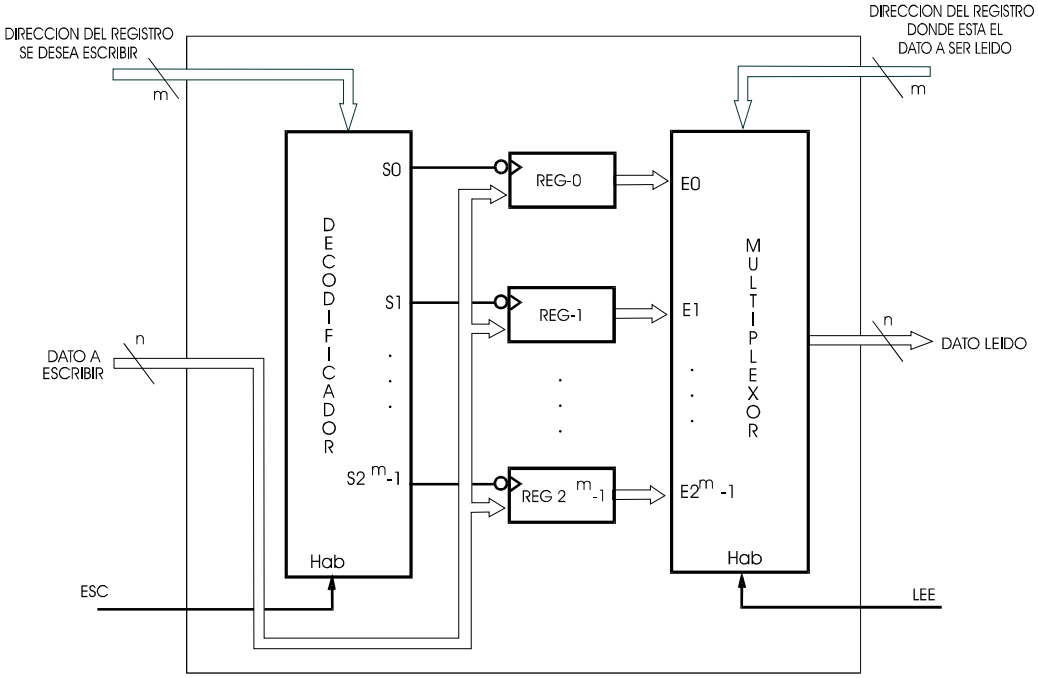


Figura 2. Implementación de un banco de registros utilizando un decodificador como puerto de escritura y un multiplexor como puerto de lectura

La ruta de datos del procesador que posteriormente estudiará el alumno, requerirá de un banco de registros con un puerto de escritura y dos puertos de lectura. Para añadir otro puerto de lectura al esquema representado en la Figura 2 simplemente habrá que conectar las salidas de los registros a otro multiplexor, que actuará como segundo puerto. La Figura 3 muestra las entradas y salidas del banco de registros a implementar.

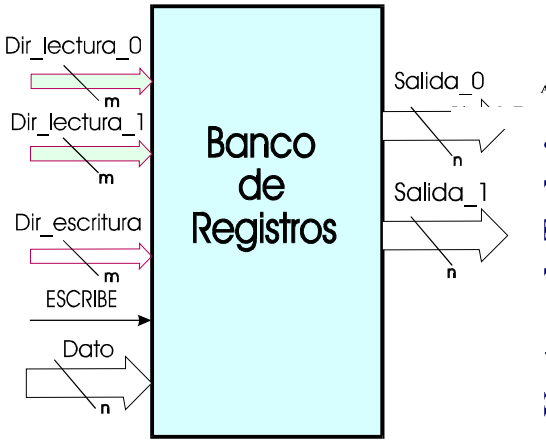


Figura 3. Entradas y salidas de un banco de registros, con un puerto de escritura y dos puertos de lectura

3. Realización de la práctica con CASCAD

3.1. Descripción del Programa

En los planes del 94 se utilizo ampliamente el programa CASCAD en las prácticas de estructura de computadores. El programa CASCAD es un entorno integrado para diseño y simulación lógica de circuitos digitales de mediana complejidad. No se trata de un paquete profesional para diseño de circuitos, sino que es una herramienta didáctica, con un tiempo mínimo estimado en unas tres horas para familiarizarse con el entorno. En esta fase de aprendizaje se planificaban una serie de ejercicios diseñados por el profesor y que permitían ir conociendo el programa mientras se construían elementos útiles para prácticas posteriores.

Esta práctica cumplía una doble función, por una parte servía para introducir al alumno en el conocimiento de la herramienta, y además se implementaban circuitos que posteriormente se utilizaban en la construcción del procesador.

Los requerimientos Hardware del programa son mínimos, aunque hace algunos años estos eran un *handicap,* hoy en día han pasado a un segundo plano teniendo en cuenta las características de los equipos de laboratorio. Por otra parte su ejecución bajo entorno DOS, y su no disponibilidad en licencia libre representan algunos de sus mayores inconvenientes. La Figura 4 muestra el entorno de trabajo de la herramienta CASCAD.

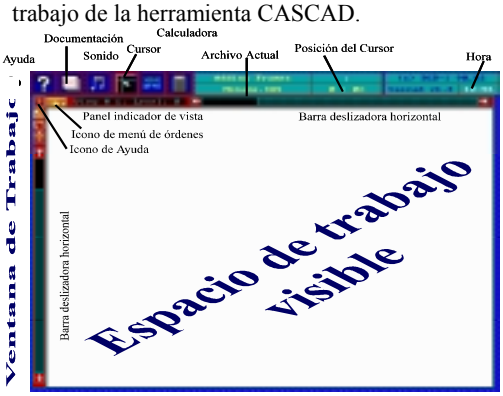


Figura 4. Pantalla principal de CASCAD

Dado que CASCAD es un programa que se ejecuta en entorno MSDOS, para la selección de las distintas funciones del entorno, CASCAD utiliza un conjunto de menús organizados en secuencia. Para hacer aparecer los menús sobre la ventana de trabajo basta pulsar el botón derecho del ratón. El primer menú que aparece es el menú principal *(main menu)*. La selección de cualquiera de sus opciones hace aparecer un segundo menú para dicha opción y así sucesivamente formando una cadena, como se muestra a continuación.

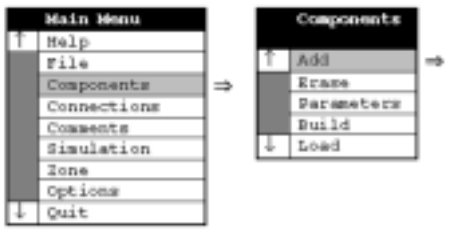


Figura 5. Secuencia de menús en CASCAD

3.2. Implementación del Banco de Registros

Como el CASCAD no ofrece ningún banco de registros como componente, se deberá construir conectando entre si los diferentes elementos necesarios. Una vez construido, el CASCAD ofrece la posibilidad de encapsular el circuito realizado para su posterior utilización en circuitos de mayor complejidad.

Los pasos generales para la implementación de cualquier circuito en general y del banco de registros en particular son los siguientes: 1) Identificación de los componentes necesarios, 2) Realización de las conexiones; 4) Simulación y circuito de prueba del circuito. Teniendo en cuenta la descripción del banco de registros del apartado 2, la Figura 6 muestra una posible implementación de un banco de registros de ocho registros de 16 bits cada uno.

Como se puede observar, la operación de lectura se ha implementado con dos Multiplexores. La operación de escritura se ha implementado se ha realizado utilizando un Demultiplexor debido a que el CASCAD no incorpora Decodificadores. El alumno conoce la equivalencia existente entre Decodificadores y Demultiplexores.

3.3. Circuito de prueba

De cara a la simulación y prueba del banco de registros, la Figura 7 muestra el circuito de prueba que se debe implementar utilizando una Unidad Aritmético Lógica o ALU. Se puede observar como el banco de registros se ha encapsulado en un nuevo componente denominado banco de registros que se puede utilizar en un nuevo circuito. Esta característica de encapsular componentes es de especial importancia de cara a implementar librerías de componentes específicos.

Finalmente hay que comentar que para un aprovechamiento de las sesiones de prácticas, debe realizarse una planificación concienzuda que requiere de un trabajo previo por parte de los profesores de la asignatura. En este sentido a los alumnos de primer curso se les suministra un proyecto inicial de partida en base al cual trabajan hasta obtener la versión definitiva.

Inicialmente se pedía al alumno que realizase todo el diseño completo; pero la experiencia nos demuestra que no es un buen método didáctico, ya que producía una pronta desmotivación en aquellos alumnos que no llegaban a obtener los resultados esperados.

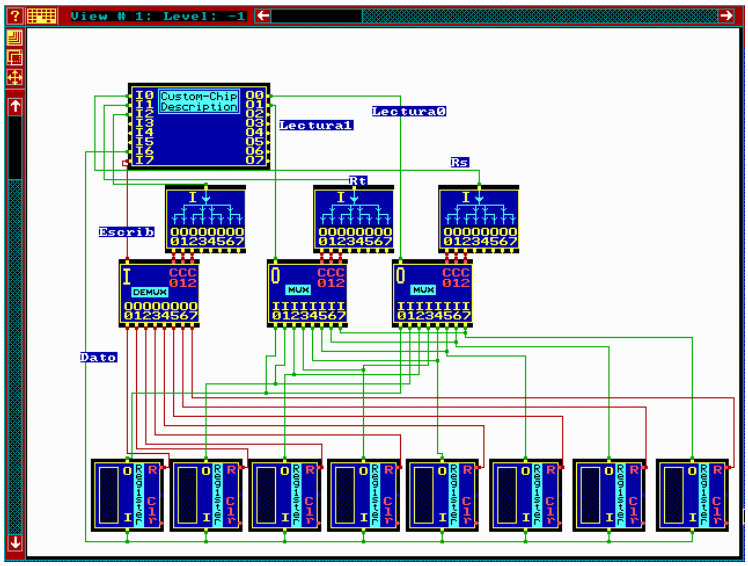


Figura 6. Implementación interna de un banco de ocho registros en CASCAD

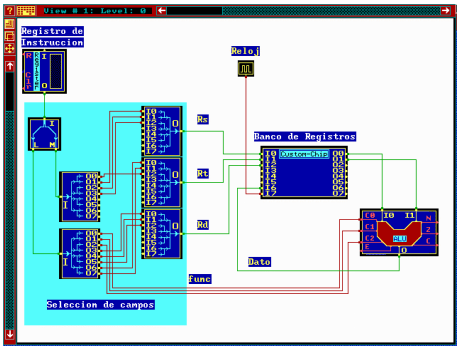


Figura 7. Circuito de prueba del banco de registros

4. Realización de la práctica con Xilinx

Debido a la importancia de la práctica, en nuestro bloque de prácticas, consideramos necesario que la práctica continúe realizándose. Sin embargo, nos encontramos en la necesidad de adaptarnos a las nuevas herramientas que ofrecen una mayor presencia y funcionalidad.

4.1. Descripción del Programa

El simulador XILINX *Foundation* 2.1 de la compañía Xilinx es un entorno profesional integrado de diseño, simulación y síntesis de circuitos y Hardware digital. Esta herramienta incorpora el lenguaje estándar de descripción hardware VHDL, permitiendo tanto el diseño de circuitos y sistemas digitales completos sin depender de un fabricante específico, como su implementación directa en Hardware.

La utilización de una herramienta de tales características ha de planificarse de forma progresiva. En los planes de estudio de 2001, hay un compromiso de profesores de distintas asignaturas, para utilizar esta herramienta progresivamente, desde primeros cursos, hasta cursos avanzados de diseño VLSI, llegando incluso a proponer Proyectos Finales de Carrera donde se lleguen a utilizar sus características más avanzadas. De esta forma, el tiempo necesario para conocer y utilizar las características de la herramienta se va distribuyendo y amortizando en diferentes cursos, acorde con la formación progresiva del alumno. La Figura 8, muestra la pantalla principal de la herramienta. Para la implementación del banco de registros, solamente utilizaremos el diseño esquemático y la simulación funcional, sin entrar en detalles específicos de implementación propios de cursos avanzados.

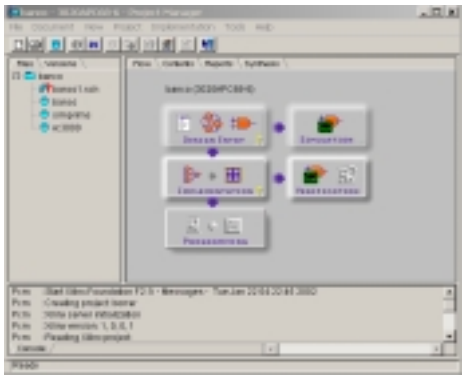


Figura 8. Pantalla principal de XILINX

4.2. Implementación del Banco de Registros

Al igual que sucedía con el programa CASCAD, el XILINX no incorpora un banco de registros. El primer paso es identificar los componentes necesarios en las librerías de XILINX. Para aquellos elementos que no se encuentren disponibles en las librerías por defecto, XILINX, incorpora una función denominada creación de módulos **logiBlox** que permite crear e incorporar como elementos de librería componentes específicos a partir de componentes genéricos. Para el diseño del banco de registros necesitaremos 1 decodificador de 3 a 8 para implementar la operación de escritura y 2 multiplexores de 8 entradas y 1 salida todas con un ancho de 16 bits. Tanto el decodificador como los multiplexores, se han creado utilizando módulos **logiBlox.** La Figura 9 muestra la creación de un Multiplexor de 8 a 1 con un ancho de bus de 16 bits a partir de un componente genérico Multiplexor.

La Figura 10 muestra la implementación del banco de registros de 8 registros de 16 bits, utilizando dos Multiplexores y el Decodificador implementados como **logiBlox,** y 8 registros de 16 bits que incorpora una de las librerías de XILINX. Para realizar diseños jerárquico, el XILINX permite la creación de nuevos componentes a partir de diseños

realizados. Una vez generado el nuevo componente, este se encuentra disponible como nuevo elemento de la librería del proyecto, pudiéndose utilizar en futuros diseños.

Figura 9. Creación de un multiplexor 8 a 1 de 16 bits

Para generar un nuevo componente necesitamos crear un símbolo a partir de un diseño ya existente. El símbolo consistirá en un nuevo componente donde el usuario sólo verá un bloque con el mismo número de entradas y de salidas que el esquema original, que englobará a éste y realizará su misma función. La Figura 11 muestra el editor de símbolos que incorpora XILINX.

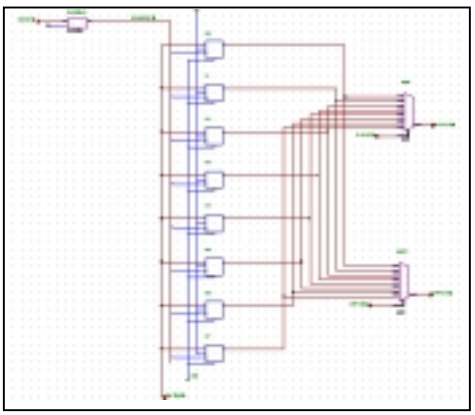


Figura 10. Implementación interna de un banco de ocho registros en XILINX.

La Figura 12 muestra el componente banco de registros. Como se puede observar, el editor de símbolos permite establecer una equivalencia visual entre el interfaz del componente generado y el interfaz teórico que se presento en la Figura 3 del apartado 2.

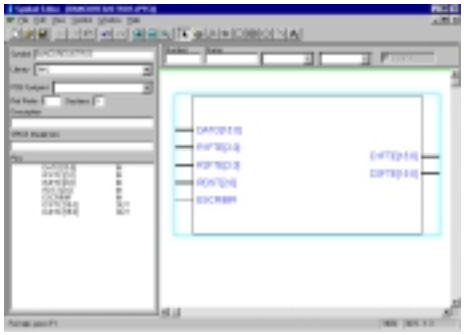


Figura 11. Editor de Símbolos de XILINX

Esta característica facilita notablemente la comprensión de los circuitos implementados, especialmente cuando la envergadura de un proyecto es considerable.

Figura 12. Componente Banco de Registros.

3.3. Circuito de prueba

Para comprobar el funcionamiento del banco de registros, El alumno dispone de un circuito de prueba similar al implementado con CASCAD. Una vez que el alumno ha comprobado el funcionamiento del circuito, se le proporciona un circuito de mayor dificultad como el mostrado en la Figura 13 y se le plantean una serie de cuestiones teórico/prácticas. El objetivo que se persigue es analizar el nivel de comprensión del funcionamiento del banco de registros cuando este se conecta con otros elementos, como son contadores y Unidades Aritmético Lógicas, elementos que se utilizarán posteriormente en el diseño de la ruta de datos de un procesador básico

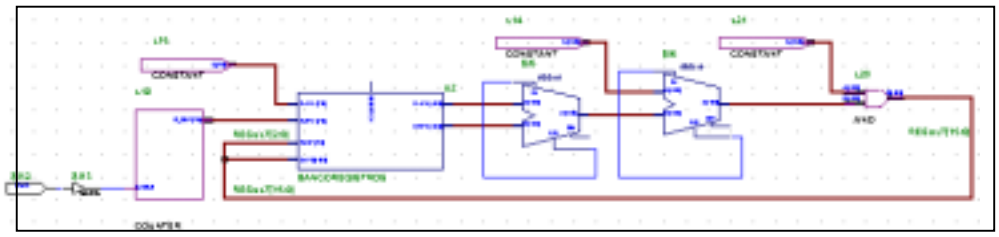


Figura 13. Circuito de prueba del banco de registros

5. Comparativa entre herramientas

En este apartado estableceremos una comparativa de ambas herramientas CASCAD (C) y XILINX (Xi), estableciendo un conjunto de características que se deben tener en cuenta de cara a seleccionar una u otra como herramienta de simulación de sistemas digitales.

- **Sistema Operativo**: (C) MSDOS. (Xi) Windows.
- **Requerimientos Procesador y RAM**: (C) 386, 1Mbyte. (Xi) Pentium III, 64Mbytes.
- **Espacio en disco**: (C) disquetes, (Xi) CDs.
- **Entorno del programa**: (C) Sencillo pero poco flexible. (Xi) Sencillo y flexible.
- **Comprobación y Simulación**: (C) Sencillo y poco flexible. (Xi) Complejo pero flexible.
- **Generación de nuevos componentes**: (C) Restringido. (Xi) Avanzado
- **Utilización en cursos avanzados**: (C) Restringido. (Xi) Aconsejable.
- **Entorno de uso**: (C) Educativo. (Xi) Educativo y profesional.

6. Conclusiones

En este trabajo se ha discutido el impacto que las nuevas herramientas de simulación han tenido y continúan teniendo sobre las prácticas docentes universitarias. El estudio se ha particularizado para la docencia en estructuras de computadores, aunque se puede generalizar a cualquier docencia universitaria.

La aparición de las nuevas herramientas, con una funcionalidad casi profesional, nos hace cuestionarnos si es adecuado continuar con las viejas prácticas y el nuevo software, o si sería preferible plantearnos la aparición de nuevas prácticas que con el software anterior no eran factibles. Quizá la solución sea un mezcla de ambas alternativas, ya que muchas de las prácticas que vienen haciéndose durante años e incluso lustros, han sufrido las críticas y los retoques de muchos profesores, así como de varias generaciones de estudiantes. Algunas de estas prácticas tienen un valor añadido especial, sin embargo, es necesario que se adapten a las nuevas tecnologías si queremos mantener el prestigio y la calidad de las mismas.

En este trabajo no sólo nos hemos centrado en las cuestiones didácticas y metodológicas, sino que también se ha realizado un caso de estudio. A modo de ejemplo, se ha presentado la práctica del "Banco de Registros", que llevamos haciendo durante años y que es la base de prácticas más complejas.

Finalmente recalcar, que creemos que la adaptación de nuevas prácticas es necesario tanto desde el punto de vista del alumno como del profesor. Desde el punto de vista del profesor porque siente satisfacción al realizar adecuadamente su función docente, y desde el punto de vista del alumno porque trabaja con una herramienta con apariencia profesional.

Referencias

[1] Computer Assited Simulation for Circuits Análisis & Design (CASCAD), Manual de referencia, EduSoft.

[2] Xilinx Foundation Series 2.1i Student Edtion software, <URL: http://www.xilinx.com/>.

[2] David. E. Vandes, Logic Design with XILINX Foundation 2.1i.

[4] J. Sahuquillo, H. Hassan, L. Lemus, J. Molero, R. Ors, F. Rodriguez, "Introducción a los Computadores", Servicio de publicaciones de la UPV, 1997.

Desarrollo de una tarjeta de adquisición de datos para la docencia de Sistemas Periféricos

Germán Galeano Gil Francisco Fernández de Vega

Dpto. de Informática

Universidad de Extremadura

Centro Universitario de Mérida

06800 Mérida

e-mail: ggaleano @unex.es e-mail: fcofdez @unex.es

http:// atc.unex.es /ggaleano http:// atc.unex.es /fcofdez

Resumen

En este trabajo presentamos una tarjeta de adquisición de datos utilizada para la docencia del diseño y control de sistemas periféricos en asignaturas de Ingeniería Telemática.

Exponemos además las experiencias de varios profesores que imparten docencia en una unidad temática formada por asignaturas del área de arquitectura de computadores en la titulación de Ingeniería Técnica en Telecomunicaciones.

1. Antecedentes

En el *Departamento de Informática* del *Centro Universitario de Mérida* [2], perteneciente a la Universidad de Extremadura [1], se ha constituido un grupo de docencia formado por profesores que imparten clases en varias asignaturas de la titulación en Ingeniería Telemática, dentro del área de *Arquitectura y Tecnología de Computadores* [3].

El principal problema que presentan los alumnos de esa titulación es que tienen dificultades para experimentar con periféricos de E/S avanzados, en especial aquellos que toman datos del exterior realizando una conversión digital-analógica, ya que en los laboratorios sólo disponen de PCs con una configuración muy básica. Por todo ello, las prácticas de la asignatura *Sistemas Electrónicos Digitales* [4] se limitaban a explicar el funcionamiento de los periféricos y realizar un par de prácticas en las que se manejan a muy bajo nivel el teclado, el sistema de vídeo y el ratón. Algo similar ocurría con la asignatura de Sistemas Operativos.

Surgió así la idea de construir una tarjeta de propósito general que sirviera para la entrada y salida de datos y que fuera utilizada como elemento básico para la explicación del funcionamiento de los sistemas periféricos.

Para el diseño de esta tarjeta se partió de un trabajo previo de F. Fernández [5].

2. Planteamiento

El objetivo de este trabajo es construir, de una forma sencilla y económica, una tarjeta de adquisición y conversión analógico-digital. Es decir, un circuito que convierta una señal digital del ordenador en una analógica, y viceversa, tome una analógica del exterior y la convierta en digital.

Con esta tarjeta se persiguen seis objetivos:

1- Motivar al alumnado realizando prácticas sobre un dispositivo que ha sido construido enteramente por ellos.

2- Estudiar la teoría subyacente en el diseño hardware de periféricos de entrada y salida.

3- Estudiar en profundidad el diseño software del control de estos periféricos y la

comunicación de éstos con el procesador y la memoria.

4- Construir una plataforma que sirva de base para el desarrollo de futuros trabajos más avanzados, tales como el desarrollo de una tarjeta de adquisición de datos más potente, o construcción de tarjetas de adquisición y control específicas para problemas concretos.

5- Utilizar los datos obtenidos para realizar estudios explicados en otras asignaturas, como aplicar la transformada de Fourier para obtener las frecuencias principales, etc.

6- Construir un dispositivo con el menor coste económico, pero que sea sencillo y funcional.

La tarjeta que se presenta sirve para la digitalización y adquisición de datos cualesquiera, aunque para ahorrar costes y hacer más sencillo el diseño hardware, se le ha limitado a una resolución de 5 bits, resolución que es suficiente para nuestros propósitos.

La tarjeta desarrollada ha sido utilizada como dispositivo de entrada y salida de sonido. Así, la tarjeta permite, en general, realizar dos tareas básicas: emisión y captura de sonidos.

Además, se puede dotar al sistema de un software básico que acompañe a la tarjeta y que permita el control de ésta sin tener conocimientos de cómo es el hardware de la misma. Este software permitiría realizar operaciones como editar la señal de sonido obtenida y su manipulación para añadir efectos, como distorsión y eco, o cálculo del Espectro de Fourier [7].

En la elaboración del software que controla la tarjeta, los alumnos deben tener en cuenta las características relacionadas con el hardware del PC: repertorio básico de instrucciones del Intel 8086/8088, mapa de memoria, modos de direccionamiento, pila, interrupciones, rutinas de servicio, etc, así como el funcionamiento básico del sistema operativo (llamadas al sistema, entradas al kernel, etc).

3. Desarrollo

El desarrollo de la tarjeta se realiza a tres niveles:

1. Desarrollo hardware: consiste en construir el circuito mediante la sobreimpresión de las pistas y el soldado de los componentes.
2. Desarrollo de un driver muy básico, mediante interrupciones software (DOS) o uno más avanzado mediante manejadores de dispositivo (Linux). Constituiría la capa software más básica.
3. Desarrollo del programa de control de la tarjeta y manipulación de los datos que incremente las presentaciones de la tarjeta, basándose en la capa de software anterior.

3.1. Desarrollo hardware

La principal función de la tarjeta construida es la de dedicarse exclusivamente a realizar la conversión digital-analógica, dejando el resto de las tareas bajo el control de la CPU del micro.

A grandes rasgos, el esquema básico de la tarjeta de sonido implementada es el mostrado en la figura 1.

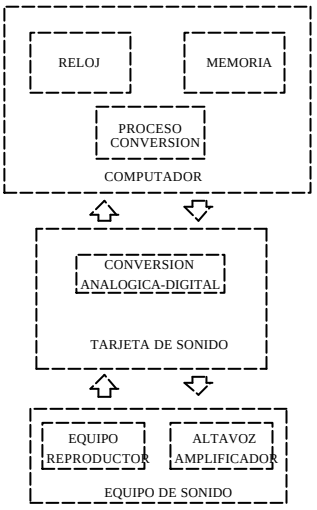


Figura 1. Esquema de la tarjeta

En este esquema, se observa que el computador se encarga de gestionar la memoria, la frecuencia de muestreo y la comunicación con la tarjeta, siendo ésta última la encargada de realizar las conversiones, tanto digital-análogica como analógica-digital, y conexión con el exterior. En el caso de la tarjeta de sonido que se ha construido, esa conexión se ha realizado con un altavoz con amplificador en la salida y con un micrófono en la entrada, permitiendo el conjunto la captura y emisión de sonido.

Teniendo en cuenta las tareas encomendadas a la tarjeta (captura -digitalización de sonido- y emisión de sonido), la tarjeta consta de dos módulos diferentes que se relacionan:

- Módulo conversión Analógica-Digital: tiene como misión capturar señales de sonido (analógicas) procedentes de un equipo de sonido, convertirlas en señales digitales manipulables por el ordenador y transferírselas.
- Módulo conversión Digital-Analógica: se encarga de recoger señales de sonido digital, procedentes del computador, transformarlas en señales analógicas y enviarlas al amplificador, para que puedan ser escuchadas por el usuario

Con todo esto, el esquema resultante una vez analizado el sistema es el mostrado en la figura 2.

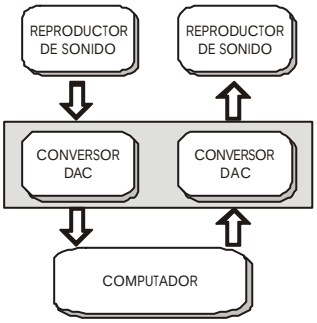


Figura 2. Esquema de bloques del sistema

Para realizar la conversión Analógica/Digital se ha utilizado un DAC 0800 de Motorola y comparadores UA741, que aunque no son de muy buena calidad, son muy económicos y ofrecen un buen resultado.

La tarjeta también permite capturar señales sonoras del exterior. Para ello se ha utilizado un método muy sencillo y económico que consiste en lo siguiente: el DAC0800 es capaz de transformar una muestra de 5 bits (un valor numérico digital) en un valor analógico de voltaje o intensidad. Así, todo valor entre 0 y 32 (valores posible con 5 bits) tiene asignado su correspondiente voltaje, que se encuentra entre un valor máximo y mínimo. Utilizando un comparador adecuado, sería posible determinar si la señal de entrada (con un voltaje que debe estar entre el valor máximo y mínimo con el que trabaja el DAC) es mayor o menor que cualquier señal procedente del ordenador.

El funcionamiento es pues el siguiente: cuando se recibe una señal de entrada, el ordenador envía a la tarjeta una secuencia de señales digitales que serán comparadas con los comparadores UA741 con la señal de entrada. De este modo, la tarjeta comprueba continuamente la señal de entrada con las sucesivas señales tipo que envía el ordenador. Cada resultado de las comparaciones es enviado por la tarjeta al ordenador. Cuando la comparación sea la adecuada, el proceso termina y el ordenador conoce la codificación digital de la señal de entrada.

El diagrama de la figura 3 muestra la iteración entre DAC0800 y el comparador:

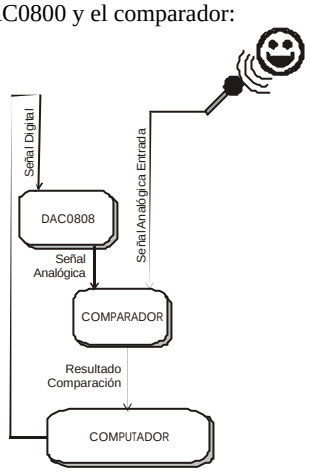


Figura 3. Proceso de conversión Analógico-Digital

La figura 4 muestra el circuito completo simplificado de la tarjeta. Como se puede apreciar, el circuito es muy compacto y puede ser implementado sin problemas sobre una superficie

de 8x8 cm. La figura 5 muestra el aspecto final del dispositivo.

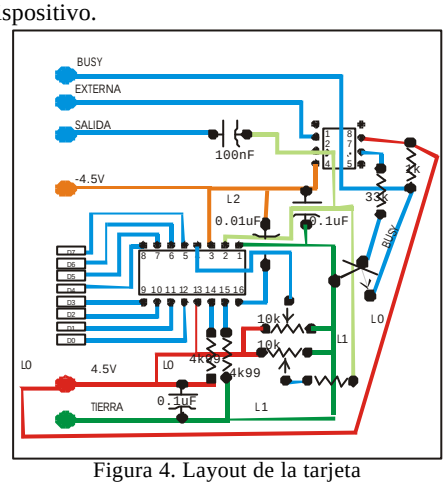


Figura 4. Layout de la tarjeta

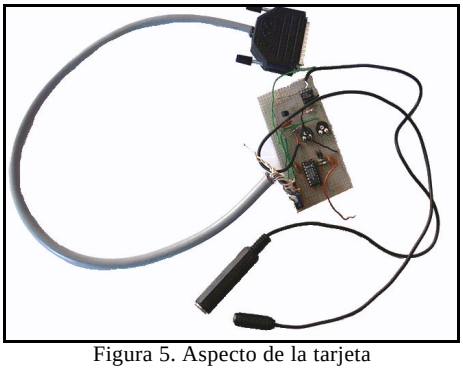


Figura 5. Aspecto de la tarjeta

3.3. Desarrollo software

Como acabamos de ver, la única misión de la tarjeta es realizar la conversión analógica-digital y viceversa. Posteriormente, la información producida es enviada al ordenador a través del puerto paralelo para que la CPU la procese.

Así pues, el procesador ha de encargarse de la siguientes tareas básicas:

- Comunicación con la Tarjeta de Sonido.
- Gestión de Memoria Dinámica en donde almacenar los resultados que se van generando.
- Gestión de Tiempos.

Será necesario establecer, tal como indica el primer punto, un modo de comunicación entre computador y tarjeta, en tiempo real, ya que la tarjeta no dispone de buffers de almacenamiento.

La comunicación se darán en dos sentidos, del ordenador a la tarjeta (emisión), y de la tarjeta al ordenador (captura).

En cuanto a la Gestión de Memoria Dinámica, el computador habrá de controlar la memoria disponible e ir almacenando en ellas las muestras enviadas por la tarjeta y al ritmo que la frecuencia de muestreo vaya marcando.

Será también el Computador el encargado de fijar, utilizando el reloj programable, la frecuencia de muestreo y emisión. Los componentes de la tarjeta deberán se capaces de trabajar a la frecuencia fijada por el micro.

4. Secuenciación docente

En un principio, la construcción y manejo de la tarjeta se realiza únicamente en las asignaturas de Sistemas Electrónicos Digitales y Sistemas Operativos, de la titulación de Ingeniería en Telecomunicaciones, rama Telemática.

Como se puede apreciar en la figura 6, la asignatura de Sistemas Electrónicos Digitales se imparte en el segundo año y se basa en los conocimientos aprendidos en Electrónica e Introducción de computadores, impartidas ambas en el primer año. Además, esta asignatura, junto con la de Estructura de Computadores, sirven de base teórica para explicar el funcionamiento de los Sistemas Opertivos (tercer año).

Par la creación de esta tarjeta y del sistema software asociado, los profesores de las asignaturas de Sistemas Electrónicos Digitales y Sistemas Operativos se pusieron de acuerdo y se repartieron las tareas.

En principio, se consideraron los siguientes objetivos:

- El objetivo de la asignatura de Sistemas Electrónicos Digitales es mostrar al alumno el funcionamiento intrínseco de un periférico de E/S y cómo se puede manejar a través de los puertos de E/S

- El objetivo de la asignatura de Sistemas Operativos es mostrar al alumno cómo el sistema operativo enmascara al periférico y dota a las aplicaciones de mecanismos para manejar el dispositivo de forma más sencilla

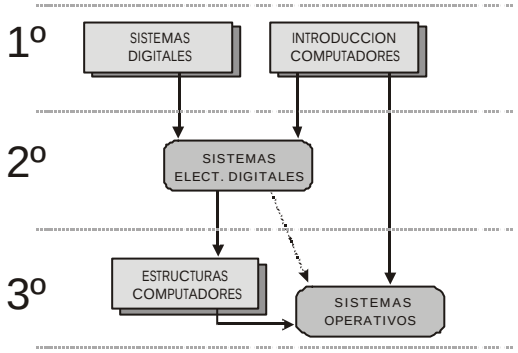


Figura 6. Asignaturas relacionadas con Sistemas Electrónicos Digitales y con Sistemas Operativos.

Partiendo de esos objetivos, se realizó un reparto de tareas y resultó entonces la unidad didáctica que se muestra en la tabla 1.

Como se aprecia en esa tabla, en la asignatura de Sistemas Electrónicos Digitales se introduce a los alumnos en el funcionamiento teórico del dispositivo y se implementa físicamente la tarjeta. Además se realizan varios programas de tests y se construye un pequeño controlador, muy sencillo y elemental, en ensamblador, simulando el funcionamiento de los servicios del sistema operativo DOS.

Sin embargo, como ya se ha comentado anteriormente, en la asignatura de Sistemas Operativos, el enfoque varía. Aquí se trata de enseñar al alumno cómo el Sistema Operativo hace de intermediario entre los programas y el hardware. Por ello, en esta asignatura se construye una librería y un dispositivo que permite controlar la tarjeta.

5. Conclusión

Esta experiencia docente ha tenido muy buenos resultados ya que los alumnos se encuentran muy motivados desde el comienzo en la realización de las prácticas.

Además, es para ellos muy gratificante observar cómo un dispositivo construido por ellos mismos funciona y responde a las órdenes enviadas por el ordenador.

Por otro lado, los alumnos asimilan mucho mejor los conceptos relativos al funcionamiento de los periféricos y cómo los programas y el sistema operativo interactúa con ellos, y se sienten muy motivados.

Los autores de este trabajo consideramos que es muy importante motivar al alumnado mediante la realización de trabajos teórico-prácticos como éste, que además sirven de hilo conducto para unir los conocimientos aprendidos en diferentes asignaturas.

Sesión	Curso	Asignatura	Objetivos	Horas
1	2º	S.E.D.	- Explicar el funcionamiento eléctrico de la tarjeta - Construirla físicamente y probarla con un programa de test	2
2	2º	S.E.D.	- Explicar el funcionamiento de la tarjeta desde el punto de vista software - Construcción de un pequeño programa de test en ensamblador	2
3	2º	S.E.D.	- Construcción de un nuevo servicio del sistema operativo DOS mediante interrupción software - Testeo del servicio con un programa realizado en C	6
4	3º	S.O.	- Testeo de la tarjeta mediante un programa que haga llamadas al sistema en LINUX	2
5	3º	S.O.	- Construcción de una librería y un fichero de dispositivo que permita manipular la tarjeta y enviar / recibir datos de ella	10

Tabla 1. Secuenciación de la implementación en sesiones prácticas de la tarjeta de sonido y su software asociado

Referencias

[1] Universidad de Extremadura. http:// www.unex.es

[2] Centro Universitario de Mérida. http:// cum.unex.es

[3] Area de Arquitectura y Tecnología de Computadores de la Universidad de Extremadura. http:// atc.unex.es

[4] http:// cum.unex.es/profesores/ggaleano/sed

[5] F. Fernández: “Sistema de adquisición de señales de sonido”. Proyecto Fin de Carrera Licenciatura en Informática. Universidad de Sevilla. 1994.

[6] Sistemas Controlados por Computador. Korl Aström. Paraninfo

[7] Introductory digital signal porcessing with computer applications. Paul A. Linn, Wolfgang Fuerst. Ed. John Wiley and sons

[8] Integrated Circuits Data Book. Ed. Burn Brown

Bases de datos

Análisis del tratamiento de las bases de datos en los curricula internacionales: comparación con el currículum de Blesa et al. (1999)

Mario Piattini, Coral Calero, Francisco Ruiz
Grupo ALARCOS
Escuela Superior de Informática
Paseo de la Universidad, 4
13071 Ciudad Real
Mario.Piattini, Coral.Calero, Francisco.RuizG@uclm.es

1. Introducción

Las bases de datos son un elemento, en bastantes casos el núcleo, de un número cada vez mayor de sistemas y aplicaciones informáticos, en efecto, el mercado[1] mundial de las bases de datos en 2001 rondaba los 16.000 millones de dólares y se espera que este mercado aumente alrededor del 18% en el año 2004 ([5]). En España, según la publicación de SEDISI (2001), en el año 2000 el mercado interior bruto para el software fue de 256.571 M Pta. de los que corresponden a software de bases de datos 41.480 con un incremento del 18,4% respecto al año anterior.

Esta circunstancia queda reflejada, tanto en los diferentes currículos propuestos por las organizaciones internacionales y universidades extranjeras de prestigio, como en la demanda actual de conocimientos exigidos por parte de las diferentes empresas internacionales y nacionales. También queda patente la importancia de esta asignatura en los diferentes Planes de Estudio de las distintas Escuelas y Facultades de Informática. El hecho de que se hayan constituido las "Jornadas en Investigación y Docencia en Bases de Datos" (actualmente integradas en las "Jornadas en Ingeniería del Software y Bases de Datos") y de que gocen de una amplia aceptación, también demuestra el interés por las bases de datos.

Fue precisamente en el marco de estas jornadas donde se elaboró un currículo para bases de datos, que tras algunas revisiones fue publicado en Novática hace tres años [1]. En este trabajo comparamos este currículum con los principales currícula internacionales más recientes: ACM/IEEE CS 2001, IRMA/DAMA 2000, ACM/AIS MSIS 2000, ISCC´99, IFIP/UNESCO ICF-2000 y con algunas otras propuestas que pueden afectar a la docencia en bases de datos, como el SWEBOK o las que tratan la calidad de datos o bases de datos y web.

2. El currículum de Blesa et al. (1999)

A raíz de las II Jornadas de Investigación y Docencia en Bases de Datos, celebradas en Julio del 97 (JIDBD'97), se creó un grupo de trabajo liderado por Pedro Blesa de la UPV con el objetivo de definir los contenidos de Bases de Datos en los estudios universitarios de informática. En Novática se publicó hace más de tres años ([1]) la versión definitiva de la propuesta realizada por este grupo, que acordó definir las asignaturas que aparecen en la tabla 1, tanto para las Ingenierías Técnicas en Informática de Gestión (ITIG) y de Sistemas (ITIS) como para la Ingeniería Informática (II).

3. Currícula internacionales más recientes

Vamos a presentar muy brevemente las versiones más recientes de los currícula que estimamos más interesantes y prestigiosos en el campo de la informática. Tradicionalmente se han estudiado también otros como el ACM/IEEE CS 1991, ACM/AIS/AITP IS'97, UNESCO-IFIP de 1995, etc., pero consideramos que estos últimos ya han quedado algo desfasados aunque muchas de sus ideas perviven en las propuestas actuales.

[1] Estas cifras corresponden a los mercados relacional, objeto-relacional y orientado a objetos

Asignatura	Créditos	II	ITIG	ITIS
Bases de Datos 1 (BD1)	9	Obligatoria	Obligatoria	Obligatoria
Bases de Datos 2 (BD2)	6	Obligatoria	Obligatoria	Optativa
Diseño de Bases de Datos (DBD)	4,5/6	Optativa	Optativa	Optativa
Administración de Bases de Datos (ABD)	4,5/6	Optativa	Optativa	Optativa
Aplicaciones de Bases de Datos (APL)	4,5/6	Optativa	Optativa	Optativa
Bases de Datos Avanzadas (BDA)	4,5/6	Optativa	-	-

Tabla 1. Currículum para Bases de Datos propuesto por [1]

3.1. CC 2001 de ACM/IEEE

En 1998 ACM[2] y la Computer Society de IEEE[3] formaron un equipo conjunto denominado Computing Curricula 2001 (CC2001), al que se le pidió que desarrollara un conjunto de guías curriculares que "abordara los desarrollos más recientes de las tecnologías informáticas en la década pasada y que perdure la siguiente década". La comunidad educativa parece que requería que este equipo adoptara una visión más amplia de la informática ya que, entre otras cosas: "*Reducir la disciplina de la informática a componentes tradicionales limita la evolución de la disciplina ...Las otras disciplinas que comprenden la visión amplia de la informática son al menos tan importantes para el currículo académico como la ciencia de la computación tradicional*".

El informe CC2001 se encuentra dividido en cinco partes:

- Vol. I : Visión General
- Vol. II : Ciencias de la Computación
- Vol. III : Ingeniería de la Computación
- Vol. IV : Ingeniería del Software
- Vol. V : Sistemas de Información

De estos volúmenes se encuentra finalizado el correspondiente a Ciencias de la Computación (CS´2001), en cuyo cuerpo de conocimiento se distinguen 14 áreas (que representan un campo particular de la disciplina) entre las que destacamos la "Gestión de la Información (IM)" que comprende las siguientes unidades (módulos temáticos): IM1. Modelos y sistemas de información, IM2. Sistemas de bases de datos, IM3. Modelado de datos, IM4. Bases de datos relacionales, IM5. Lenguajes de consultas de bases de datos, IM6. Diseño de bases de datos relacionales, IM7. Procesamiento de transacciones, IM8. Bases de datos distribuidas, IM9. Diseño físico de bases de datos, IM10. Minería de datos, IM11. Almacenamiento y recuperación de la información, IM12. Hipertexto e hipermedia, IM13. Sistemas e información multimedia, IM14. Bibliotecas digitales

Las tres primeras unidades forman parte del núcleo básico requerido para cualquier alumno de un programa de informática, que pueden combinarse de diferentes formas. Así, por ejemplo, en el "enfoque tradicional" se encuadra el curso CS270T Bases de datos, y en el "enfoque comprimido" el curso CS262C Gestión de la información y del conocimiento, en el "enfoque basado en sistemas" el curso CS271S Gestión de la información, y en el "enfoque basado en la web", el curso CS261W Inteligencia artificial e información. Dentro de los cursos avanzados del área IM se proponen el CS370 Sistemas de bases de datos avanzados, CS371 Diseño de bases de datos, CS372 Procesamiento de transacciones, CS373 Bases de datos distribuidas y de objetos, CS374 Minería de datos, CS375 Almacenes de datos, CS376 Sistemas de información multimedia, y CS377 Bibliotecas digitales.

3.2. IRMA/DAMA 2000

Este currículo es el resultado de dos años de esfuerzo conjunto de dos asociaciones profesionales norteamericanas de gran relevancia en el área de bases de datos: IRMA[4] y DAMA[5],

[2] Association for Computer Machinery
[3] Institute of Electronic and Electrical Engineers

[4] Information Resources Management Association

que empezaron en 1998 la revisión de la edición existente anteriormente.

En este currículo se insiste en la necesidad de que los ingenieros en informática no limiten sus conocimientos a los aspectos técnicos de los SI, sino que posean una visión más global que la gestión de los datos, considerando a ésta como parte de la "Gestión de los Recursos de Información" (GRI, en siglas inglesas IRM). En este sentido, en [3] se señala que es necesario superar la visión limitada de los sistemas de información basados en ordenador compuestos sólo de hardware y software ya que los trabajadores de los departamentos de procesamiento de la información necesitan una comprensión mayor de los recursos humanos y las aplicaciones empresariales.

En este currículum se diferencian ocho cursos relacionados con la GRI: IRM1 Principios de Gestión de Recursos de Información, IRM2 Tecnología de Sistemas de Información, IRM3 Gestión de la Información y Conceptos de Algoritmos, IRM4 Almacenes de datos, minería de datos y sistemas de soporte a las decisiones, IRM5 Estructuras y Administración de Recursos de Datos, IRM6 Diseño de Implementación de la Gestión de Recursos de Información, IRM7 Tecnología de la Comunicación y Gestión de Información, IRM8 Gestión de Información Global, IRM9 Gestión de Sistemas de Información para Ejecutivos, IRM10 Temas Seleccionados en Gestión de Recursos de Información. El más directamente relacionado con bases de datos es el IRM5, que contempla: Recursos de Datos e Información, Conceptos y Aplicaciones de Bases de Datos, Conceptos de diseño de bases de datos, Utilización y aplicaciones de bases de datos, Administración y seguridad de bases de datos, Gestión y Planificación estratégica de bases de datos, Aplicaciones de la Administración de datos, Planificación de Sistemas de Información Estratégicos.

3.3. MSIS 2000 de ACM/AIS

La necesidad de revisar los curricula en SI, especialmente el de ACM de 1982 y el IS'97, lleva a la creación, en enero de 1998, de un "Comité Curricular Conjunto" (*Joint Curricular Committee -JCC-*) de ACM y AIS[6], a fin de elaborar un curriculum de Master para SI, el MSIS 2000 (publicado en *The DATABASE for Advances in Information Systems*, vol. 31, Nº 1, invierno 2000).

El modelo curricular se diseña en cuatro bloques interrelacionados: *fundamentos de SI*, tanto informáticos (tecnología hardware y software, programación, datos y estructuras de objetos) como de gestión (contabilidad financiera, marketing, comportamiento organizacional), *núcleo de SI* (un conjunto de cursos exigidos a todos los graduados: gestión de datos, análisis, modelado y diseño, comunicaciones de datos y redes, gestión de proyectos y del cambio, estrategia y política de SI), *integración* (de la empresa, de la función de SI y de las tecnologías de SI), *intensificaciones o perfiles profesionales* (*career tracks*). Cada perfil profesional consta de cuatro o más cursos opcionales que preparan a los estudiantes en una especialización. Una de estas intensificaciones es en gestión de datos y almacenes de datos, que consta de cuatro cursos: almacenes de datos, gestión de conocimiento, administración de bases de datos y planificación de sistemas de bases de datos.

La disciplina de bases de datos aparece en el bloque de "*Fundamentos*", en el curso: "Programación, Datos y Estructura de Objetos" donde se imparten conceptos de diseño y aplicación de datos, así como estructura de ficheros. El curso MS2000.1 "*Gestión de Datos*" ("Data Management") del bloque que constituye el *Núcleo* (*Core*) del programa está totalmente dedicado a las bases de datos. Asimismo, en otros cursos, aparecen cuestiones puntuales de gestión de datos (como en el Análisis, modelado y diseño –MSIS2000.2-); también en el bloque de *Integración* se debe estudiar la integración del recurso datos con otros recursos, de las tecnologías de gestión de datos con otras tecnologías, y de las funciones relativas a los datos con otras funciones de la empresa.

3.4. ISCC´99

El curriculum ISCC'99 (Information Systems-Centric Curriculum) pretende preparar

[5] Data Administration Management Association

[6] Association for Information Systems

especialistas de información para el desarrollo y utilización de grandes sistemas de información, y ha sido desarrollado por un equipo compuesto tanto por miembros de la comunidad universitaria como empresarial.

Se caracteriza por proponer un modelo invertido (en el que los estudiantes primero experimentan en el contexto de un Sistema de Información, después dominan los detalles y por último adoptan un punto de vista sistémico para completar su experiencia) y un enfoque centrado en sistemas de información que se centra en la organización de un sistema de información utilizando la información como un activo empresarial.

Se insiste en el curso de bases de datos (y en general en todo el curriculum) que los estudiantes trabajen en grupos pequeños sobre experiencias prácticas, en una enseñanza y aprendizajes "Justo a tiempo".

En la tabla 2 se muestra el contenido del curso ISCC-41 Information Databases and Transaction Processing, en el que señala el nivel de rendimiento a alcanzar (6: evaluación, 5: síntesis/diseño, 4: análisis, 3: aplicación, 2: comprensión, 1: recordatorio, 0: seguimiento de instrucciones) basándose en una modificación de la conocida taxonomía de Bloom.

Contenido	Rend.
1. Modelado de la información	
1.1 Modelos de datos:ANSI/SPARC	3
1.2 Bases de datos basadas en ficheros	3
1.3 Modelo E/R	3
1.4 Modelo Relacional	3
1.5 Modelos de datos semánticos	3
1.6 Modelo Orientado a objetos	3
1.7. Desarrollo de esquemas	3
2. Estructuras de información internas	
2.1 Estructuras de datos	3
2.2 Independencia de adtos	3
2.3 Medios de almacenamiento (menú principal, caché, disco)	3
2.4 Repositorios primarios, secundarios y terciarios	3
3. Construcción de una base de datos	
3.1 Conceptos de normalización	4
3.2 Diseño de bases de datos	4
3.3 Dependencias de datos	4
3.4 Restricciones de integridad	4
3.5 Procesamiento de análisis en línea	4
4. Procesamiento de consultas y optimización	
4.1 Lenguajes de consulta	4
4.2 Optimización de consultas	4
5. Operaciones de la base de datos	
5.1 Procesamiento de modificaciones	4
5.2 Procesamiento de transacciones	4
5.3 Control de concurrencia y seriabilidad	3
5.4 Recuperación	3
5.5 Sistema de procesamiento de transacciones en línea	4
6. Temas avanzados	
6.1 Bases de datos embebidas	4
6.2 Lenguajes de consult a abiertos	3
6.3 Almacenes y minería de datos	3
6.4 Integración con sistemas heredados	3

Tabla 2. Contenido del curso ISCC-41

3.5. ICF-2000 de IFIP/UNESCO

IFIP[7] y UNESCO[8] han diseñado el *Informatics Curriculum Framework* (ICF-2000) con el fin de abordar la situación de cambio constante a la que se enfrenta la informática. Como indica su nombre, realmente se trata de un marco (*framework*), a partir del cual se pueden construir diferentes implementaciones de currícula de una manera directa. En el propio documento (IFIP, 2000) –disponible en http://www.ifip.or.at- se presentan ocho especificaciones de currícula para ocho categorías de roles profesionales: Usuarios-I: Perfil BIP (*Basic Instrumental I-Profile*), Aplicadores Conceptuales-I: Perfil BCP (*Basic Conceptual I-Profile*), Aplicadores de Interfaz-I, Aplicadores de Investigación-I, Aplicadores de Dirección-I: Perfil MIP (*Minor I-Profile*), Trabajadores-I: Trabajadores operacionales-I, Trabajadores de Ingeniería-I, Trabajadores de Investigación-I: Perfil MAP (*Major I-Profile*).

Dentro de la categoría de trabajadores operacionales-I se encuentran los administradores de bases de datos, y dentro de los trabajadores de ingeniería-I, los desarrolladores de bases de datos. Como señala el propio marco, en el caso de una universidad con un departamento de informática consolidado, el perfil más interesante es el MAP, que incluye los tres últimos roles.

Dentro de las especificaciones de unidades destacan por su relación con bases de datos, las cuatro siguientes:

[7] International Federation on Information Processing
[8] United Nations Educational, Scientific and Cultural Organization

- MAP-02: arquitectura de sistemas de información,
- MAP-10: calidad y seguridad,
- MAP-12: modelado y desarrollo de sistemas,
- MAP-14: miscelánea/estado del arte (que incluye almacenes de datos).

Dentro de estas unidades se incluye como referencias, partes de los currícula CC91, IS97 e ISCC99. Así, por ejemplo, se incluyen todos los aspectos del curso "IS´97.8-Diseño físico e implementación con SGBD" que abarca: Modelos de datos y técnicas/herramientas de modelado, Enfoques estructurado y diseño de objetos, Modelos para bases de datos: relacional, jerárquico, red y orientado a objetos, Herramientas CASE, Diccionarios, repositorios y almacenes de datos, Implementación: codificación y/o implementación Windows/IGU; generación código/aplicaciones, Planificación, prueba e instalación cliente/servidor, Conversión de sistemas, formación de usuario final/integración y revisión post-implementación.

3.6. Otras propuestas

El SWEBOK ([9]) ofrece una panorámica general de las diferentes áreas de conocimiento a tratar dentro de la Ingeniería del Software. Cabe destacar que en algunos de sus capítulos referencia a las bases de datos, como en el tercero, de diseño de software, en el que se incluyen como notaciones de diseño el modelo E/R y los diagramas de clases. Sin embargo, no entra en profundidad en temas concretos relativos a la tecnología de bases de datos.

Propuestas sobre Bases de Datos y Web de [8]. Estos autores proponen atender la demanda de profesionales de comercio electrónico, impartiendo diversos cursos (dentro de las Escuelas de Negocio): EBT101: Programación Java para la Web, EBT201 Programación Web Avanzada, EBT301 Comercio Electrónico, EBT380 Herramientas y Técnicas para Aplicaciones Web, EBT393 Redes de Comunicaciones de Comercio Electrónico, EBT397 Desarrollo de Bases de Datos para Comercio Electrónico, EBT402 Desarrollo de Comercio Electrónico, EBT460 Gestión de Sistemas Empresariales, EBT488 Desarrollo de Aplicaciones para Comercio Electrónico y EBT494 Análisis y Diseño de Sistemas basados en la Web.

El curso sobre Desarrollo de Bases de Datos para Comercio Electrónico "proporciona una visión en profundidad de las cuestiones de la tecnología de bases de datos incluyendo el modelado de datos, CASE, diseño lógico e implementación en un SGBD relacional. Los estudiantes obtienen una experiencia práctica en la utilización de herramientas de desarrollo empresariales". La duración la fijan en 3 horas por semestre.

Por otro lado, cada vez hay más expertos tanto en el mundo académico como profesional que insisten en la necesidad de formar a los alumnos en Calidad de Datos. Así, por ejemplo, en [6] los autores proponen cuatro módulos educativos: Módulo 1: Importancia de la calidad de datos y el papel de los procesos de negocio en la calidad de datos, Módulo 2: El papel de la calidad de datos en el ciclo de vida de los datos, Módulo 3: Medida, valoración, traza y mejora de la calidad de datos y Módulo 4: Técnicas para mejorar la calidad de datos en el diseño y prueba de bases de datos.

En [4] se revisa el tratamiento de la calidad de a información en los diferentes currícula internacionales proponiendo el Information Quality Curriculum Model (IQCM) que consiste en cinco cursos: Introducción a los Sistemas de Información (IQ1), Gestión de los Datos (IQ2), Análisis de los requisitos de la Información y Diseño de Sistemas (IQ3), Proyectos de Sistemas de Información (Diseño físico e implementación) (IQ4) y Gestión de la Información (IQ5).

4. Análisis de los temas en los currícula

En [7], se comparan algunos de estos currículos y se representa un continuo entre la orientación más matemática y la más empresarial, así, el IRMA/DAMA 2000 tiene un enfoque más empresarial, ya que reconoce la información como el activo más importante y se prepara para aplicar las TI para resolver problemas empresariales, mientras que el CC2001 de ACM/IEEE sigue poniendo el énfasis importante en la parte matemática (especialmente el Volumen II de Ciencias de la Computación).

Hemos llevado a cabo un análisis exhaustivo

de los diferentes temas que incluye cada una de las asignaturas propuestas en el currículum de [1], por motivos de espacio sólo representamos el resumen final, que se muestra en la tabla 3, en la que se señala si, a nuestro juicio, los temas propuestos por el currículum cubre prácticamente en su totalidad (T) o de forma parcial (P) el propuesto en las asignaturas de [1].

Asignatura	CC2001 ACM/ IEEE	IRMA/ DAMA 2000	MSIS 2000 ACM/AIS	ICF-2000 IFIP/ UNESCO	ISCC´99
Bases de Datos 1 (BD1)	T	T	T	T	T
Bases de Datos 2 (BD2)	P	T	T	P	P
Diseño de Bases de Datos (DBD)	T	P	P	P	T
Administración de Bases de Datos (ABD)	T	P	T	P	P
Aplicaciones de Bases de Datos (APL)	T			P	
Bases de Datos Avanzadas (BDA)	P		P	P	P

Tabla 3.[1] frente a los currícula internacionales

Hay que destacar que:

- Varios currícula ya presentan los modelos de bases de datos orientados a objetos e incluso el multidimensional (almacenes de datos) como conocimientos básicos que el alumno debe aprender.
- Hay temas como las bases de datos deductivas que no se tratan específicamente en ningún currícula.
- Las distribuidas tampoco se consideran dentro de los conocimientos básicos en la materia

Además hay que destacar algunos temas que los currícula internacionales tratan pero que no se encuentran dentro del de [1]. En este sentido:

- CS`2001 incluye el almacenamiento y recuperación de la información, y las bibliotecas digitales, por un lado; mientras que por otro también contempla hipertextos/hipermedia y multimedia, además de minería de datos.
- El IRMA/DAMA 2000, enfatiza papel de las bases de datos como soporte para la toma de decisiones, Planificación estratégica de bases de datos, Sistemas de Información para Ejecutivos.

Con este análisis el nuevo currículum y el realizado con los planes de estudio de varias universidades y con los principales textos de bases de datos ([2]), proponemos el nuevo currículum que mostramos en la tabla 4.

Proponemos fundir DB1 y DB2 en una sola asignatura “Bases de Datos “ (BD), modificando substancialmente su contenido. Como la asignatura BD podría ser la única obligatoria, pensamos que debería introducir al alumno en los diferentes modelos de datos que se encontrará en su quehacer profesional. Tras una introducción a las BD, se presentarían las diferentes arquitecturas de un SGBD para pasar a presentar el concepto de modelo de datos. A continuación, se empezaría presentando muy brevemente los modelos en red, para dedicar la mayor parte del tiempo al relacional, seguidamente se presentarían los sistemas objeto-relacionales, así como los orientados a objetos puros. También se presentarían el modelo multidimensional y los almacenes de datos, así como los modelos semiestructurados y el XML. Se finalizaría con una visión general de la administración de bases de datos.

La asignatura de diseño se dejaría como está, aunque en nuestra opinión cada vez se justifica menos proponer asignaturas diferentes para el diseño de los datos y el de las aplicaciones debiendo ambos aspectos incluirse en una sola asignatura de diseño de sistemas o de ingeniería del software. La administración de bases de datos también permanecería prácticamente igual mientras que la de APL habría que actualizarla incorporando nuevos estándares como SQLJ, etc.

ASIGNATURA	CONTENIDO
BASES DE DATOS (BD) **9/12 créditos.** **Obligatoria para II (primer ciclo), ITIS e ITIG**	1. Introducción a las BD 2. Arquitectura de un SGBD 3. Concepto de modelo de datos 4. Modelos en red (opcional) 5. Modelo relacional 6. Modelos de objetos 7. Modelo multidimensional 8. Modelos semiestructurados (XML) 9. Introducción a la administración de bases de datos 10. Introducción al desarrollo de aplicaciones con bases de datos 11. Futuro de las bases de datos Prácticas: Utilización de SQL con bases de datos relacionales, objeto-relacionales y multidimensionales
DISEÑO DE BASES DE DATOS (DBD) **6 créditos.** **Optativa para II, ITIS.** **Obligatoria ITIG**	1. El proceso de creación de una base de datos 2. Una metodología para el diseño de bases de datos 3. Modelado conceptual 4. Modelo E/R extendido/Modelo UML 5. Diseño lógico 6. Diseño físico 7. Diseño de bases de datos distribuidas Prácticas: Casos complejos de diseño utilizando alguna herramienta CASE e implementando la base de datos sobre algún producto existente.
ADMINISTRACIÓN DE BASES DE DATOS (ABD) **6 créditos.** **Optativa para II, ITIS, ITIG**	1. Introducción 2. Confidencialidad 3. Integridad 4. Concurrencia 5. Recuperación 5. Optimización de Bases de Datos: Técnicas básicas para mejora de rendimiento y de afinamiento de BD Prácticas: Instalación, creación de usuarios, privilegios, etc. Análisis del optimizador y caminos de acceso, afinamiento, etc. Bloqueos y concurrencia
APLICACIONES DE BASES DE DATOS (APL) **6 Créditos** **Optativa para II, ITIG**	1. Técnicas de programación en cliente servidor. 2. SQL embebido estático y dinámico. Trabajo con cursores. Comparativa con CLI. 3. Trabajo en entornos 4GL y RAD. 4. APIs: ODBC, SQLJ, JDBC, etc. 5. Desarrollo de interfaces Web para aplicaciones de bases de datos (CGI y otros). 6. Integración de aplicaciones de bases de datos en entornos ofimáticos. Prácticas: Desarrollo completo de una aplicación cliente/servidor utilizando algún API y con acceso vía web
BASES DE DATOS AVANZADAS (BDA) **6 créditos** **Optativa para II (Segundo Ciclo)**	1. Introducción 2. Ampliación a las bases de datos orientadas a objetos 3. Ampliación a las bases de datos multidimensionales 4. Bases de datos distribuidas y heterogéneas 5. Bases de datos documentales y bibliotecas digitales 6. Bases de datos móviles 7. Bases de datos paralelas 8. Bases de datos geográficas/espaciales 9. Bases de datos multimedia 10. Calidad de la Información 11. Otros modelos de bases de datos: deductivas, seguras. Prácticas: Presentaciones por parte de los alumnos de ciertos temas relacionados con avances de BD Diseño de una BDOO con ODMG Diseño de una BDORel con SQL:1999 Uso de algún prototipo de BD deductiva

Tabla 4. Nuestra propuesta

5. Conclusiones

La tecnología de bases de datos está evolucionando y resulta fundamental adaptar los currículum a estos cambios. En este artículo hemos presentado una propuesta de currículum realizada estudiando los diferentes currícula internacionales y comparándolos con el propuesto por [1]. Aunque sabemos que es discutible, también consideramos que es necesaria para poder asegurar a los alumnos los conocimientos necesarios para enfrentarse a las tecnologías y al diseño de las bases de datos actuales.

Referencias

[1] Blesa, P., Brisaboa, N., Canivell, V., Garbajosa, J., Maudes, J. y Piattini, M. (1999), "Propuesta de contenidos en bases de datos de los planes de estudio de Informática". NOVÁTICA (Revista de la Asociación Técnicos de Informática), Nº 137, Enero/febrero 1999, pp. 60-63.

[2] Calero, C., Piattini, M. y Ruiz, F. (2002). Propuesta de actualización del currículum de bases de datos. Novática.

[3] Cohen, E. y Kozminski, L. (2001). Rationale for the IRMA/DAMA 2000 Model Currículum. Proc. of the 2001 IRMA (Information Resource Management Association) International Conference, 612-616.

[4] Kahn, B. y Strong, D. (2001) Where Inforamtion Quality in Information Systems Education is?. In Information and Database Quality, Piattini M., Calero, C. and Genero, M. (eds). Kluwer Academic Publishers.

[5] Leavitt, N. (2000). Whatever happened to object-oriented databases. IEEE Computer Society. August.

[6] Mathieu, R.G. y Khalil, O. (1998). Data Quality in the Database Systems Course. Data Quality Journal, 4 (1), 1-12.

[7] Scime, A. (2001). Information Systems and Computer Science Model Curricula: A Comparative Look. Proc. of the 2001 IRMA (Information Resource Management Association) International Conference, 496-501.

[8] Surynt, T. y Augustine, F. (2001). E-Business Technology: A Component-Based Curriculum for the Future. Proc. of the 2001 IRMA (Information Resource Management Association) International Conference, 686-689.

[9]Guide to the software engineering body of knowledge (SWEBOK). IEEE trial version (2001). Disponible en http://swebok.org

Una Herramienta para el Aprendizaje del Álgebra Relacional

Carmen Hernández, Yania Crespo, Pilar Romay, Miguel Angel Laguna
Departamento de Informática
Universidad de Valladolid
47011 Valladolid
e-mail: {chernan, yania, mpromay, mlaguna}@infor.uva.es

Resumen

El estudio de las bases de datos relacionales forma parte del currículum de los estudios de Ingeniería Técnica Informática. Los lenguajes de consulta abstractos, entre ellos el Álgebra Relacional, forman una parte importante de este estudio, pero nuestra experiencia docente nos ha demostrado que, para los alumnos, es difícil conocer cuándo las consultas expresadas en un papel en términos de estos lenguajes son correctas y responden a los requisitos de información planteados.

En este trabajo se describe una herramienta de apoyo que se ha desarrollado en la Universidad de Valladolid que permite realizar consultas expresadas en Álgebra Relacional sobre cualquier base de datos. El alumno puede explorar, así, las diferentes posibilidades de este lenguaje abstracto, comprobando por sí mismo la calidad de su aprendizaje. La herramienta ha sido desarrollada siguiendo guías metodológicas propugnadas en el ámbito del diseño de entornos de aprendizaje.

1. Introducción

El modelo relacional se ha establecido como el principal modelo de datos para la construcción de Sistemas de Información. Este modelo tiene unos sólidos fundamentos matemáticos, ya que se basa en la Teoría de Conjuntos. Esta base teórica ayuda al diseño de las bases de datos relacionales y, particularmente, al procesamiento eficiente de las peticiones de información de los usuarios. El modelo define de forma precisa los diferentes lenguajes abstractos con los que el usuario solicita información de la base de datos. Uno de estos lenguajes es el Álgebra Relacional.

El estudio de las bases de datos relacionales forma parte del currículum de los estudios de Ingeniería Técnica Informática desde la implantación de éstos en la Universidad de Valladolid. En particular, los lenguajes de consulta abstractos forman una parte importante de este estudio. Sin embargo, nuestra experiencia docente nos ha demostrado que para los alumnos es difícil conocer cuándo las consultas expresadas en el papel en términos del Álgebra Relacional son correctas y responden a los requisitos de información planteados.

Parecía de especial interés, por tanto, desarrollar una herramienta que proporcionara un mecanismo mediante el cual los alumnos puedan explorar las diferentes posibilidades de estos lenguajes. Además, se debería permitir una realimentación inmediata, de manera que vieran el resultado de la consulta que habían propuesto en ese mismo momento y pudieran realizar las oportunas modificaciones.

La herramienta que hemos construido pone especial cuidado en conseguir la pretensión inicial de facilitar el aprendizaje. Hemos revisado algunas de las teorías existentes sobre el diseño de entornos de aprendizaje, intentando concretar las principales características de este tipo de entornos y las guías existentes para la construcción de los mismos.

En la siguiente sección se presenta el entorno educativo en el que se ha desarrollado la herramienta. La tercera sección sirve para introducir brevemente algunos conceptos fundamentales del diseño de entornos de aprendizaje. En la cuarta sección presentamos la herramienta que hemos desarrollado. Finalmente, exponemos las líneas de nuestro trabajo futuro y las conclusiones de éste.

2. Entorno educativo

El modelo relacional define de forma precisa los diferentes lenguajes abstractos con los que un usuario solicita información de la base de datos. Estos lenguajes de consulta pueden clasificarse en procedimentales y no procedimentales. En los procedimentales, el usuario indica al Sistema Gestor de Base de Datos Relacional (SGBDR) la serie de operaciones que ha de realizar en la base de datos para obtener el resultado deseado. En los no procedimentales el usuario describe la información deseada sin dar un procedimiento concreto para obtener esa información.

La mayor parte de los SGBDR ofrecen un lenguaje de consulta que incluye elementos de los enfoques procedimental y no procedimental. El lenguaje SQL (Structured Query Language) es un ejemplo de lenguaje que se ajusta a los dos enfoques.

Los lenguajes abstractos ilustran de forma precisa las técnicas fundamentales para la extracción de datos de las bases de datos, obviando los aspectos sintácticos de los lenguajes comerciales. El Álgebra Relacional es un lenguaje abstracto de consulta procedimental. Consta de un conjunto de operaciones que toman como entrada una o dos relaciones y producen como resultado una nueva relación.

Como ya se ha comentado, el estudio de las bases de datos relacionales forma parte del currículum de los estudios de Ingeniería Técnica Informática. Los lenguajes de consulta abstractos, fundamentalmente el Álgebra Relacional, forman una parte importante de este currículum.

Nuestra hipótesis de trabajo es que los alumnos aprenden más rápidamente el lenguaje SQL cuando ya están familiarizados con las cuestiones que subyacen en los lenguajes de consulta formales, mediante el uso del Álgebra Relacional [3]. En particular, la sentencia *select-from-where* de SQL se define precisamente en términos del Álgebra, de manera que el mecanismo mediante el cual el SGBDR es capaz de recuperar información surge ante el alumno de una manera casi espontánea.

Nuestra experiencia, tras varios años impartiendo esta asignatura, nos permite asegurar que para los alumnos es difícil conocer cuándo una consulta expresada en Álgebra y escrita en un papel es correcta. Esta dificultad se extiende al profesor, al tener que controlar la corrección de algunas consultas "creativas" que a veces se proponen.

Todo esto justifica la necesidad de utilizar una herramienta que proporcione un mecanismo mediante el cual los alumnos puedan explorar las diferentes posibilidades de estos lenguajes, permitiendo, además, una realimentación inmediata, de manera que ellos vean el resultado de la consulta que habían propuesto en ese mismo momento y puedan realizar las oportunas modificaciones, hasta alcanzar una respuesta satisfactoria.

Durante algunos años hemos estado utilizando en nuestro laboratorio de Bases de Datos un prototipo de una herramienta que ha sido desarrollada en Arizona State University de EE.UU. [2], que también tiene como objetivo el aprendizaje de estos lenguajes.

Este prototipo no se ajusta estrictamente a los requisitos que nosotros tenemos planteados, ya que no tiene incorporados todos los operadores que presentamos en la asignatura, con lo que algunas de las cuestiones que planteamos al alumno no tienen respuesta en el ámbito de esta herramienta.

Con esta experiencia y con las lecciones que hemos aprendido en nuestra práctica docente, hemos desarrollado nuestra propia herramienta, que se ajusta exactamente a los contenidos de nuestra asignatura, de modo que, realmente, se facilite el aprendizaje del alumno.

3. Diseño de entornos de aprendizaje

Todo proceso de aprendizaje está condicionado por las variables del medio en que se desarrolla, de manera que este proceso está determinado por el entorno en el que tiene lugar. Cuando se trata de un proceso condicionado por un determinado software, el aprendizaje es estimulado y dirigido por aspectos diversos que vienen condicionados por el nivel del producto alcanzado, por su riqueza cualitativa y por su complejidad.

Podemos entender el entorno en el que se desarrolla el aprendizaje como un marco de situaciones y actividades que orientan y guían el aprendizaje, lo condicionan y lo estimulan para que tome una dirección u otra. En este sentido, el

entorno lleva de la mano al usuario por los caminos previamente fijados por el diseñador, ya que cualquiera que sea la propuesta que se presente al usuario, abierta o cerrada, o con múltiples opciones, todas han sido previamente definidas en función de la idea de guiar el proceso [6].

Se podría decir, entonces, que lo que el diseñador hace con su trabajo es una anticipación del proceso de aprendizaje que realizará el usuario. El diseñador realiza una actividad de enseñanza al organizar los ambientes y los elementos que orientarán el aprendizaje de los usuarios. La tarea de enseñar implica siempre, en cualquier contexto, ya sea virtual o presencial, el diseño de entornos y la previsión de situaciones cuya intención es que conduzcan al éxito en el aprendizaje.

Teniendo todo esto en cuenta, el diseño de software educativo ha de ser concebido como una construcción metodológica. Esto significa que no es absoluta, ni para todas las situaciones, ni para todos los contenidos e individuos y que, por lo tanto, debe ser planteada, en el marco de situaciones concretas, en un contexto determinado [5]. De esta manera, se puede hablar del diseño como un proceso que toma la forma de espiral constructivo, y no como una secuencia lineal de selección de los elementos que lo integran. Todo esto determina el ciclo de vida que mejor se ajusta al tipo de sistema que pretendemos construir.

En el diseño de este tipo de sistemas intervienen cuatro elementos básicos a partir de los cuales se determina el material a construir: la estructuración del contenido, las actividades a realizar, los recursos con que se cuenta y las formas de interacción que se prevén, generando en conjunto un ambiente o entorno que ayude u oriente el aprendizaje del usuario [4].

La secuencia y organización de los contenidos determina las formas de apropiación de los conocimientos por parte de los usuarios. Esto hace que la selección de estos sea una etapa crucial que demanda del diseñador un profundo conocimiento de las disciplinas científicas que están en juego en lo referente a los conceptos principales que se quieren trasmitir.

Otra etapa fundamental en este planteamiento es la estructuración de las actividades que van a desarrollar los usuarios para operar con el sistema. Las actividades que se propongan deben estimular la actitud de interrogación. El usuario debería visualizar las consecuencias de su acción sobre el problema, elaborar hipótesis y comprobarlas durante el proceso, visualizando los resultados de sus deducciones, a través de la experimentación.

Planear la actividad implica generar un proceso reflexivo y de construcción del conocimiento por parte del diseñador, que debe enfrentarse al mismo desafío que se planteará el usuario.

Como consecuencia de todo lo expuesto, hemos afrontado la tarea de diseñar esta herramienta mediante un ciclo de vida en espiral [1], de forma que nos encontramos delante del primer prototipo de esta herramienta en el que se han tenido en cuenta, fundamentalmente, la estructuración de los contenidos y los recursos que se van a necesitar.

El equipo de diseñadores ha estado formado por profesores involucrados en las diferentes asignaturas de Bases de Datos que se imparten y por alumnos que realizaban su Proyecto Fin de Carrera. Los primeros se han encargado de proporcionar el conocimiento de la disciplina que se intentaba dar a conocer y los segundos han asumido perfectamente el papel del usuario que debe enfrentarse al problema de aprender un lenguaje de consulta nuevo, como es el Álgebra Relacional.

4. La herramienta

La herramienta que se ha desarrollado [7] pretende facilitar la comprensión de las siguientes materias:

- Operadores que constituyen el Álgebra Relacional.
- Combinaciones de operadores para formar consultas.
- Semántica del Álgebra Relacional.
- Sintaxis y semántica del SQL.

Según lo expuesto en el apartado anterior, para que la herramienta sea útil desde un punto de vista didáctico, los alumnos deben de poder emitir respuestas a las cuestiones que les hayan sido planteadas y el sistema tendrá que detectar los errores concretos que se hayan cometido, además de ofrecer la posibilidad de hacer modificaciones a lo respondido.

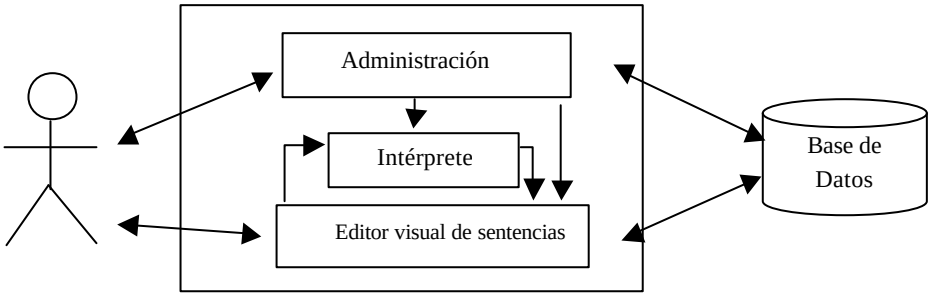


Figura 1. Arquitectura del sistema

Las funcionalidades básicas que la herramienta debe cubrir son:

- Mantener el esquema de la base de datos con la que se va a trabajar.
- Mantener la instancia de la base de datos.
- Definir consultas en Álgebra Relacional: Se introducirá una consulta mediante el teclado y el sistema la evaluará. Se debe dar la posibilidad de almacenar la consulta para utilizarla en otro momento.
- Visualizar resultados: El sistema evalúa la consulta y presenta el resultado en pantalla, pudiendo almacenarse la nueva tabla obtenida de forma permanente en la base de datos como una vista.

En todo momento se buscará la máxima interactividad con el alumno, de manera que el sistema siempre mantendrá informado al usuario de los errores que vaya cometiendo.

Para permitir la máxima disponibilidad de la herramienta, se ha desarrollado sobre una plataforma Windows, de forma que los alumnos puedan utilizarla en sus propios ordenadores.

4.1. Arquitectura

La arquitectura, que se muestra en la Figura 1, favorece la disponibilidad y usabilidad de la herramienta desarrollada. En ella se pueden apreciar los distintos módulos que constituyen el sistema y los interfaces que se establecen entre ellos.

Se ha optado por dividir la funcionalidad del sistema construido en dos partes; por un lado se encuentra el proceso encargado de la traducción de la sentencia en Álgebra Relacional a un conjunto de instrucciones SQL equivalentes que serán utilizadas para obtener los resultados de la consulta y, por otro, el proceso que actúa de interfaz con la base de datos y con el usuario.

Al dividir el sistema de este modo se puede afrontar la posterior implementación de cada uno de los procesos con las tecnologías más adecuadas para cada caso, teniendo en cuenta las funciones que deben cumplir.

El módulo *Intérprete* se encarga del proceso de traducción de las sentencias del Álgebra Relacional al conjunto de instrucciones SQL para *Access*, a partir de las cuales se obtendrá el resultado de la consulta equivalente a la sentencia inicial.

El módulo *Editor visual de sentencias* se encarga de la funcionalidad necesaria para crear, editar, compilar y guardar sentencias del Álgebra Relacional. Una vez obtenidas las instrucciones SQL para Access mediante el *Intérprete*, este módulo obtiene el resultado a través del motor de base de datos *Microsoft Jet 3.0* y lo visualiza.

El módulo *Administración* permite al usuario determinar la base de datos con la que va a trabajar. Obtiene la *ruta del origen de datos* y la estructura de la base de datos, que son almacenadas de forma apropiada para su posterior uso por el módulo *Editor visual de sentencias.*

La base de datos contiene los datos sobre los cuales va a trabajar el usuario a través de las sentencias del Álgebra Relacional.

En la implementación de la herramienta se ha optado por utilizar *Visual C++* en su versión 6.0 y la biblioteca *MFC* (*Microsoft Foundation Class*). Además, se ha utilizado *Access* como base de datos y su motor de base de datos *Microsoft Jet* en su versión 3.0, lo que permite la comunicación con el sistema a través de las funciones de las clases *DAO* (*Data Access Objects*). De esta

manera, el usuario podrá practicar con diferentes bases de datos, sin más que modificar el *origen de datos* con el que quiere trabajar.

4.2. Interfaz de usuario

La primera vez que se utiliza la herramienta se advierte al usuario de la necesidad de configurar el *origen de datos*; es decir, la ruta de la base de datos sobre la que se va a trabajar. Esta base de datos debe existir, aunque posteriormente se podrá modificar su esquema añadiendo o eliminando nuevas tablas y actualizando las tuplas que pertenecen a cada una de ellas.

Figura 2. Operaciones implementadas.

Las operaciones que se han implementado en esta herramienta aparecen disponibles en una ventana flotante, de manera que el usuario puede escribir una sentencia en términos del Álgebra Relacional de una manera muy cómoda. Estas operaciones son: Selección, Proyección, Intersección, Unión, Cociente, Reunión (Join), Producto Cartesiano, Diferencia y Renombrado.

En la Figura 2 aparece esta ventana con las operaciones disponibles.

En la aplicación también se cuenta con otra ventana en la que aparecen una serie de pestañas que muestran información de cierto interés para el usuario. Esta ventana aparece en la Figura 3. Las pestañas son las siguientes:

- *Errores*: Muestra la descripción de los errores que se hayan podido producir durante el proceso de compilación.
- *SQL:* Muestra las instrucciones SQL resultantes del proceso de compilación si éste ha terminado con éxito.
- *Tablas*: Muestra la descripción de la base de datos con los nombres de las tablas y los atributos correspondientes.

Por último, la ventana de edición permite al usuario introducir la sentencia del Álgebra Relacional, visualizando cada operador con la fuente adecuada.

En la Figura 4 aparece la ventana de edición con una consulta expresada en términos del Álgebra Relacional y que permite proporcionar el nombre, apellidos y dirección de todos los empleados que trabajan en el Departamento que tiene por nombre “Investigación”.

Los resultados de la consulta realizada por el usuario aparecen en otra ventana. Allí aparecen las tuplas que pertenecen a la relación definida por la expresión introducida por él.

Los resultados, evidentemente, no pueden ser modificados, pero a la vista de lo obtenido, el alumno puede modificar su consulta, o introducir nuevos datos en las tablas de la base de datos que le permitan aceptar o rechazar la expresión que haya construido como respuesta a la necesidad de información previamente planteada por él o por una serie de cuestiones que le son planteadas en las sesiones de laboratorio.

Figura 3. Ventana de información

Figura 4. Ventana de edición

5. Trabajo futuro

Como ya hemos comentado, nos encontramos ante la primera versión de esta herramienta. Es necesario recordar que nos planteamos el diseño de este tipo de software como un proceso evolutivo y dinámico, de manera que una de nuestra líneas de trabajo se centra en medir la efectividad de dicha herramienta en el proceso de aprendizaje de los alumnos.

El diseño del experimento para llevar a cabo esta tarea nos hace pensar en la comparación de dos tratamientos: aprendizaje usando la herramienta y aprendizaje sin usarla (como se estaba haciendo antes de incorporarla). Esta tarea se desarrollará durante el primer cuatrimestre del curso 2002/2003.

Otra línea de trabajo se centra en revisar el diseño inicialmente realizado para intentar eliminar la necesidad de utilizar un determinado SGBDR, como ocurre en la versión actual, incorporando tecnología *JDBC/ODBC*, lo que indudablemente incrementaría la disponibilidad de la herramienta. Este trabajo se ha acometido mediante la propuesta de un Proyecto Fin de Carrera, que ya se está realizando en la actualidad y que previsiblemente estará terminado en el mes de Julio de este año.

Por último, mantenemos una línea de trabajo abierta para posibilitar el uso de la herramienta en la World Wide Web. De esta manera, enlazaríamos este trabajo con otro que ya está en marcha y que pretende la construcción de un portal web en castellano para el aprendizaje de lenguajes de manejo de bases de datos. Para la realización de este proyecto se ha solicitado una subvención a la Junta de Castilla y León dentro de su **"**Programa de Ayudas para la Elaboración de Recursos de Apoyo y Experiencias Innovadores en la Enseñanza Universitaria".

6. Conclusiones

En este trabajo se ha presentado una herramienta que facilita el aprendizaje del Álgebra Relacional. Se trata de un sistema capaz de realizar consultas sobre cualquier base de datos. Ante una sentencia expresada en Álgebra, el sistema traduce esta sentencia a SQL. Esta traducción se muestra al alumno haciéndole más patente la relación entre ambos lenguajes. Se le muestran los operadores que puede utilizar y las tablas, junto con sus atributos, que puede utilizar para hacer la consulta. Además, se permite una realimentación inmediata, al incluir la posibilidad de rescribir su consulta, de manera que puede ver el resultado de la consulta que había propuesto y puede realizar las oportunas modificaciones.

La herramienta ha sido desarrollada siguiendo guías metodológicas propugnadas en el ámbito del diseño de entornos de aprendizaje, para garantizar que cumple los fines con los que ha sido concebida.

Referencias

[1] Boehm, B., "A Spiral Modelo for Software Development and Enhancement". *Computer*, vol. 21, nº 5, pp. 61-72.

[2] Dietrich, S.W., Eckert, E., Piscator, K., "WinRDBI: A Windows-based Relational Database Educational Tool", *Proceedings of the 28th ACM SIGCSE Technical Symposium on Computer Science Education*, San Jose, California, February 27-March 1, 1997, pp. 126-130.

[3] Dietrich, S.W., Urban, S.D., "Cooperative Learning Approach to Database Group Projects: Integrating Theory and Practice", *IEEE Transactions on Education*, November 1998, CD-ROM Directory 06, 11p.

[4] Furlán, A., "Metodología de la enseñanza". *Aportaciones a la didáctica en la Educación Superior*, UNAM-ENEP, Iztacala, México, D.F. 1989.

[5] Gewerc, A., "Diseño de entornos de aprendizaje", http://www.quadernsdigitals.net/articles/quadernsdigitals/quaderns24/q24disenyo.htm. 2002

[6] Gros, B., "*Diseño de programas educativos",* Barcelona, Ariel, 1997.

[7] Quintana, R.A., Valentín, T., "*Diseño e implementación de un Intérprete del Álgebra Relacional para uso docente*", P.F.C., Universidad de Valladolid, Enero 2002.

Utilización de versiones de tablas en la docencia de SQL interactivo

Mª Belén Vaquerizo García, Angel Arroyo Puente, Jesús Manuel Maudes Raedo

Escuela Politécnica Superior, Departamento de Ingeniería Civil,

Área de Lenguajes y Sistemas Informáticos, Universidad de Burgos,

Av. Cantabria s/n., 09006, Burgos (España)

e-mail: {belvagar, aarroyo, jmaudes}@ubu.es

Resúmen

La docencia de SQL en los cursos iniciales de un currículo en informática es una materia con una gran componente práctica. En este artículo se discute como plantear el esquema de base de datos sobre el que hacer prácticas en función de la metodología docente y contenidos a impartir, teniendo en cuenta la dicotomía entre dotar de privilegios al alumno para que experimente, o acotárselos para tenerlo más controlado. Finalmente, se discutirán las ventajas e inconvenientes de una aproximación basada en versiones de tablas que puede servir de punto de equilibrio en cuanto a permisos sobre el sistema para los alumnos.

1. Introducción

La docencia en Bases de Datos requiere, ya en sus fases más tempranas, que el alumno se enfrente a los lenguajes de bases de datos practicando con ellos. En [1], [2] se propone un esquema secuencial de asignaturas de Bases de Datos distribuidas a lo largo de las titulaciones de informática. Estos documentos recomiendan que la primera asignatura a la que se enfrenten los alumnos (*Bases de Datos I*) abarque entre sus contenidos, el trabajo con SQL interactivo.

La experiencia de los que subscriben el presente documento es que la docencia de estos contenidos tiene unas componentes más pedagógicas dentro del aula de prácticas que dentro de una clase tipo lección magistral. Esta opinión es refrendada en las encuestas de calidad rellenadas por los alumnos y tiene, probablemente, sus orígenes en los siguientes motivos:

- Los grupos de prácticas son más reducidos que los grupos correspondientes a las lecciones magistrales, lo que permite al profesor dar una mayor atención individualizada a cada alumno.
- El alumno tiene la oportunidad de experimentar, y aprender por sí mismo; sin dedicar la mayor parte del tiempo a tomar apuntes.
- El profesor puede observar la marcha de sus alumnos, teniendo la oportunidad de reaccionar e incidir sobre aquello que ha quedado menos claro.
- Los alumnos tienen una mayor motivación, al tender a identificar las prácticas con el que es, para la mayoría de ellos, objetivo último de sus estudios, que no es otro que la preparación para insertarse convenientemente en el mercado de trabajo.

Todos estos puntos, hacen necesario cuidar (i) los contenidos, (ii) la metodología y (iii) los medios utilizados en la presentación de contenidos

en las prácticas en general, y en nuestro caso concreto, en las prácticas de SQL interactivo en Bases de Datos.

En cuanto a los contenidos de SQL interactivo, es interesante que el alumno pueda experimentar con el DML (*Data Management Language*) de SQL por completo, y con el DDL (*Data Definition Language*) a nivel de creación de tablas y vistas. Las operaciones de actualización en DML deben de permitir al alumno experimentar con la concurrencia a nivel transaccional, y referente a las vistas, ha de ser posible trabajar con vistas actualizables.

Con relación a la metodología en el aula de prácticas, hay varias posibilidades: (i) explicaciones del profesor, (ii) problemas a resolver por el alumno solo, (iii) problemas a resolver por el conjunto de los alumnos guiados por el profesor y discutiendo cada una de las posibles soluciones. La explicación de los problemas por parte del profesor no fomenta la participación del alumno, pero tiene como ventaja que es la forma de avanzar más deprisa en el temario y puede ser útil en aquellos puntos en los que el profesor estime que no es interesante incidir demasiado. Dejar al alumno que resuelva los problemas por sí mismo suele consumir demasiado tiempo, pero el alumno suele sacar bastante partido de su trabajo. Esta fórmula es idónea para aquellas horas en las que el aula de prácticas esté libre para uso de los alumnos, o para un acceso web desde casa del alumno al Sistema de Base de Datos utilizado en prácticas. Finalmente, el resolver los problemas por el conjunto de los alumnos guiados por del profesor (de ahora en adelante *prácticas guiadas*) es una fórmula intermedia que fomenta la participación y el aprendizaje, optimizando el tiempo dedicado. Si bien, nosotros nos decantamos a abusar de esta última fórmula, el profesor debe además utilizar las otras dos allí donde estime que los contenidos, la aptitud y actitud del alumno puedan hacerlas más interesantes.

Finalmente, en cuanto a los medios, históricamente ha habido aproximaciones de Sistemas de Bases de Datos orientados a docencia, como WinRDBI[1] [3] y [4]. El inconveniente principal de éste y otros sistemas similares, es que aunque intenta abarcar otros lenguajes como Álgebra y Cálculo Relacional, la parte de SQL no es muy fiel al SQL estándar, especialmente en lo concerniente a la definición de tablas, restricciones y vistas. Para nosotros es necesario que el sistema sea lo más fiel posible al estándar ISO de SQL, preferiblemente a la versión de 1992 [6], por ser más sencilla y más extendida en la industria. Otro inconveniente de WinRDBI es que es monousuario impidiendo experimentar con la concurrencia.

La solución a concurrencia y acercamiento al estándar viene de la mano de la utilización de Sistemas Gestores de Bases de Datos (SGBDs) comerciales. Aunque en la actualidad hay buenas ofertas económicas para docencia de este tipo de sistemas, hay que valorar la adecuación de productos de libre distribución como por ejemplo *PostgreSQL*[2] y *MySQL*[3].

Sin embargo, los SGBDs comerciales plantean el problema de cómo articular los privilegios de los alumnos para que puedan compartir datos con el profesor en una práctica guiada, pero a la vez puedan modificarlos para experimentar, sin que por ello arruinen la labor del profesor, ni de sus compañeros. En la medida que al alumno se le provee independencia y privilegios sobre el entorno, el profesor pierde el control sobre el contenido de las bases de datos de los alumnos, lo que puede impedirle impartir adecuadamente unas prácticas guiadas. Por el contrario, recortar demasiado las posibilidades del alumno, puede ir en contra de que experimente por su cuenta, lo que también es perjudicial.

Este artículo trata sobre cómo plantear soluciones a este problema para cada uno de los posibles escenarios que aparecerán en función del método y contenidos empleados. Para ello se estructura de la siguiente forma: En el apartado 2 plantearemos las diversas posibilidades de

[1] http://www.eas.asu.edu/~winrdbi/

[2] http://postgresql.org/

[3] http://www.mysql.com/ Desafortunadamente la versión actual de *MySQL* no soporta transacciones, por lo que es poco ideal para DML con concurrencia.

esquemas y privilegios en función del tipo de contenidos. En el apartado 3 describiremos la aproximación basada en versiones de tablas, adecuada para prácticas guiadas. En el apartado 4, describiremos los puntos más notables de una implementación real en *PostgreSQL* de la aproximación basada en versiones de tablas. Finalmente, en el apartado 5 se comentarán las conclusiones y trabajos futuros.

2. Planteamiento del Esquema de Base de Datos y Permisos en función de los Contenidos

Los contenidos posibles de SQL interactivo en una asignatura introductoria a las Bases de Datos deben de cubrir el DML por completo, y el DDL a nivel de definición de tablas y vistas. La experimentación con concurrencia y con vistas actualizables también debe de estar amparada. En este apartado mostraremos cómo plantear los esquemas y privilegios de la base de datos para cada uno de estos contenidos.

2.1. Las Consultas SELECT

La configuración más habitual para una práctica guiada sobre consultas SELECT es hacer que todos los alumnos compartan las tablas del profesor, desde un usuario distinto de forma que los alumnos no puedan modificar su contenido, sino que tan solo puedan consultarlo. Las tablas del profesor tendrían únicamente permiso de lectura para los usuarios alumnos. Los alumnos podrían acceder perfectamente desde un único usuario del sistema para facilitar el mantenimiento. Este usuario alumno no necesitaría privilegios que le permitiesen crear objetos de la base de datos como tablas o vistas. En esta situación el alumno puede experimentar consultas con las tablas del profesor:

1 Sin temor a que el alumno cree algún tipo de problema por excederse en la ocupación de disco, pues no puede crear tablas, ni modificar las del profesor.

2 Sin temor a que bloquee las operaciones de otros compañeros, pues todos están accediendo en modo consulta.

3 El profesor puede cambiar la extensión de las tablas sobre la marcha para conseguir efectos particulares en las consultas (introducir valores nulos para ver comportamientos atípicos etc …) teniendo la certeza de que esos efectos son inmediatamente visibles a todos los alumnos. Dado que los alumnos y el profesor acceden a los mismos datos, es conveniente que el sistema admita relajar el control de la concurrencia de forma que en la misma transacción del alumno se puedan obtener distintas lecturas de las tablas del profesor, las cuales pueden ir cambiando por acción de transacciones cometidas de éste. Este nivel de concurrencia relajado es el conocido como *Read Commited* en el estándar SQL/ISO[6]. Esta configuración viene activada por defecto en algunos sistemas como *Oracle*. Otra solución posible al mismo problema, es que los alumnos tengan activada la opción de *autocommit*, habitual en casi todos los sistemas, de forma que cada una de sus consultas sea una transacción en sí misma.

Desde el punto de vista estético, este planteamiento puede resultar feo en sistemas como *Oracle* [5] que tienen un espacio de nombre de tablas distinto para cada usuario. Esto obliga a que el alumno para referenciar a la tabla *X* del usuario *profesor*, tenga que escribir *profesor.X*. Este problema puede ocultarse definiendo una vista[4] con el mismo nombre que la tabla (*X*) en el alumno (CREATE VIEW X AS SELECT * FROM Profesor.X).

Desde el punto de vista funcional, el principal inconveniente de esta aproximación, es que el alumno no puede experimentar por si sólo, creando sus propios casos (crear sus propias tablas, modificar la extensiones de las tablas existentes) para ver cómo varían las respuestas a una misma consulta.

Esta aproximación sólo es válida para cuando el profesor explica sin que el alumno participe, y –aparentemente- para la clase de prácticas guiada. Nótese que si en una clase guiada el profesor, en

[4] O bien un sinónimo (SYNONYM) de la tabla del profesor en el usuario alumno en el caso de aquellos sistemas que dispongan de este tipo de objetos, como por ejemplo *Oracle* [5].

lugar de pedir que dado el enunciado de una consulta los alumnos encuentren la respuesta, pide que dada una SELECT -que parece resolver un enunciado- los alumnos encuentren una extensión de la tabla para la que dicha SELECT no cumple su cometido, los alumnos no pueden realizar dicho ejercicio utilizando el sistema.

Por lo tanto, en este caso se necesita una aproximación que de más libertad al alumno. Para que el alumno pueda cambiar las extensiones de las tablas del profesor necesita poder hacer INSERT-UPDATE-DELETE. Si el profesor comparte sus tablas permitiendo estas operaciones a los alumnos, se verá obligado a pasar todo el tiempo restaurando su copia de seguridad de las mismas, con el disgusto de los alumnos que estuviesen interesados en mantener sus cambios. Por el contrario, si cada alumno tiene sus propias tablas, el profesor no tiene un mecanismo sencillo de propagar sus cambios sobre las mismas a los alumnos. Además los alumnos, serían responsables de recuperar las tablas con los contenidos que tenga el profesor en cada momento.

Por todo ello, vemos que ninguna de las soluciones planteadas son del todo satisfactorias como para impartir clases prácticas que versen sobre el comando SELECT.

2.2. Insert-Delete-Update y la Concurrencia

Los comandos de modificación del contenido de las tablas exigen que los alumnos tengan los permisos correspondientes.

Además, es necesario que el sistema disponga de, al menos, el nivel control de concurrencia *serializable* de SQL/ISO[6] para que los alumnos puedan experimentar con la concurrencia manteniendo -cada alumno- varias sesiones abiertas que accedan sobre los mismos datos. Si los alumnos comparten las mismas tablas, pueden bloquearse entre ellos si no cierran las transacciones, dejando en espera a sus compañeros. Por ello, parece más indicado que cada alumno tenga sus propias tablas.

Sin embargo, como ya se ha comentado en 2.1, si los alumnos trabajan con sus propias tablas, el profesor pierde el control sobre el estado de la base de datos de cada alumno, por lo que le es difícil proponer ejercicios en los que de antemano se sepa que el sistema va a producir el mismo resultado en todos los alumnos. De otro lado, si los alumnos comparten las mismas tablas que el profesor, pudiendo modificar su estado, además los problemas de concurrencia ya mencionados, tampoco el profesor controlaría el estado de la base de datos. Nótese que este caso es muy similar al de la SELECT, y tampoco se tiene una solución adecuada.

Tanto si se comparten o no las tablas del profesor, el crecimiento de las tablas y su ocupación de espacio de disco debe de estar acotado para no perjudicar a otros posibles usuarios del sistema, quizás alumnos de otras asignaturas.

2.3. DDL

Los contenidos a impartir sobre DDL son la creación de tablas, incluyendo restricciones, y vistas. Para que el alumno pueda experimentar se hace necesario que tenga privilegios para crear y eliminar este tipo de objetos. Debido a que los alumnos tenderán a crear tablas con nombres parecidos a los de sus compañeros, habría que habilitar algún mecanismo para que cada alumno tuviese un especio de nombres distinto. En sistemas como *Oracle*[5], donde cada usuario tiene su propio espacio de nombres, la solución es dotar a cada alumno con una cuenta distinta de usuario.

Asimismo, hay que controlar que las tablas que creen los alumnos no puedan crecer indefinidamente, asociándolas una cuota de disco máxima por defecto, y un espacio de disco que no interfiera con otros posibles usos de la base de datos. En este sentido, nuevamente es recomendable que cada alumno sea un usuario distinto; de lo contrario, un alumno puede eliminar o hacer crecer demasiado tablas de sus compañeros intencionadamente o por descuido.

Finalmente, el sistema que elijamos debe de permitir actualizar vistas fácilmente, pues la actualización de vistas es parte de los contenidos a impartir.

Como conclusión, las prácticas DDL *si* que tienen una respuesta adecuada, simplemente dando determinados privilegios a los alumnos, pero sin que puedan tocar las tablas del profesor. La única cuestión relevante es tener un sistema con espacios

de nombres diferenciados por cada usuario, y con posibilidad de actualización de vistas.

3. La Aproximación Basada en Versiones de Tablas

En el apartado anterior se ha visto que el dotar a los alumnos de sus propias tablas origina ciertos problemas, pero que el compartirlas también. La aproximación basada en versiones de tablas que se propone resuelve en buena parte el problema.

Una versión de tabla *T* (tabla base) consiste en otra tabla *T'* (tabla/versión derivada) con la misma estructura y restricciones que *T*, pero con distinta extensión [7]. Las versiones de tablas ya son contempladas por algunos sistemas como *Postgres* [7]. En cierta forma, las vistas son muy parecidas a las versiones. Los cambios de extensión en la tabla base se propagan a las vistas y versiones derivadas. Lo que diferencia a versiones y vistas, es que los cambios en la versión no se propagan a la tabla base, a diferencia de las vistas actualizables.

Nuestra aportación consiste en la aplicación de estos conceptos para obtener un sistema de prácticas de SQL, sobre el que el profesor pueda tener un cierto control sin constreñir demasiado las posibilidades de experimentación libre del alumno. Para ello, las tablas del profesor serán tablas base, mientras que las tablas sobre las que los alumnos experimenten serán versiones derivadas de las tablas del profesor. Los alumnos podrán consultar, pero no modificar, las tablas base; mientras que sí podrán modificar el estado de sus versiones.

Con las versiones de tablas no hay problemas de bloqueos indeseados por concurrencia, pues cada alumno sólo hace cambios sobre sus versiones, y las tablas del profesor se acceden sólo en lectura. Sin embargo, se puede experimentar con concurrencia, sin más que el alumno mantenga varias sesiones simultáneas sobre el mismo usuario, esto es, sobre sus propias versiones.

El profesor tiene control absoluto sobre sus tablas, que no pueden modificarse por los alumnos, pero además puede introducir cambios en las versiones de los alumnos, como efecto de la propagación de los cambios que él haga sobre sus tablas base. Puede introducir filas nuevas, cambiar contenidos de las filas de sus tablas, o eliminarlas, y esos cambios van a propagarse a las versiones que tengan los alumnos, con lo que el profesor puede hacer ciertas suposiciones sobre el estado de las versiones, que le permitan, con mucha probabilidad, saber cual va a ser de antemano la respuesta del sistema, incluso en las distintas versiones de los alumnos.

Por tanto, la aproximación de versiones de tablas es la más adecuada para poder impartir clases guiadas de SQL interactivo, el problema es que no cualquier sistema permite implementarla.

4. Implementación Genérica

La implementación de un sistema basado en versiones de tablas ya viene descrita por encima en [7]. La idea es que por cada tabla que se versione (por ejemplo *Clientes*), existan dos nuevas tablas soporte (en nuestro caso *vadd_Clientes* y *vdel_Clientes*).

La tabla *vadd* tiene la misma estructura que la tabla base, y contiene:

- Las filas que se inserten por el alumno en la versión de la tabla.
- Las filas que el alumno haya cambiado con los valores nuevos.

La tabla *vdel* sólo contiene un identificador que haga referencia a las filas de la tabla base. En el caso de los sistemas objeto-relacionales, como *Postgres*, dicho identificador es el OID (*Object Identifier*). Este identificador indicará:

- Las referencias a filas que los alumnos han eliminado de la tabla base.
- Las referencias a filas de la tabla base que los alumnos hayan cambiado.

Todo ello, permite definir la versión como una vista tal y como muestra la figura.

```
CREATE VIEW VersionClientes AS
(SELECT OID as _OID, CIF, Nombre,etc…
FROM Clientes
WHERE NOT EXISTS
       (SELECT * FROM vdel
       WHERE oid=Clientes.oid)
UNION
(SELECT OID as _OID, CIF, Nombre,etc…
FROM Clientes);
```

Ilustración 1. Vista que implementa una versión de la tabla Clientes.

Mediante *triggers* se controla la actualización de esta vista haciendo:

Caso 1. Que cuando el alumno realice una inserción sobre la versión, el sistema realmente lo que hace es insertar la fila sobre *vadd*. Ver **Ilustración 2**.

Caso 2. Cuando se realiza una eliminación de una fila de la tabla base, lo que se hace realmente es apuntar esa fila como borrada en *vdel*. Ver **Ilustración 3**.

Caso 3. Si se modifica una fila de la tabla base por un alumno, por primera vez, se trata como una baja de la fila con los valores viejos (la fila antes de la modificación es registrada en *vdel*), más un alta de la misma con los valores nuevos (la fila que se obtiene después de la modificación es apuntada en *vadd*). Ver **Ilustración 4**.

Caso 4. Si se elimina una fila que se haya insertado o modificado desde la versión (está en *vadd*), se elimina directamente de *vadd*. Ver **Ilustración 3**.

Caso 5. Si se modifica una fila insertada por el alumno en la versión, o una fila de la tabla base ya modificada por el alumno en la versión (está en *vadd*),simplemente se hace el cambio directamente sobre la fila en *vadd*. Ver **Ilustración 4**.

El tipo de *triggers* que se necesitan para hacer esta implementación necesitan el modo de disparo *instead*, que permite hacer que la acción del *trigger* se ejecute sustituyendo al evento que lo dispara. Es decir, si el evento es una operación de modificación (p.e. una inserción) sobre la vista, esta operación no se ejecuta, y en su lugar se ejecuta la acción especificada en el *trigger* (p.e. la inserción sobre *vadd*). Los *triggers instead* no están disponibles en la mayoría de los sistemas actuales, lo que limita el número de gestores en el que se puede aplicar la aproximación de versiones de tablas.

2. Implementación de la Aproximación de Versiones en PostgreSQL

En el presente apartado se muestra la implementación en PostgreSQL.

Regla de inserción:

Implementa el Caso 1 del apartado anterior.

```
CREATE RULE versionClientes_ins AS
ON INSERT TO versionClientes
DO INSTEAD
//Caso 1
INSERT INTO vadd_Clientes
  CIF, nombre, etc…) VALUES
  (NEW.CIF, NEW.nombre, etc…);
```

Ilustración 2. Regla de Inserción

Regla de borrado:

Implementa los Casos 2 y 4 del apartado anterior.

```
CREATE RULE versionClientes_del AS
ON DELETE TO versionClientes
DO INSTEAD (
//Caso 2
INSERT INTO vdel_Clientes(DOID)
   SELECT Clientes.OID FROM Clientes
   WHERE OLD._OID = Clientes.OID
     AND OLD._OID NOT IN
    (SELECT DOID FROM vdel_Clientes);
//Caso 4
DELETE FROM vadd_Clientes
   WHERE vadd_Clientes.oid = OLD._oid;
);
```

Ilustración 3. Regla de Borrado

Un inconveniente de la implementación *PostgreSQL* es que las vistas no tienen OID. Ni siquiera tienen el OID de la tabla de la que se derivan, por lo que es necesario definir un campo en la vista que deliberadamente contenga el valor del OID de las filas que la componen (el campo _OID en **Ilustración 1**). Este campo lo llamaremos _OID y como se puede ver es utilizado en la Regla

de Borrado y en la de Actualización (ver más adelante) por los *triggers*.

Otra observación interesante es que *vdel* como tabla, ya tiene su propio OID, por ello es necesario que su único campo, que contiene los OIDs de las filas borradas/modificadas, tenga un nombre distinto de OID. En nuestro cado hemos elegido como nombre DOID, que también se utiliza en las Reglas de Borrado y Actualización.

Regla de actualización:

Implementa los Casos 3 y 5 del apartado anterior.

```
CREATE RULE versionClientes_upd AS
ON UPDATE TO versionClientes
DO INSTEAD(

//Caso 3
INSERT INTO vadd_CLIENTES
  (CIF, nombre, telefono)
   select NEW.CIF, NEW.nombre, etc…
   FROM Clientes
   WHERE old._OID = Clientes.OID;

INSERT INTO vdel_Clientes(DOID)
  SELECT oid FROM Clientes
  WHERE old._oid = Clientes.oid;

//Caso 5
UPDATE vadd_Clientes
SET CIF=NEW.CIF, nombre=NEW.nombre,
telefono=NEW.telefono
WHERE vadd_Clientes.oid=OLD._oid;
);
```

Ilustración 4. Regla de Actualización

La implementación mostrada sólo concierne a una de las tablas y a uno de los usuarios. Para que el entorno sea eficaz, deberá de crear automáticamente las vistas y *triggers* correspondientes para cada alumno y versión de tabla del profesor. Este proceso se puede automatizar a través de *scripts* del sistema operativo que consulten el catálogo de la base de datos del profesor. Estos *scripts* se ejecutarían al dar de alta a cada usuario alumno, al principio del curso.

Así, para una tabla *Clientes* del profesor, este *script* crearía automáticamente la versión *Clientes_1* al dar de alta al alumno 1, *Clientes_2* al dar de alta el alumno 2, etc… De esta forma el alumno *n* puede disponer de la versión del profesor –*Clientes*- y de la suya propia –*Clientes_n*.

Un problema que hemos pasado por alto es el de las restricciones. En principio, parece que para que las versiones tengan las mismas restricciones que sus tablas base, basta con volver a declarar dichas restricciones en *vadd*. Sin embargo, esto no es así para las restricciones SQL de PRIMARY KEY, UNIQUE y FOREIGN KEY. Las claves primarias y candidatas han de mantener su unicidad no sólo en la tabla base y en *vadd* por separado, sino que deben tener valores únicos para la unión de las extensiones de ambas tablas. Con las claves foráneas ocurre lo mismo. Si por ejemplo una fila de la tabla *Pedidos* referencia a la tabla *Clientes*, no basta con declarar que *vadd_Pedidos* referencie a *vadd_Clientes*, pues debe poder referenciar además a un cliente de la tabla base *Cientes*.

La solución a estos problemas es controlar las violaciones de estas restricciones, nuevamente a través de *triggers* declarados, en esta ocasión, sobre eventos en *vadd*. En el caso de las claves foráneas los *triggers* controlaran que las inserciones y modificaciones de *Pedidos* y las eliminaciones de *Clientes* no dejan a los pedidos sin cliente asociado. Esta implementación mediante *triggers* es muy flexible, permitiendo implementar distintas semánticas (rechazos, acciones en cascada etc.). En el caso de las claves primarias y candidatas, los *triggers* controlarán que las inserciones y modificaciones de la clave en *vadd* no generen duplicados.

Todos estos *triggers* también pueden ser creados directamente en el *script* de creación de usuario. Aunque las malas noticias son que todos estos controles por *trigger* acaban por degradar el rendimiento del sistema, las buenas noticias son que con el escaso número de filas que suelen tener las bases de datos de docencia, este efecto no es apreciable.

Parte de la materia de SQL interactivo, es el trabajo con vistas actualizables. Un problema de la versión actual de *PostgreSQL* es que las vistas que definan los alumnos no son actualizables, a no ser que se especifiquen vía *triggers*, las acciones asociadas a la propagación de las modificaciones sobre la vista a las tablas base. Sin embargo, este

problema tiene solución, como se indica en el apartado de trabajos futuros.

5. Conclusiones y Trabajos Futuros

La docencia en SQL es recomendable abordarla a través de clases prácticas. El entorno en el que se realicen las prácticas y su configuración es muy importante para poder aplicar metodologías docentes que permitan al alumno experimentar por sí mismo y que sean compatibles con la compartición de un esquema de base de datos inicial del profesor. Se han identificado posibles problemas y soluciones de las aproximaciones tradicionales de cómo abordar las prácticas de SQL, teniendo en cuenta si se trata de explicar el comando SELECT, los comandos INSERT-DELETE-UPDATE o el DDL. Los problemas detectados tienen su orígen en que unas soluciones dan demasiada libertad al alumno, y otras son demasiado restrictivas.

Las versiones de tablas han demostrado que son una solución que mantienen un punto de equilibrio entre dar demasiada libertad al alumno, y otorgar un cierto control al profesor sobre el contenido de los datos.

Un problema de esta aproximación es que no es posible implementarla en cualquier tipo de Sistema de Base de Datos, siendo necesario que el sistema soporte *triggers* en modo *instead*, y que los triggers puedan definirse a partir de eventos que surjan durante el intento de actualización de una vista. Se ha dado una implementación para *PotgreSQL* que es un sistema gratuito y fiable que sí tiene estas características en cuanto a *triggers*.

Un inconveniente de la aproximación que se ha implementado es que no es directamente utilizable desde web, por lo que no sirve para que el alumno pueda practicar desde casa, salvo que le proporcionemos una entrada *Telnet* a lo que probablemente se opongan los responsables de seguridad de nuestro sistema. Por ello, estamos trabajando en un cliente web que permita interactuar con el sistema. Nuestro propósito final es que dicho cliente web admita las consultas en álgebra y cálculo relacional; lenguajes de consulta que le son propios a la misma asignatura de Bases de Datos en la que se revisa el SQL interactivo. Para poder implementar esta mejora desarrollaremos un *parser* en el lado del servidor que traduzca las consultas a *PostgreSQL*. Otra función futura del *parser* será la creación transparente al alumno de los *triggers* necesarios para la actualización de vistas definidas por los alumnos sobre tablas y versiones de su usuario.

Agradecimientos

Por su colaboración a los alumnos Laura Izquierdo Rodrigo y Alberto Tovar González.

Referencias

[1] P. Blesa, N. Brisaboa, V. Canivel, J. Garbajosa, J. Maudes, M. Piattini, A. Urpi. *Contenidos en Bases de Datos de los Planes de Estudio de Informática*. Actas de la III JIDB. Eds. J.C. Casamayor, M. Celma, L. Mota, M.A. Pastor. DSIC, Universidad Politécnica de Valencia. Valencia, 1998.

[2] P. Blesa, N. Brisaboa, V. Canivel, J. Garbajosa, J. Maudes, M. Piattini. *Propuesta de Contenidos en Bases de Datos de los Planes de Estudio de Informática*. Novática, Enero/Febrero 1999. pp 60-63.

[3] Dietrich, S. W. *An Educational Tool for Formal Relational Query Languages*, Computer Science Education. Vol 4, pp.157-184, 1993.

[4] Dietrich, S. W., Eckert, E., Piscator, K., *WinRDBI: A Windows-Based Relational Database Educational Tool*, Proceedings of the 28th ACM SIGCSE Technical Symposium on Computer Science Education, San Jose, California, pp. 126-130, de 27-02 a 01-03 de 1997.

[5] Loney K., Koch G. *Oracle 8i the Complete Reference*. Oracle Press 2000.

[6] Melton J., Simon A.R. *Understanding the new SQL: a Complete Guide*. Morgan Kaufmann 1993.

[7] Potaminos S. & Stonebraker M. *The Postgres Rule* System. En Widom, J & Ceri, S. *Active Database Systems. Triggers and Rules For Advanced Database Processing*, Morgan Kaufmann Publishers, Inc. San Francisco, California. 1996.

SQL Programado desde Java: profundización en el diseño y acceso a bases de datos consolidando el conocimiento del lenguaje

Josep Maria Marco Simó
Profesor de los Estudios de Informática y Multimedia de la UOC
e-mail: jmarco@uoc.edu

Resumen

Se presenta la experiencia de desarrollo e implantación de los contenidos sobre SQL programado desde Java para la asignatura *Bases de Datos II*, asignatura que se imparte en los estudios de Informática de las Ingenierías Técnicas y Superior de la *Universitat Oberta de Catalunya -UOC-*.

Nuestra universidad se ha planteado la utilización de este lenguaje para abordar el tema de la programación de SQL dada su evidente implantación en entornos reales y la preferencia que otorga al aprendizaje de este lenguaje.

Este desarrollo, sin embargo, se ha enfocado con una serie de objetivos paralelos que van más allá del propio aprendizaje sobre SQL programado desde Java, y que pretenden que el estudiante desarrolle también otros conocimientos y habilidades adquiridos con anterioridad, en cuanto a:

- El diseño y uso de componentes lógicos de base de datos, así como su acceso.
- La perspectiva y/o introducción al mundo Java.
- El modelo SQL/CLI.

Exponemos en este documento la génesis, el diseño y la implementación de esta propuesta, así como sus resultados y sus perspectivas de futuro.

1. Motivación

En los planes de estudio de las Ingenierías Técnicas de Informática –especialidades Gestión y Sistemas- el aprendizaje sobre bases de datos se inicia en la asignatura obligatoria *Bases de Datos I* [1], cuyos contenidos se resumen en:

- Ficheros.
- Introducción a las Bases de Datos.
- Modelo relacional y álgebra relacional.
- Introducción a SQL.
- Introducción al diseño de Bases de Datos.

En *Bases de Datos II* [2], obligatoria para la Ingeniería Técnica de Gestión y optativa para la Ingeniería Técnica de Sistemas y la Ingeniería Superior, se amplía este aprendizaje mediante el estudio de:

- Componentes lógicos de datos y de control (SQL avanzado).
- Componentes de almacenamiento.
- Implementación de métodos de acceso.
- Transacciones.
- Bases de datos distribuidas y cliente/servidor.
- Programación con SQL (SQL hospedado y SQL-CLI)

En ambas asignaturas se realizan prácticas. En la primera están centradas en el diseño de bases de datos y en el uso de SQL. En la segunda se pretende consolidar dos aspectos fundamentales que refuerzan, a su vez, lo aprendido anteriormente: la técnica CLI de SQL programado [3] y el uso avanzado de componentes lógicos de datos (tablas, vistas y restricciones) y componentes lógicos de control (procedimientos, disparadores, transacciones, control de acceso e índices) [4]

En el origen de la asignatura *Bases de Datos II*, los modelos de SQL programado se presentaban con aplicaciones en lenguaje C.

Aunque ésta es una aproximación que permite entender con claridad las características estándares de los modelos hospedado y CLI, también es cierto que presenta cierta complejidad en su implementación práctica.

Así, dado que el lenguaje Java es uno de los objetivos y de los lenguajes de referencia de los estudios de Informática de la UOC, se planteó ampliar las perspectivas sobre las técnicas de SQL programado, presentando también el acercamiento de Java al tema. Este acercamiento, por un lado, es relativamente más simple que el tradicional modelo en C y, por otro, permite introducir a los estudiantes en un tema de amplia implantación en los entornos profesionales: sólo hay que tener en cuenta su importancia en los aplicaciones de Internet y en los sistemas distribuidos en general.

Aquí pues tuvo su origen nuestra propuesta: presentar un modelo más actual y más sencillo para la programación con SQL.

Sin embargo, había que tener en cuenta otros aspectos:

En los primeros cuatrimestres en que se iba a implantar esta propuesta, iba a servir también como vía para introducir a muchos estudiantes en el lenguaje Java. Esto sería debido a que, al tratarse de una asignatura de los últimos cursos de las Ingenierías Técnicas y de los centrales de la Ingeniería Superior, muchos de ellos no conocerían el lenguaje por haber empezado los estudios cuando Java todavía no aparecía en las asignaturas de inicio.

Adicionalmente, se pretendía reincidir en las destrezas de diseño adquiridas anteriormente en *Bases de Datos I* y cuyo refuerzo pasa siempre por la vía de los ejercicios prácticos.

Teniendo en cuenta todas estas consideraciones, se confeccionaron unos contenidos adicionales [5] a la asignatura *Bases de datos II* con la idea de integrarlos. En resumen, los objetivos que se pretendían cubrir eran:

1) Presentar la programación SQL desde Java.
2) Profundizar en el diseño, uso de componentes lógicos y acceso a bases de datos.
3) Ofrecer una perspectiva y/o introducción al mundo Java.
4) Presentar a los estudiantes un sistema simplificado del modelo SQL/CLI.

2. Construcción de la propuesta

El proceso seguido para construir la propuesta tuvo dos partes diferenciadas. En primer lugar se verificaron las posibilidades prácticas de la programación SQL en Java. Una vez constatada la problemática subyacente, se planteó cuál sería un método válido para su exposición teórica.

2.1. Los componentes prácticos

Dado el carácter introductorio que se pretendía dar a los contenidos y la facilidad del modelo, se optó por presentar JDBC, la propuesta CLI de Java, en lugar de otras propuestas Java desarrolladas por los propios fabricantes de SGBD. Estas propuestas alternativas resultan ser mucho más específicas y, evidentemente, mucho menos estándar.

Como es conocido, JDBC es un API de Java que simplifica el modelo CLI (fuertemente ligado al interfaz ODBC) y lo extiende siguiendo los principios del lenguaje. Entre las simplificaciones que aporta cabe destacar:

- El proceso de conexión a la base de datos.
- La gestión del entorno de ejecución de sentencias
- La creación y uso de sentencias SQL parametrizadas.
- La gestión de cursores y la recuperación de resultados mediante éstos.
- La oferta de métodos que van más allá del propio SQL.

En el momento de la elaboración de estos contenidos, la especificación del JDBC se hallaba en su segunda versión e incluía ya, entre otras cosas, soporte para tipos SQL:1999 (el correspondiente al *draft* SQL3), movimiento libre de cursores, modificaciones *batch*, métodos específicos de manipulación de filas y tablas, etc.

Para la aplicación práctica de los contenidos, era necesario disponer de una implementación del API (implementación que se conoce como *driver*) dependiente del SGBD utilizado. Estos *drivers*, adicionalmente, tienen diferentes tipologías atendiendo al grado en que suscriben un modelo de tres capas.

En nuestra universidad, y hasta el momento, el SGBD suministrado para estas asignaturas es *Informix*. El mismo fabricante ofrece una

implementación de libre distribución del JDBC 2.0. En ese mismo sentido, la *web* de *Sun* ofrece un sistema de localización de *drivers* para un elevado número de SGBD comerciales [6].

El entorno Java utilizado fue el Java 2 (JDK 1.2 y posteriormente JDK 1.3) de *Sun*. Se eligió *Windows 9x* como plataforma de desarrollo por ser la habitual para el uso de nuestro campus virtual.

Desde el momento en que se pudo disponer de las herramientas prácticas se empezó a verificar sus capacidades. Como es habitual, fue necesario detectar aquellos puntos en los que la realidad de nuestro entorno (*Informix + Windows 9x + Java 2 + driver JDBC Informix type 4)* no seguía las especificaciones estándar. En esa misma línea, se verificó el comportamiento de otros productos SGBD, de diferentes tipos de *drivers* y de varias plataformas operativas.

Tras este trabajo se estaba en condiciones de determinar qué capacidades podían formar parte del eje de una descripción generalista del modelo.

2.2. La exposición teórica.

Entendiendo que inicialmente el material que se debía elaborar iba a ser complementario a otro suficientemente largo y complejo, se optó por una exposición basada en ejemplos y en la comparación con los modelos de SQL/CLI en C ya estudiados. No se pretendía, pues, rechazar la experiencia de la que se disponía utilizando el modelo en C, si no de que ésta, por comparación, subrayase las posibilidades de Java. Se trataba, pues, de reforzar una serie de aspectos básicos:

- La apuesta por el modelo CLI que evita la dependencia de base con el SGBD.
- La simplificación del modelo CLI de JDBC en relación al modelo en C tradicional.
- La comprensión en profundidad de los fundamentos de dicho modelo al liberarlo de las complejidades artificiales que inducen otros lenguajes.
- La justificación teórica de los elementos prácticos.
- La integración del modelo JDBC con la filosofía de Java.

De esta forma, la exposición empieza detallando los fundamentos teóricos que justifican el modelo JDBC, para seguir con la presentación de la estructura de un programa sencillo basado en dicho modelo:

- Localización y carga del driver.
- Conexión con el SGBD.
- Operaciones con el SGBD y tratamiento de excepciones.
- Desconexión del SGBD.

Para cada punto se especifica cómo realizarlo en el entorno de trabajo de referencia (variables de sistema, configuraciones de las herramientas, librerías requeridas por el lenguaje, etc.)

El programa presentado utiliza una sencilla interfaz texto para no mezclar las complejidades de un entorno gráfico con las necesidades JDBC. Este programa pasa a utilizarse entonces como una aplicación patrón y a partir de ella se construyen todos los ejemplos que introducen los diferentes aspectos detallados:

- Sentencias de modificación.
- Sentencias de consulta.
- Cursores con movimiento libre.
- Control de transacciones.
- Sentencias preparadas.
- Procedimientos almacenados.
- Modificaciones por lotes.

Se intenta también perfilar con claridad cuáles son las clases del API protagonistas de la estructura básica del patrón, así como sus métodos elementales y los que las interrelacionan (fig.1).

Finalmente se revisan las capacidades adicionales del modelo JDBC en aspectos relacionados con los tipos SQL:1999.

En definitiva, la idea subyacente es que la lectura motive al estudiante a probar los contenidos expuestos. De hecho, cada uno de los aspectos puede incorporarse a la aplicación patrón y probarse contra una pequeña base de datos de ejemplo.

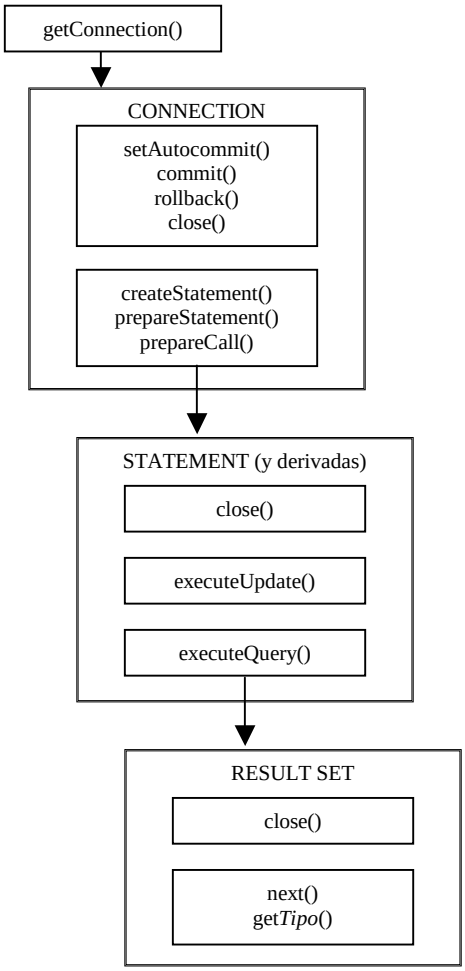


Fig.1: Clases y métodos elementales de JDBC

3. Implantación y aspectos metodológicos de la propuesta

En los tres cuatrimestres que lleva en funcionamiento esta propuesta, se ha ido ajustando atendiendo a requisitos académicos y, principalmente, a criterios metodológicos destinados a facilitar y asegurar el aprendizaje.

3.1. Redistribución de los contenidos

Como se ha dicho, *Bases de datos II* tiene un capítulo en el que se exponen las características del SQL programado. Los contenidos sobre JDBC se han suministrado como un complemento a este material.

Esto ha provocado que la carga del tema se haya incrementado considerablemente. Por esta razón se está optando por presentar las técnicas de SQL hospedado y SQL/CLI en C desde una perspectiva meramente generalista e introductoria, poniendo en relieve su filosofía, sus condicionantes y su estudio comparativo. Con esta base adquirida, es el tema práctico desde la perspectiva JDBC el que adquiere la máxima relevancia.

3.2. Obligatoriedad de la práctica

En un principio, la práctica no era obligatoria. Sin embargo, sí se hizo una amplia difusión entre los estudiantes de la importancia de afrontarla. Esto, unido con que tenía un peso del 20% de la nota final, produjo que la gran mayoría de los estudiantes la presentaran.

Como consecuencia de esto y de que se había confirmado como una herramienta excelente para consolidar paulatinamente los objetivos de aprendizaje, se decidió convertirla en obligatoria. Para conseguir además que realmente fuera un trabajo continuo en el tiempo, se introdujo la entrega en dos partes, inicialmente con la primera entrega voluntaria (acumulándose toda la entrega en la segunda si no se realizaba la primera) y posteriormente con las dos entregas obligatorias en el tiempo.

También parecía claro que lo importante no era escatimar el tiempo dado a los estudiantes entre la entrega del enunciado y la recogida de las soluciones y, así, se ha llegado a un intervalo aproximado de ocho semanas, en unos cursos que tienen una duración lectiva de alrededor de trece semanas.

3.3. Detalle del caso objeto de la práctica

También el contenido del caso de la práctica ha pasado de ser muy guiado (dando el diseño de la base de datos) a dejarse prácticamente abierto (mediante la exposición descriptiva de un caso), y planteando una serie de preguntas-objetivo que han de constituir el núcleo de la respuesta del estudiante.

Con esta apertura del caso hemos reforzado el objetivo de desarrollar las capacidades del estudiante en cuanto a:

- Diseño conceptual de la base de datos, aprendido en *Bases de Datos I*. Este aprendizaje previo es objeto de uso en muchas de las asignaturas de la carrera basadas en casos o proyectos, sin mencionar, por supuesto, la relevancia que tiene en la práctica profesional.
- Decisión sobre el uso adecuado de los componentes lógicos (de control y de datos) presentados. El estudiante ha de ser capaz, por ejemplo, de ver cuáles son las vías para implementar una restricción y elegir la que le parezca más adecuada.

3.4. Trabajo en grupo de la práctica

Al principio, el trabajo de la práctica se planteó individualmente, cosa que se justificaba por la voluntariedad de la misma. Eliminado este condicionante se optó por permitir que aquellos estudiantes que así lo quisieran pudieran entregar la solución en grupos de dos personas.

No se ha establecido éste como un criterio obligatorio por dos motivos básicos:

- Los estudiantes de nuestra universidad ya tienen un elevado número de asignaturas donde el trabajo en grupo es obligatorio con lo que se cubre el objetivo de dar prioridad a este aspecto.
- Es interesante que las tareas de diseño de la base de datos de un caso complejo se puedan pensar también individualmente. Si en la práctica profesional es habitual trabajar en grupo, también lo es que, en situaciones más o menos comunes, haya que tomar decisiones de diseño individualmente (y no sólo sobre aspectos sencillos).

3.5. Soporte Java

En cuanto al soporte que se da al estudiante para enfrentarse al lenguaje, se queda básicamente en el programa patrón y en las indicaciones de configuración e instalación dadas en la exposición teórica de los contenidos.

Se explica también la importancia de la documentación y los tutoriales sobre el API JDBC suministrado en el JDK [7,8]. Además se les recomienda tener a mano la documentación del propio fabricante del *driver* para *Informix* [9].

Una vez más, se prima que el estudiante indague por su cuenta, como pasa en la realidad, las capacidades de un nuevo contenido en el que se introduce. Evidentemente, no se está pidiendo un nivel elevado sobre el uso del lenguaje y, en consecuencia, para cumplir los mínimos no es necesario un gran trabajo de investigación.

Se entiende también que, dado que cada vez hay menos estudiantes de planes antiguos, estos estudiantes ya están introducidos en los elementos básicos de la orientación a objetos. Además, llegados a esta altura de sus estudios (donde, en buena lógica, deben haber cursado asignaturas con un fuerte contenido en orientación a objetos como *Fundamentos de Programación II, Ingeniería del Software I* o *Técnicas de Desarrollo de Software)* es de suponer que los estudiantes han de tener recursos suficientes para seguir avanzando.

4. Resultados

En general se ha observado una acogida muy buena de los contenidos propuestos por parte de los estudiantes. Esencialmente tenemos a nuestro favor varios hechos:

- Las asignaturas sobre bases de datos tienen habitualmente una consideración positiva por parte del estudiante.
- La programación en SQL es percibida como un componente fundamental para el trabajo con bases de datos.
- Java abre las puertas a otros aspectos muy bien apreciados: la elaboración no sólo de aplicaciones sino de *applets*, el mundo cliente/servidor o de Internet y el acceso remoto o distribuido.

Por otro lado, en el primer cuatrimestre se detectaron algunos problemas, principalmente técnicos, derivados del uso del *driver* y del JDK sobre plataformas que no eran las recomendadas o que no estaban en buenas condiciones de mantenimiento. Estas situaciones se han repetido

cada vez menos ya que se han ido dando indicaciones muy claras al respecto. Adicionalmente, se han ido realizando tareas periódicas de verificación con los nuevos entornos que han ido apareciendo.

4.1. Consecución de objetivos

En cuanto al objetivo básico referido a la programación en SQL desde Java, creemos que se ha logrado ampliamente. A esto contribuye la complejidad de los casos prácticos propuestos que obligan al estudiante a plantearse soluciones para acceder intensiva y extensivamente a la base de datos, imaginando estrategias para una programación eficiente. Así, acaba interrogándose sobre cómo mejorar aspectos concretos, lo que le lleva a profundizar en las posibilidades y limitaciones del sistema.

Del mismo modo, el objetivo referido al diseño de base de datos y al uso de componentes lógicos (de datos y de control) se puede considerar alcanzado, al observar, una vez más, la complejidad de los casos desarrollados. El carácter abierto del caso deja espacio a la profundización sobre todo un conjunto de situaciones diferentes. En la mayoría de los casos, el análisis de estas situaciones realizado por el estudiante ha sido muy completo y las soluciones aportadas han demostrado que ha entendido cómo lograrlas, tanto desde la perspectiva de diseño como desde la de implementación con los componentes lógicos.

Por lo que respecta al objetivo de la introducción al mundo Java, hemos constatado una serie de comportamientos distintos aunque previsibles:

- El estudiante con conocimiento de orientación a objetos y Java ha dedicado tiempo a mejorar aspectos del código de la programación así como del interfaz, aunque ello no era objetivo del curso ni mejoraba la puntuación.
- El estudiante con conocimiento de orientación a objetos pero sin experiencia en Java, no ha tenido problemas en cumplir lo exigido y, en la mayoría de casos, no ha percibido el lenguaje como un problema.
- El estudiante que, por formación académica antigua, sólo está introducido mínimamente a la orientación a objetos y que no tiene conocimiento de Java, ha sabido utilizar el programa patrón suministrado y llegar con él a todas las respuestas planteadas. Es cierto que, en algunos casos, no se ha planteado entender completamente cómo se justificaba la propuesta del patrón, con lo que se puede decir que ha visto el aspecto de un programa mínimo en Java, pero no que será capaz de elaborar uno desde cero. Sea como sea, con esta situación de partida, este resultado se puede considerar un hito relevante.

En general, pues, Java no ha resultado ser un elemento problemático, sino más bien al contrario. El interés por profundizar en él ha beneficiado la consecución de los otros objetivos de aprendizaje. Es decir, lo que en principio podía parecer más conflictivo se ha acabado convirtiendo en un motor para el seguimiento de los contenidos.

4.2. Otros efectos constatados

Como se ha ido demostrando en las diferentes asignaturas y carreras que imparte la UOC, en un entorno de aprendizaje a distancia y virtual la constancia en el estudio y en el mantenimiento de un ritmo regular de dedicación al mismo tienen una importancia si cabe mayor que en los entornos presenciales.

Así, el hecho de que la práctica sobre JDBC se plantee periodificada con dos puntos de entrega y que se disponga de un elevado número de semanas para resolverla ha facilitado el aprendizaje paulatino. Cuando no existía la obligación de una entrega intermedia, se produjeron bastantes casos de abandono ya que el estudiante dejaba acumular todo el trabajo al final, muy cerca de la fecha de entrega definitiva.

Un aspecto que ha aparecido espontáneamente ha sido el trabajo colaborativo entre los estudiantes. En nuestra universidad, la existencia de un espacio común de intercambio (el foro) ha facilitado que expusieran en público sus problemas para solicitar ayuda de los compañeros. La respuesta del grupo, en general, ha sido estimulante: en muchas ocasiones no se esperaban a que el profesor diera alguna indicación al problema sino que se ofrecían soluciones y experiencias particulares. Curiosamente, los problemas expuestos no han sido tanto de soluciones de diseño (los contenidos originales y

propios de cada estudiante), sino de temas más prácticos sobre la implementación con el lenguaje o con el *driver* y la configuración del entorno. La generación de este comportamiento colaborativo nos parece especialmente importante porque denota una capacidad para compartir experiencias que, de mantenerse como una constante en la actividad del estudiante, constituirá un valor muy preciado para el trabajo en grupo en un entorno profesional.

Las principales críticas de los estudiantes han venido por aspectos menos relacionados con nuestros objetivos: el SGBD utilizado (que resulta una herramienta didáctica suficiente pero que no representa un producto con mucha implantación actual), así como la dificultad para encontrar información adicional, incluso de las propias fuentes. Estas consideraciones se irán teniendo en cuenta progresivamente, como vía natural para incrementar la calidad de la propuesta.

4.3. Datos comparativos

Como colofón, hemos querido contrastar nuestra percepción sobre la aceptación de la propuesta entre los estudiantes, así como su efecto positivo en el rendimiento académico.

En el primer aspecto sólo podemos remitirnos a los comentarios públicos y voluntarios de los estudiantes, dado que no se ha realizado una encuesta formal entre ellos. De estos comentarios, y con las reservas que impone su origen, hemos creído detectar una elevada satisfacción con el aprendizaje adquirido a pesar de su complejidad y del esfuerzo que requiere. Incluso hemos constatado que algunos de los matriculados más recientes lo han hecho por recomendación de antiguos estudiantes.

En cuanto al rendimiento académico hemos resumido los datos sobre la nota media de cada cuatrimestre, así como las diferencias objetivas entre ellos, resumen que se concreta en la siguiente tabla:

	Media
Semestre 1: ▪ Voluntaria ▪ Una entrega ▪ Caso guiado. ▪ Estudiantes Ing. Técnicas	6.93
Semestre 2: ▪ Obligatoria ▪ Dos entregas (primera voluntaria) ▪ Caso abierto ▪ Estudiantes Ing. Técnic. y Superior	7.48
Semestre 3: ▪ Obligatoria ▪ Dos entregas ▪ Caso abierto ▪ Estudiantes Ing. Técnic. y Superior	7.98

A continuación, para intentar detectar si el incremento aparente en dicha nota puede considerarse significativo, hemos sometido los datos a un test de ANOVA (figura 2).

El resultado de este test indica que, en efecto, la nota media obtenida en el tercer semestre es significativamente superior (a un nivel de significación del 95%) a la nota media del primer semestre (los intervalos correspondientes son

One-way ANOVA: Sem1; Sem2; Sem3

```
Analysis of Variance
Source     DF        SS        MS        F        P
Factor      2     22,96     11,48     7,10    0,001
Error     153    247,29      1,62
Total     155    270,25
                                  Individual 95% CIs For Mean
                                  Based on Pooled StDev
Level       N      Mean     StDev  -+---------+---------+---------+-----
Sem1       29     6,931     0,873  (---------*---------)
Sem2       63     7,476     1,430               (------*-----)
Sem3       64     7,977     1,255                        (------*-----)
                                  -+---------+---------+---------+-----
Pooled StDev =    1,271           6,50      7,00      7,50      8,00
```

Fig. 2: Resultados del test de ANOVA

disjuntos), lo que corrobora nuestra impresión de mejora.

5. Perspectivas de futuro

Para terminar queremos esbozar unas líneas sobre cómo podría seguir avanzando esta propuesta.

En primer lugar, se constata que las herramientas de desarrollo rápido de aplicaciones (RAD) tienen un uso relativamente poco extendido entre los estudiantes y los ejercicios que se les exigen en diferentes asignaturas. Esto contrasta con la elevada implantación que tienen en los entornos profesionales.

Por este motivo, podría considerarse en un futuro, y dado que los estudiantes cada vez llegan con un conocimiento más profundo en Java y en orientación a objetos, realizar la práctica en un entorno gráfico y utilizando alguna de estas herramientas para Java. Las necesidades de los casos que se trabajan lo hacen perfectamente aplicable y justificable. De esta forma, a su vez, al estudiante podrá introducírsele en el código Java que generan este tipo de aplicaciones. En este código conviven objetos generados automáticamente y que representan clases sólo necesarias para la implementación del GUI, con los objetos que representan las clases que responden al problema planteado. Los modelos que implementan estas clases específicas para el interfaz gráfico, constituyen un conjunto muy importante sobre el que el estudiante debería tener alguna referencia a lo largo de sus estudios.

En segundo y último lugar, el acceso a bases de datos desde programación en Java es un aspecto que podría exportarse e incluirse extendidamente en otro tipo de asignaturas orientadas a la resolución de proyectos, a sistemas distribuidos o a aplicaciones Internet entre otras. Es decir, se podría llegar a reorientar el planteamiento y el uso de este material y que el modelo JDBC se explicara en un contexto de aprendizaje más general, no restringido sólo al de las bases de datos.

Referencias

[1] Sistac Planas, Jaume (Coord) y otros. *Bases de dades I*. Ed. UOC. Barcelona, 1999.

[2] Sistac Planas, Jaume (Coord) y otros. *Bases de dades II*. Ed. UOC. Barcelona, 1999.

[3] Rodríguez González, M.Elena. *Programació amb SQL* en: Sistac Planas, Jaume (Coord) y otros. *Bases de dades II*. Ed. UOC. Barcelona, 1999.

[4] Mallafré i Porta, Francesc Xavier. *Components lògics d'una base de dades* en: Sistac Planas, Jaume (Coord) y otros. *Bases de dades II*. Ed. UOC. Barcelona, 1999.

[5] Marco Simó, Josep Maria. *SQL i Java*. UOC. Material de la asignatura *Bases de dades II*, 2001.

[6] http://industry.java.sun.com/products/jdbc/drivers

[7] Sun Microsystems, Inc. (1996, 1997) *JDBC Guide: Getting Started* Documentación del JDK 1.2.

[8] Fisher, (M.) Sun Microsystems, Inc. (1997) *JDBC Database Access: Trail.* Documentación del JDK 1.2.

[9] Informix Sotware, Inc. (1997) *Informix JDBC Driver: Programmers Guide.* Documentación del driver JDBC ver. 2.11 **.**

Calidad y evaluación de la docencia

Mejoras en la calidad de la docencia dentro de la asignatura Fundamentos de Informática

Miguel A. Vega Rodríguez, Juan M. Sánchez Pérez, Juan A. Gómez Pulido
Departamento de Informática. Universidad de Extremadura
Escuela Politécnica. Campus Universitario, s/n. 10071 Cáceres. Spain
E-mail: mavega@unex.es, sanperez@unex.es, jangomez@unex.es. Fax: +34-927-257202

Resumen

La asignatura Fundamentos de Informática, impartida en la Universidad de Extremadura (UEX), pertenece al plan de estudios de la titulación de Ingeniería Técnica de Telecomunicaciones, especialidad Sonido e Imagen. Titulación que fue implantada recientemente en la UEX, curso 1998/1999. Los autores han sido responsables de dicha asignatura desde su inicio, y la han ido mejorando paulatinamente hasta llegar a la situación actual. En esta ponencia se presentan las mejoras docentes que los autores han aplicado en la asignatura desde su creación, con el ánimo de darlas a conocer a otros profesores, y buscando siempre el intercambio de ideas y experiencias. La ponencia no sólo presenta las mejoras en sí, sino que también indica cómo se han obtenido los fondos necesarios para llevarlas a cabo, cuestión también de especial interés.

1. Introducción

La asignatura Fundamentos de Informática (FI) es una asignatura obligatoria anual de 15 créditos (9 teóricos y 6 prácticos), que corresponden a 150 horas lectivas (90 teóricas y 60 prácticas), que se imparte en el primer curso de la titulación de Ingeniero Técnico en Telecomunicación, especialidad Sonido e Imagen (ITTSI), dentro de la Universidad de Extremadura.

Esta asignatura ha sido impartida por los autores de este trabajo desde la implantación en la UEX de la titulación de ITTSI, es decir, desde el curso académico 1998/1999. Describimos en este trabajo la filosofía del programa desarrollado durante los cuatro últimos años, sus principales puntos de interés, así como las experiencias docentes y conclusiones obtenidas. También destacamos las mejoras docentes que se han ido introduciendo en la asignatura a lo largo de este periodo, y cómo se han obtenido los fondos necesarios para llevarlas a cabo.

2. Contexto

2.1. Objetivos de la asignatura

Para la elaboración de los contenidos de la asignatura se ha tenido en cuenta el hecho de que se trata de la única asignatura, entre las troncales y obligatorias en el plan de estudios de ITTSI, que tienen los alumnos para adquirir conocimientos en el amplio campo de la Informática. Por este motivo se intenta dar una visión general y práctica de la Informática en sus dos vertientes: hardware y software. También se ha tenido en cuenta la relación de esta asignatura con las restantes del plan de estudios.

Los objetivos fundamentales que se pretenden conseguir con esta asignatura son:

- Introducir al alumno en el campo de la Informática.
- Conocer en profundidad la organización y estructura de un computador, centrándose en la representación de la información y en el estudio de los sistemas digitales.
- Profundizar en la metodología de la programación, con el objetivo de desarrollar algoritmos y programas en C, así como conocer y manejar las estructuras de datos que se utilizarán en los programas.
- Introducir los conceptos básicos sobre sistemas de comunicación o teleinformáticos.

2.2. Programa de la asignatura

El programa está dividido en dos grandes módulos con los que se pretenden alcanzar los objetivos antes mencionados. Los dos primeros objetivos se consiguen gracias al módulo I, mientras los dos restantes se alcanzan con el módulo II.

Al tratarse de una asignatura anual y ser ambos módulos de igual importancia, cada módulo es impartido en un cuatrimestre, es decir, se realiza la impartición de los cinco primeros temas en el primer cuatrimestre y el resto en el segundo. La relación de módulos es la siguiente:

- Módulo I: Introducción. Representación de la Información. Sistemas Digitales.
- Módulo II: Programación. Estructuras de Datos. Sistemas de Comunicación.

El primer módulo está compuesto por los temas que se indican a continuación:

- Tema 1. Introducción a la Informática. El computador.
- Tema 2. Representación de la información. Datos numéricos y alfanuméricos.
- Tema 3. Especificación e implementación de sistemas combinacionales.
- Tema 4. Módulos combinacionales básicos.
- Tema 5. Especificación e implementación de sistemas secuenciales.

Por otro lado, el módulo segundo consta de los seis temas siguientes:

- Tema 6. Topología de la programación.
- Tema 7. Lenguaje de programación C. Programación básica.
- Tema 8. Organización de los datos en la memoria central.
- Tema 9. Lenguaje de programación C. Programación avanzada.
- Tema 10. Ficheros y bases de datos.
- Tema 11. Introducción a la teleinformática.

Para obtener información más detallada sobre el programa de la asignatura el lector puede consultar la referencia [5].

2.3. Implementación del programa

La tabla 1 resume la distribución horaria que, aproximadamente, se emplea para la asignatura de Fundamentos de Informática. En esta tabla aparecen reflejados los dos módulos del programa, subdivididos en temas, indicando la carga lectiva por tema. Obsérvese que se respeta tanto el número de créditos teóricos como prácticos que posee la asignatura.

DISTRIBUCIÓN HORARIA			
TEMAS	HORAS		
	TEÓRICAS	PRÁCTICAS	TOTALES
Módulo I			
Tema 1	2	0	2
Tema 2	8	4	12
Tema 3	12	12	24
Tema 4	9	7	16
Tema 5	14	7	21
Totales Módulo I	**45**	**30**	**75**
Módulo II			
Tema 6	7	4	11
Tema 7	12	11	23
Tema 8	7	2	9
Tema 9	8	10	18
Tema 10	7	2	9
Tema 11	4	1	5
Totales Módulo II	**45**	**30**	**75**
Total Asignatura	**90**	**60**	**150**

Tabla 1. Distribución horaria por temas de FI

Remarcamos el carácter aproximado de la tabla 1. Partir exclusivamente de unos contenidos preconcebidos no conduce a ningún resultado positivo. Igualmente hay que valorar la respuesta del alumno a los contenidos que se van impartiendo, las preguntas que formulan, cómo las formulan, los ejercicios que resuelven, las dudas que tienen, etc. como un indicador dinámico de su nivel de asimilación. De aquí que el programa deba considerarse como un marco de referencia general, susceptible de ser dinámicamente ampliado o simplificado, en puntos determinados de su desarrollo, según lo aconsejen las circunstancias de cada momento.

3. Mejoras docentes

3.1. Edición del libro "Fundamentos de Informática"

Aunque cada vez sean más los profesores que ofrecen a sus alumnos el material adecuado (apuntes) para el seguimiento correcto de las clases. Todavía se imparten clases magistrales en las que a los alumnos no se les entregan apuntes de apoyo, y en las que la exposición de los

conceptos se realiza simplemente de manera oral (incluso "dictada") y con ayuda de la pizarra.

Sin embargo, pensamos que con esta metodología docente se corre el riesgo de explicar los conceptos teóricos con demasiada celeridad desde el punto de vista del estudiante. Para el alumno es difícil seguir el ritmo de clase, es decir, mantener la concentración durante toda una hora. Además, le cuesta captar todas las explicaciones dadas oralmente, al mismo tiempo que copia datos escritos en la pizarra. Como consecuencia, cada alumno puede llegar a tener su propia versión de los conceptos teóricos, algunos alumnos no llegan a adquirir todos los conceptos, etc. Más aún, el alumno no dispone de tiempo suficiente para razonar todo aquello que copia, llegando a convertirse en un acto mecánico. En conclusión, los alumnos tienden a sentirse inseguros sobre la corrección de lo que copian, sobre todo cuando a la hora de repetir una explicación no se hace utilizando exactamente las mismas palabras.

Debido a los inconvenientes de este método, hemos optado por dar los apuntes por escrito a los alumnos desde el primer día, exponiendo oralmente, y con ayuda de la pizarra y/o transparencias, según el caso. De esta forma, los alumnos poseen apuntes de gran calidad, completos, y ajustados al temario. Además, este método permite que el alumno se centre en comprender las explicaciones del profesor, y que el profesor disponga de más tiempo para aclarar, debatir, trabajar con el grupo, realizar problemas, etc. Es decir, se busca que en clase no sólo se repase el contenido de los apuntes, sino que se añadan matices, ejemplos,... de interés que no aparecen en éstos. Esto hace atractivas las clases, haciendo que el alumno no pierda el interés por las mismas.

El esfuerzo desarrollado en la escritura de dichos apuntes se ha visto completado y recompensado con la edición de un libro en el que se recogen los mismos, junto con los problemas de clase [4]. Para la edición del libro se contactó con el Instituto de Ciencias de la Educación (ICE) de la UEX, que anualmente oferta una convocatoria para publicación de manuales y libros de apoyo a la docencia. Tras pasar por una revisión estricta en la que sólo fueron aceptados dos títulos, el libro se encuentra actualmente pendiente de publicación.

Creemos que este libro será de gran utilidad para el alumnado, pues ofrece las ventajas de los apuntes junto con una mayor calidad, debida a una edición impresa y no fotocopiada.

3.2. Enseñanza a través de Internet

Internet se está convirtiendo en un importante recurso educativo gracias a que permite superar las limitaciones de lugar y tiempo. Además, no debe olvidarse el efecto que la interactividad tiene en el proceso de aprendizaje. Por eso, desde un principio nos pareció importante su aplicación dentro de la asignatura Fundamentos de Informática.

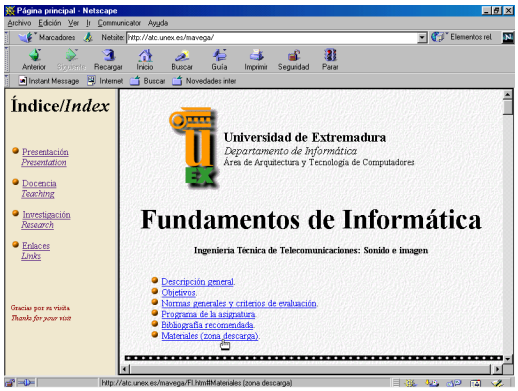


Figura 1. Sitio web para la asignatura FI

Nuestro primer acercamiento a la utilización de Internet, como soporte a la docencia, fue el diseño del sitio web de la asignatura FI [5]. Como puede observarse en la figura 1, estas páginas web han sido especialmente diseñadas para ser consultadas por los alumnos, y en ellas pueden encontrar información adicional sobre la asignatura: temario completo, bibliografía detallada, profesorado, horarios, normas generales, criterios de evaluación, tutorías, etc.; además de ficheros con material para seguir la asignatura. También se ofreció a los alumnos, desde un principio, el correo electrónico como vía adicional a las tutorías presenciales para realizar sus consultas, con las ventajas que esto implica: consultas desde casa, flexibilidad de horarios, …

Sin embargo, nuestra inquietud en este ámbito nos llevó a la solicitud de una ayuda para Proyectos de Innovación Docente al Vicerrectorado de Innovación Educativa y Calidad Docente, y al Instituto de Ciencias de la Educación, ambos de la Universidad de

Extremadura. Ayuda que fue concedida en fechas recientes, y gracias a la cual se está diseñando durante este curso académico el sistema SD2I (Sistema para la Docencia de Sistemas Digitales a través de Internet) [6]. El objetivo global de este proyecto consiste en el desarrollo de un sistema para la enseñanza, control docente y evaluación del aprendizaje a través de Internet de parte de la materia de la asignatura Fundamentos de Informática. En particular, este sistema se centra en el temario impartido durante el primer cuatrimestre y dedicado a los sistemas digitales.

Figura 2. Una de las páginas web del sistema SD2I

El sistema SD2I combina técnicas propias en lenguaje HTML [2], Java [3] y CGI [7], junto con software comercial (Macromedia Authorware [1]); y reside y se administra en un PC con Windows NT Server 4.0, configurado como servidor de Internet. Los contenidos docentes se almacenan con una estructura y formato estándar, de manera que estos contenidos puedan generarse en PCs no conectados en red y ajenos al servidor, para posteriormente ser administrados por el servidor.

Las lecciones son accesibles desde cualquier computador conectado a Internet, desde el que se puede seleccionar fácilmente la materia a estudiar, interactuando con ella. De esta forma los alumnos podrán acceder al sistema tanto desde su centro universitario como desde su propio domicilio, con lo que se potencia el trabajo en casa y el autoaprendizaje, redundando en una mejora de la calidad de la enseñanza. Los resultados de esta interactividad (tiempo de aprendizaje, respuestas a las cuestiones, evaluaciones, etc.) son gestionados por el servidor. Así conoceremos el rendimiento de los estudiantes y la calidad de las lecciones desarrolladas (control docente y evaluación de los alumnos, generación de estadísticas, etc.). La figura 2 presenta, como ejemplo, una de las páginas web que componen la parte teórica del sistema, mientras que la figura 3 muestra uno de los múltiples ejercicios de autoevaluación del mismo.

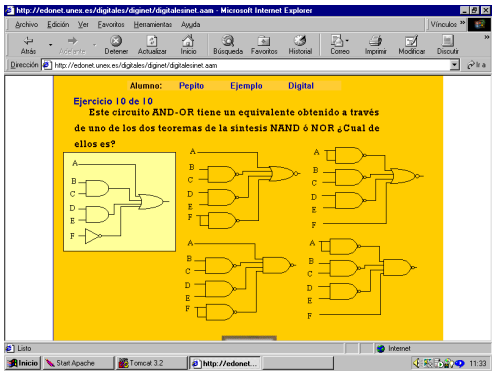


Figura 3. Ejemplo de ejercicio de autoevaluación

Como hemos dicho, el servidor monitoriza y almacena, en un formato dado, los resultados del proceso de aprendizaje. Los alumnos sólo pueden conocer algunas parcelas de la información del resultado de su interacción con el contenido didáctico (calificación de la prueba, porcentajes de éxito, etc.). Sin embargo, el resto de la información se almacena internamente, sin que el usuario tenga conocimiento de ello. Esta información resulta de gran interés, de hecho, constituirán los únicos elementos de juicio de que dispondremos para poder evaluar el rendimiento o aprovechamiento de las lecciones por parte del alumno, y esto habrá sido generado por el propio sistema de forma automática. De esta manera, además se asegura la confidencialidad de los datos obtenidos, ya que al residir éstos en el servidor, no cabe la posibilidad de "retocarlos" desde fuera.

Se prevé que se finalizará la construcción del sistema SD2I antes de que acabe el curso académico actual. Una vez concluido, se pedirá a los alumnos matriculados en la asignatura Fundamentos de Informática que lo evalúen. De esta forma, se detectarán los defectos y virtudes de SD2I frente al modelo de enseñanza actual. Esta evaluación se llevará a cabo mediante la utilización del sistema de manera masiva por parte

de los alumnos, y la realización de encuestas a los mismos. Los errores y deficiencias que se detecten por parte de los alumnos, serán depurados, buscando que el sistema esté totalmente preparado para el curso académico siguiente.

Para obtener información más detallada sobre SD2I el lector puede consultar la referencia [6].

3.3. Adquisición de material avanzado para prácticas

Empezaremos este apartado con una breve descripción sobre el material utilizado en prácticas desde un inicio, para finalizar indicando las últimas adquisiciones que se han realizado, con el objetivo de mejorar la docencia en prácticas, gracias a una ayuda obtenida a través de la Junta de Extremadura (nº de expediente MCD99C006).

Al encontrarse el programa de la asignatura dividido en dos grandes módulos, también las prácticas de la misma pueden agruparse en esos dos módulos. Cada módulo de prácticas se desarrolla durante un cuatrimestre, teniendo una relación directa con el temario impartido durante ese cuatrimestre.

Desde un principio se buscó que las prácticas permitieran al alumno el contacto directo con las dos vertientes de la Informática: hardware y software. Para la vertiente software (segundo cuatrimestre), se pensó en el lenguaje de programación C, en lugar del Pascal, por ser un lenguaje con mayor proyección en el mundo académico e industrial. Dando la posibilidad al alumno de extender posteriormente sus conocimientos en programación con otros lenguajes basados en él (C++, Visual C++,...) y de amplia aceptación en la actualidad.

Para estas prácticas, cada grupo (dos alumnos) utiliza un puesto de ordenador en una red local con acceso al compilador Borland C++ 3.1. Este compilador posee un entorno de desarrollo integrado, visual e interactivo, con todas las facilidades necesarias para la construcción de programas en C y C++, tanto para entorno MS-DOS como para Windows: editor de texto, compilador, enlazador, depurador, explorador de código fuente, administrador de proyectos, sistema de ayuda, etc. La versión 3.1 del compilador puede ejecutarse en entorno DOS y Windows con unos requisitos mínimos, lo que facilita que el alumno también pueda disponer de ella en casa.

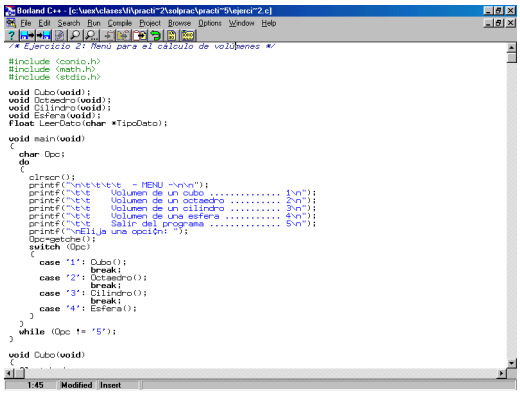


Figura 4. Aspecto del compilador Borland C++ 3.1

Se eligió este compilador por ser el más extendido en entorno DOS/Windows para la programación en C. Se eligió esta versión por ser la más alta que existe de dicho compilador (Borland C++) que permita el desarrollo de programas en C para MS-DOS. Las versiones posteriores obligan al uso de la plataforma Windows, y con ello a conceptos de programación más complejos, más propios del lenguaje C++, como: orientación a objetos, eventos, paso de mensajes, etc. Por otro lado, las versiones anteriores presentan, como es lógico, una menor calidad y facilidad en su manejo. En particular, la versión 3.1 posee las siguientes mejoras importantes: edición sintáctica por colores (se colorean las palabras del código de un programa según su tipo: palabras reservadas, constantes, definiciones de usuario,...; lo cual presenta una gran utilidad didáctica), verificación más estricta de la sintaxis de un programa emitiendo una mayor cantidad de mensajes de error y *warnings* (facilitando la programación por parte de los alumnos), capacidad para modificar el número de ficheros que pueden estar abiertos a la vez en un programa de MS-DOS, etc. La figura 4 muestra el aspecto de esta herramienta en su versión Windows, el sistema operativo más utilizado habitualmente por los alumnos en casa.

Respecto a la vertiente hardware (primer cuatrimestre), las prácticas se centran principalmente en el estudio de sistemas digitales. Desde un principio se pensó que era importante

que los alumnos trabajaran físicamente con los circuitos digitales a construir, tomando una mayor conciencia de los mismos y del entorno hardware en que se apoya la Informática. Por este motivo, a cada grupo de prácticas (dos alumnos) se le proporciona un entrenador lógico LT-536-10 (conjunto autocontenido que permite la realización de prácticas sobre circuitos lógicos sin necesidad de utilizar instrumentación exterior, ver figura 5), con los cables dotados de conectores tipo banana de 2 mm. que sean necesarios para realizar las conexiones de cada montaje; Además de una bolsita donde se incluyen circuitos integrados de la familia 74 (TTL compatibles). Los alumnos utilizan unos u otros chips según los necesiten.

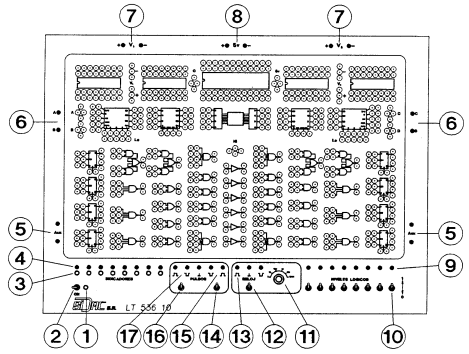


Figura 5. Entrenador lógico LT 536 10

Sin embargo, también creímos útil la utilización por parte de los alumnos de herramientas software de tipo EDA (*Electronic Design Automation*, Automatización del Diseño Electrónico), puesto que su implantación en entornos industriales y académicos es cada vez mayor. En este sentido nos decantamos por la herramienta MultiSIM (de Electronics Workbench). MultiSIM es una herramienta actual para Windows, con la que se pueden aprender y desarrollar técnicas existentes en la industria real para el diseño e implementación de dispositivos electrónicos. Este software es utilizado por ingenieros de diseño, profesores y estudiantes en todo el mundo. Se trata de un conjunto de herramientas EDA muy populares y extendidas, que permiten realizar CAD (Diseño Asistido por Computador), DFM (Diseño para Fabricación) y CAM (Fabricación Asistida por Computador). MultiSIM permite el diseño de circuitos electrónicos para implementarlos a muy distintos niveles, desde tarjetas de circuito impreso (PCBs) a dispositivos lógicos programables (FPGAs-CPLDs). Entre otras características podemos destacar las siguientes:

- Captura de esquemáticos.
- Amplia base de datos de información de componentes (agrupados por familias).
- Simulación SPICE analógica/digital completa.
- Capacidad amplia de análisis del diseño con un total de 19 instrumentos de medida.
- Capacidad de reingeniería, puesto que es posible modificar los parámetros del circuito mientras el simulador se está ejecutando, viendo instantáneamente cómo los cambios afectan al diseño.
- Estudio de circuitos electrónicos RF (Frecuencia de Radio).
- La versión "Education Lab" incluye un conjunto adicional de facilidades para educación: Posibilidad de preparación y presentación de circuitos interactivos; configuración de las opciones del software, estableciendo restricciones en los circuitos o palabras de paso a ciertas facilidades (útil para realizar pruebas y test a los alumnos); construcción de subcircuitos o "cajas negras", para simplificar un circuito (diseño modular) u ocultar un grupo de componentes de manera intencionada; posibilidad de control remoto, permitiendo un enlace bidireccional entre el profesor y sus alumnos, pudiendo el profesor controlar lo que sus alumnos ven en pantalla,...

Sin embargo, a pesar de sus muchas ventajas, su mayor inconveniente es su elevado coste, por lo que sólo fue posible su adquisición gracias a una ayuda de la Junta de Extremadura, solicitada durante el curso académico 1999/2000, dentro de su Programa de Mejora de la Calidad Docente de la Universidad de Extremadura. Esta ayuda nos permitió adquirir las licencias en red necesarias para dotar a un laboratorio (20 equipos) de la herramienta MultiSIM 6.22, versión educación. La figura 6 muestra el aspecto de esta herramienta.

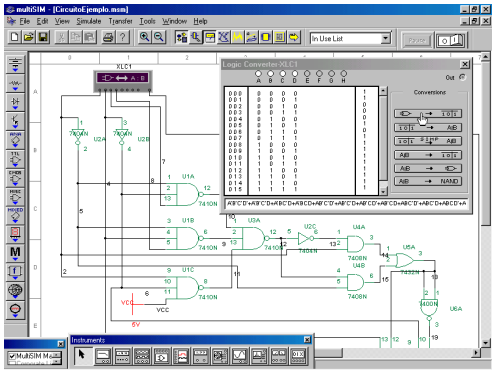
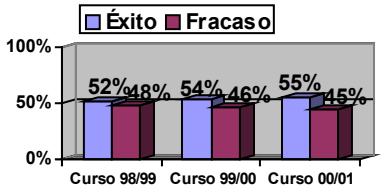


Figura 6. MultiSIM 6.22 para Windows



Curso académico	Alumnos matriculados
98/99	80
99/00	102
00/01	113
01/02	119

Figura 7. Estadísticas para la asignatura FI

Por tanto, en la actualidad nuestros alumnos compaginan el uso del entrenador lógico LT-536-10 con el software MultiSIM. En conclusión, pensamos que la utilización de herramientas hardware (entrenador lógico y chips) y software (MultiSIM) permite al alumnado conocer ambos aspectos. Por un lado, trabaja físicamente con chips, realizando las conexiones adecuadas. Por otro lado, utiliza una herramienta software de gran implantación industrial, y cuyo manejo podría requerírsele en el mercado laboral.

4. Resultados

Aunque siempre es difícil evaluar los resultados obtenidos gracias a las mejoras docentes introducidas, una posible vía objetiva es el estudio estadístico de la evolución de la asignatura. En nuestro caso, desde la implantación en la UEX de la titulación de ITTSI, es decir, desde el curso académico 1998/1999. Mediante la evaluación estadística de la asignatura podemos obtener el porcentaje de éxito y fracaso de los alumnos que cursan la asignatura a lo largo del tiempo. Si las mejoras docentes han tenido realmente éxito es de esperar que el porcentaje de fracaso entre los alumnos disminuya, aunque sí es cierto que este porcentaje también depende de otros muchos factores. La figura 7 muestra una gráfica donde se indica el porcentaje de aprobados y suspensos para la asignatura en los distintos cursos académicos.

Para evaluar estos datos con mayor fiabilidad deben tenerse en cuenta las siguientes consideraciones:

- No se dispone de los datos del curso académico 01/02, pues éste aún no ha finalizado. Aunque se espera seguir la tendencia de años anteriores.
- El número de nuevas plazas que son ofertadas anualmente asciende a 80, es decir, cada curso académico sólo pueden entrar 80 nuevos alumnos (primera matriculación), siendo el resto repetidores. Hasta la fecha siempre se han agotado las 80 plazas posibles.
- La figura 7 muestra la cantidad de alumnos matriculados por curso. Podemos ver un ascenso paulatino de dicha cantidad. Se espera que la misma se incremente en años posteriores (al tratarse de una titulación de reciente implantación) hasta llegar a una situación estable.
- También es importante recordar que se trata de una asignatura anual y obligatoria de primer curso con un total de 15 créditos, y con un amplio temario en Informática, no existiendo otra asignatura en la titulación de ITTSI con esa enorme cantidad de créditos.

5. Conclusiones

En este trabajo se ha descrito cómo se imparte la asignatura Fundamentos de Informática dentro de la titulación ITTSI de la UEX. También se han indicado las mejoras docentes que se han introducido en la asignatura desde el curso

académico 1998/1999 hasta la actualidad, resaltando el camino seguido para obtener los fondos necesarios. Estas mejoras se distribuyen en varias vertientes: edición de un libro de apoyo a la docencia, utilización de Internet para la enseñanza, control docente y evaluación del aprendizaje, y finalmente, adquisición de material más avanzado para las prácticas.

Todas las mejoras han sido bien aceptadas por los alumnos, y como muestra objetiva de ello se ha presentado un estudio estadístico de la evolución del porcentaje de éxito dentro de la asignatura.

Finalmente, indicar que estas mejoras no sólo recaen en nuestra asignatura sino que también revierten en otros ámbitos. Por ejemplo, una vez que el sistema SD2I se encuentre totalmente desarrollado podrá ser utilizado no sólo dentro de la asignatura FI, sino también dentro de otras muchas (Sistemas Digitales, Electrónica Digital, Sistemas Electrónicos Digitales, etc.), dedicadas a la temática de los sistemas digitales, y que se imparten en los primeros cursos de muchas de las Ingenierías en la mayoría de Universidades.

Referencias

[1] Kellog, O.; Ziajka, J. *Authorware 5 Attain Authorized*. Peachpit Press, 1998.

[2] Ray, D.S.; Ray, E.J. *Mastering HTML 4.0*. Sybex, 1997.

[3] Stanek, W.R.; Ketzler, M.; DeRose, S.J. *HTML, Java, CGI, VRML, SGML Web Publishing Unleashed*. Sams, 1996.

[4] Vega, M.A.; Sánchez, J.M. *Fundamentos de Informática*. Instituto de Ciencias de la Educación (ICE), Universidad de Extremadura, 2001. (ACEPTADO)

[5] Vega, M.A. http://atc.unex.es/mavega/FI.htm, 2002.

[6] Vega, M.A.; Sánchez, J.M.; Chávez, F.; Gómez, J.A. *SD2I: Sistema para la Docencia de Sistemas Digitales a través de Internet*. V Congreso de Tecnologías Aplicadas a la Enseñanza de la Electrónica, TAEE'2002, pp. 127-130, Febrero 2002.

[7] Weinman, W.E. *The CGI Book*. New Riders Publishing, 1996.

Estudio de la influencia sobre el rendimiento académico de la nota de acceso y procedencia (COU/FP) en la E.U. de Informática

Jorge Más[1], José M. Valiente[2], Luisa Zúnica[3], Rosa Alcover[4],
José V. Benlloch[5], Pedro Blesa[6]
Escuela Universitaria de Informática
Universidad Politécnica de Valencia
46022 Valencia
e-mail: jmas@fis.upv.es[1], jvalient@disca.upv.es[2], lrzunica@eio.upv.es[3] , ralcover@eio.upv.es[4] ,
jbenlloc@disca.upv.es[5], pblesa@dsic.upv.es[6]

Resumen

Se ha desarrollado una herramienta informática que permite realizar estudios de rendimiento académico. En este trabajo se muestran los resultados, a lo largo del tiempo, de los alumnos de nuevo ingreso en la EUI teniendo en cuenta su procedencia y su nota de acceso. Se ha hecho un estudio estadístico ponderado que muestra la influencia de los factores apuntados, relacionándolos con otros anteriormente observados.

1. Motivación y objetivos

Con la implantación de lo que en su día dieron en llamarse "nuevos planes de estudio", aquellos que introdujeron el concepto de crédito, se crearon unas expectativas de mejora del rendimiento académico del alumnado que, con el paso del tiempo, han quedado en una cierta frustración, y en una contrarreforma que actualmente está impulsando planes de estudio nuevos; la citada frustración proviene, en gran medida, de los numerosos estudios que se han llevado a cabo en los últimos años sobre rendimiento académico. Hay que recordar, en este sentido, que el propio Consejo de Universidades recomendó la realización de este tipo de estudios para evaluar los planes de estudio vigentes [3].

En la EUI (Escuela Universitaria de Informática) de la UPV, esta sensación de escaso rendimiento académico se ha visto confirmada por distintos estudios llevados a cabo por grupos de profesores de la misma [1][2][5][6][7][8], de modo que la Escuela tomó algunas medidas encaminadas a mitigar el elevado fracaso que en ella se producía. Con la intención de sistematizar este tipo de estudios, y en la línea de las recomendaciones del Consejo de Universidades, un grupo de profesores de la Escuela comenzamos a trabajar en un Proyecto [10] encaminado a elaborar una herramienta informática que, de forma sencilla y sin necesidad de conocimientos de informática avanzados, pudiera extraer información de la base de datos de la Universidad, que incorpora datos de alumnado, de matrículas, de planes de estudio, de calificaciones, y otros de los que la Universidad dispone; trabajos preliminares ya han sido previamente publicados [7]. Mediante esta herramienta hemos pretendido, en este trabajo, hacer una revisión de los resultados académicos de la Escuela durante el período de vigencia de los actuales planes de estudio, analizando la incidencia que sobre ellos tienen dos factores que creemos claves: la procedencia del alumno (dualidad FP/COU), factor que ha motivado actuaciones por parte de la Dirección [2] y la nota de acceso de los alumnos a la Escuela.

2. Estudio realizado

El estudio que aquí se muestra consiste en la obtención y análisis de los resultados académicos

de los alumnos de nuevo ingreso en cada uno de los cursos académicos a partir del curso 1993-94 (año en que se implantaron los nuevos planes de estudio); el último curso del que se muestran datos es del curso 1999-2000. Los resultados han sido obtenidos de forma desagregada, distinguiendo entre:

- Alumnos procedentes de COU/LOGSE, que en la EUI tienen reservado un cupo del 60% de las plazas de nuevo ingreso (200 plazas para la titulación de Ingeniero Técnico en Informática de Sistemas, y otras 200 para la titulación de Ingeniero Técnico en Informática de Gestión).
- Alumnos procedentes de FP de 2º grado, que tienen reservado un cupo del 30% de las 400 plazas de nuevo ingreso.

Además del estudio realizado para cada grupo de alumnos según su procedencia, también se realizó distinguiendo entre la nota con la que accedían a la Escuela: alumnos con nota de entrada inferior a 7, y alumnos con nota de entrada mayor o igual que 7. La segmentación de alumnos según su nota de acceso sólo pudo hacerse hasta el curso 1997-98, porque no pudimos consultar la nota de acceso de los alumnos en cursos posteriores.

El estudio se llevó a cabo de forma global para ambas titulaciones, sin distinguir entre ellas.

Para cada uno de los alumnos estudiados se contabilizó:

- Número de créditos matriculados, presentados, aprobados y suspensos en el curso académico en el que ingresó en la Escuela.
- Rendimiento académico de cada alumno en el curso en el que ingresó en la Escuela. La definición de rendimiento académico utilizada ha sido descrita en otros trabajos [4][7], y su resultado es un número entre 0 y 100 que tiene en cuenta la convocatoria en la que se aprobó la asignatura (ordinaria o extraordinaria), el año en el que se aprobó (referido al de primera matrícula en cada asignatura), y la calificación obtenida al aprobar. Aunque el rendimiento obtenido no es un valor interpretable por sí mismo, en términos relativos sí que sirve como comparación entre colectivos de alumnos distintos.

Así pues, se realizaron 7 consultas (una para cada curso académico desde el 93-94 hasta el 99-00) para el global de los alumnos procedentes de COU/LOGSE, y otras 7 para el global de los alumnos procedentes de FP. Además, se realizaron otras 5 consultas (una para cada curso académico desde el 93-94 hasta el 97-98) para los alumnos procedentes de COU/LOGSE con nota de entrada inferior a 7, y otras 5 consultas para los alumnos procedentes de COU/LOGSE con nota de entrada mayor o igual a 7.

Otras 10 consultas análogas se hicieron sobre los alumnos procedentes de FP.

En total se llevaron a cabo 34 consultas sobre la base de datos.

El resultado de cada estudio es mostrado como una tabla en la que se indica, para cada grupo estudiado, la frecuencia de alumnos cuyo número de créditos (matriculados, presentados, etc....) se encuentra en cada uno de los intervalos de amplitud 10 créditos entre 0 y 110.

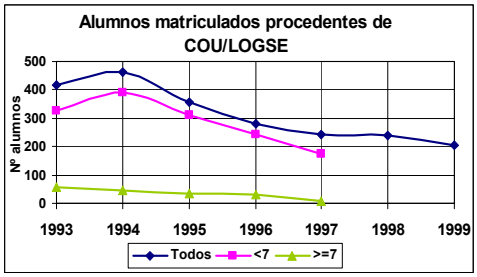
A partir de estos resultados se puede calcular, para cada grupo, el número medio de créditos matriculados, aprobados, presentados y suspensos por alumno. Además, teniendo en cuenta la calificación obtenida en cada asignatura también se puede calcular, para cada alumno, el rendimiento académico de cada curso, y el rendimiento medio del grupo de alumnos, así como la desviación típica del mismo.

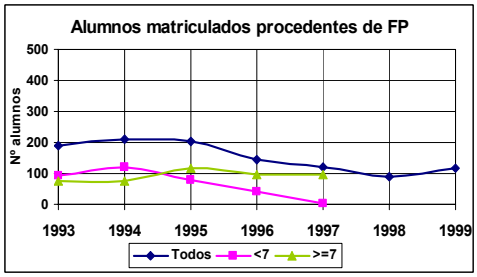
La herramienta informática que hace estos estudios realiza consultas mediante Open Acces sobre la base de datos de la Universidad, que es periódicamente actualizada.

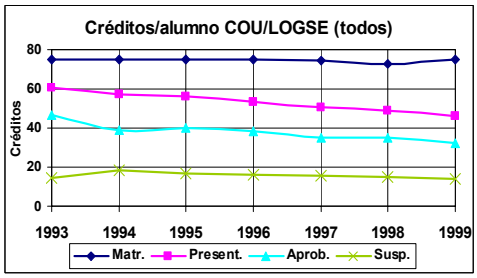
3. Resultados obtenidos

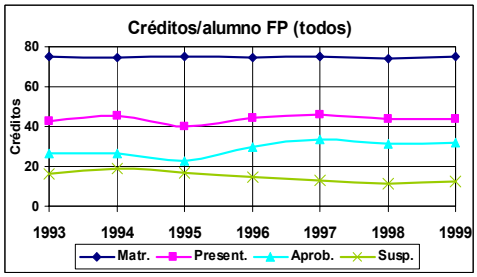
En las figuras de la página siguiente se muestran los resultados obtenidos para cada uno de los grupos de alumnos estudiados. En primer lugar, el número absoluto de alumnos que ingresó en la EUI según su procedencia y nota de acceso; después aparece el número medio por alumno de créditos matriculados, presentados, aprobados y suspensos en cada uno de los colectivos estudiados, y los resultados del rendimiento académico medio para cada grupo de alumnos.

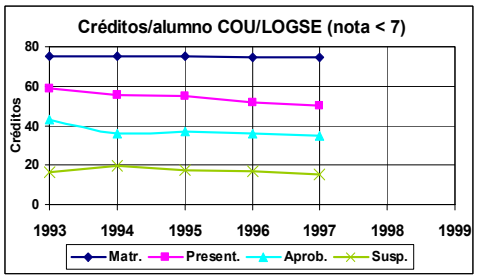
Si nos fijamos en los valores del conjunto de alumnos procedentes de COU, podemos destacar:

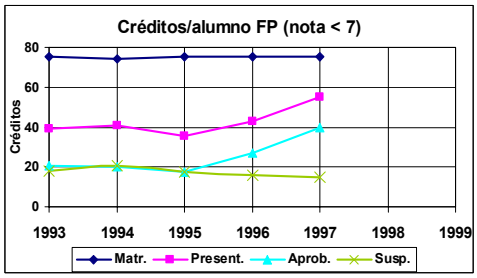

Alumnos matriculados procedentes de COU/LOGSE
Nº alumnos
500
400
300
200
100
0
1993 1994 1995 1996 1997 1998 1999
Todos <7 >=7

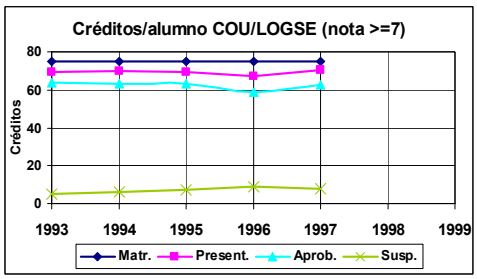

Alumnos matriculados procedentes de FP
Nº alumnos
500
400
300
200
100
0
1993 1994 1995 1996 1997 1998 1999
Todos <7 >=7

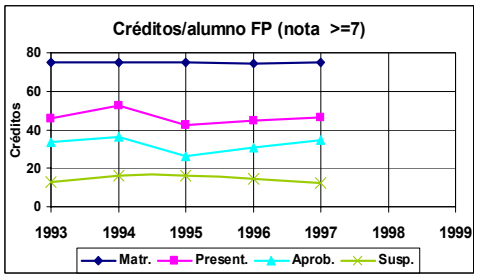

Créditos/alumno COU/LOGSE (todos)
Créditos
80
60
40
20
0
1993 1994 1995 1996 1997 1998 1999
Matr. Present. Aprob. Susp.

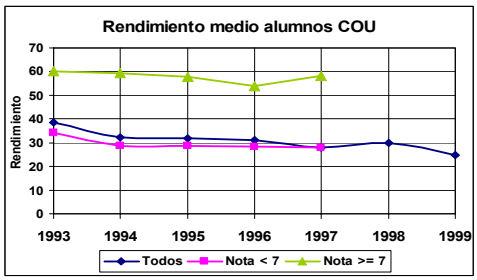

Créditos/alumno FP (todos)
Créditos
80
60
40
20
0
1993 1994 1995 1996 1997 1998 1999
Matr. Present. Aprob. Susp.

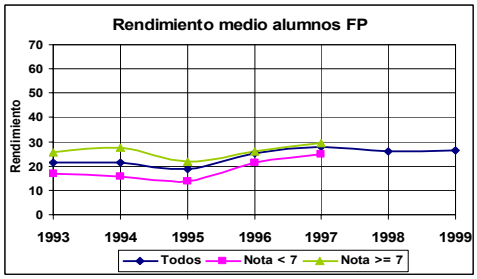

Créditos/alumno COU/LOGSE (nota < 7)
Créditos
80
60
40
20
0
1993 1994 1995 1996 1997 1998 1999
Matr. Present. Aprob. Susp.


Créditos/alumno FP (nota < 7)
Créditos
80
60
40
20
0
1993 1994 1995 1996 1997 1998 1999
Matr. Present. Aprob. Susp.


Créditos/alumno COU/LOGSE (nota >=7)
Créditos
80
60
40
20
0
1993 1994 1995 1996 1997 1998 1999
Matr. Present. Aprob. Susp.


Créditos/alumno FP (nota >=7)
Créditos
80
60
40
20
0
1993 1994 1995 1996 1997 1998 1999
Matr. Present. Aprob. Susp.


Rendimiento medio alumnos COU
Rendimiento
70
60
50
40
30
20
10
0
1993 1994 1995 1996 1997 1998 1999
Todos Nota < 7 Nota >= 7


Rendimiento medio alumnos FP
Rendimiento
70
60
50
40
30
20
10
0
1993 1994 1995 1996 1997 1998 1999
Todos Nota < 7 Nota >= 7

- El número de créditos presentados por alumno cae de forma sostenida desde 1993 hasta 1999, desde 60 hasta 45.
- El número de créditos aprobados por alumno tiene una caída similar, desde casi 50 hasta casi 30 en el mismo período.
- El número de créditos suspensos por alumno se mantiene, salvo pequeñas variaciones, constante en todo el período, en torno a 15 créditos suspensos/alumno.

En los siete años analizados, los alumnos procedentes de COU, en conjunto, muestran un nivel de absentismo en los exámenes creciente, y dado que el número absoluto de créditos suspensos por alumno se mantiene, ello se traduce en un número de créditos aprobados cada vez más bajo. En sintonía con estos resultados, el rendimiento medio de este grupo de alumnos tiene una tendencia bajista, desde valores del 40% en el 93, hasta el 25% en el 99.

Esta constante disminución del rendimiento observado en este conjunto de alumnos puede correlacionarse con otros estudios llevados a cabo sobre los conocimientos básicos (Matemáticas y Física) de los alumnos de primer curso de la EUI [8], que muestran una notable disminución de estos conocimientos básicos entre los años 1988 y 1997, lo que avalaría la hipótesis de que estos alumnos, al tener unos conocimientos básicos cada vez menores, obtienen cada vez peores resultados; y los alumnos son conscientes de esta situación, porque cada vez se presentan a un número de créditos menor.

Este análisis global podemos reproducirlo segmentando la población de alumnos de nuevo ingreso procedentes de COU según su nota de acceso. Desde el año 1993 hasta el 99, la nota media de acceso de esta población de alumnos se ha mantenido esencialmente constante, a pesar de la estudiada disminución de conocimientos básicos; ello debería inducir a interpretar la nota de acceso no como una "medida" absoluta de los conocimientos de los alumnos, sino como un parámetro de ordenación relativa de los conocimientos, habilidades, motivaciones y capacidades de los alumnos; y en este sentido, parece que el segmento de alumnos que accede a la EUI es el mismo durante todo el período estudiado.

Para aquellos alumnos con una nota de acceso inferior a 7, que es la mayoritaria, se puede hacer un análisis similar al del conjunto de esta población, aunque con unos valores absolutos algo diferentes:

- El número de créditos presentados por alumno cae desde 60 en 1993 hasta 50 en 1997, de forma sostenida.
- El número de créditos aprobados por alumno cae, en el mismo período, desde 42 hasta 34, manteniéndose más o menos constantes los créditos suspensos, lo que se traduce en rendimientos a la baja.

En cambio, aquellos alumnos con notas de entrada superiores a 7 muestran un número de créditos presentados de 70, salvo en el año 96, en el cual presenta un comportamiento distinto al de todos los demás años; de estos 70 créditos presentados, aprueban más de 60 (el número de aprobados cae ligeramente con los años), y suspenden un número de créditos entre 5 y 10. Estos valores se traducen, al combinarlos con las calificaciones obtenidas, en valores del rendimiento cercanos al 60%, aunque con una ligera tendencia a la baja.

En cambio, si nos fijamos en los valores de los créditos por alumno procedente de FP que ingresaron durante el período estudiado en la EUI:

- El número de créditos presentados por alumno se mantiene sin alteraciones importantes (salvo el año 1995) desde 1993 hasta 1999, en torno a 45 créditos, igualándose de este modo con los alumnos procedentes de COU.
- El número de créditos aprobados por alumno ha experimentado un avance de unos 10 créditos, estando estabilizado en los últimos años del período, en torno a los 35 créditos, valor similar o incluso mayor que el de los alumnos procedentes de COU.
- El número de créditos suspensos por alumno muestra una pequeña caída en el período, desde 20 créditos a unos 15 créditos suspensos/alumno, igualándose a los alumnos de COU.

En los siete años analizados, los alumnos procedentes de FP, en conjunto, muestran una clara mejoría en sus resultados, de forma más clara a partir del año 1996 y de forma más notable si los comparamos con sus homólogos procedentes de COU, que teóricamente deberían mostrar unos mejores resultados; es más, en el último año de los estudiados, 1999, los alumnos

procedentes de FP muestran unos resultados mejores que los alumnos procedentes de COU:

- Tienen un número de créditos presentados por alumno superior a los de COU, lo que unido a que el número de créditos suspensos es prácticamente el mismo, lleva a un número de créditos aprobados mayor, y a valores de rendimiento superiores en los alumnos procedentes de FP.

No obstante, estos sorprendentes resultados deberían confirmarse con estudios relativos a los resultados de los cursos 2000-2001 y 2001-2002.

En cuanto a la nota de acceso de estos alumnos, resulta interesante comprobar que, si en el 93 ingresaban más alumnos con nota inferior a 7 que superior a ella, esta relación ha ido invirtiéndose, de modo que ya en el año 97, el número de alumnos que accedía con nota inferior a 7 era prácticamente testimonial. Por ello, los resultados que arroja el estudio en este aspecto, si no se analizan cuidadosamente, deben ser tomados con precaución.

4. Análisis estadístico

Como es bien conocido, los Modelos de Regresión Lineal permiten analizar la posible relación existente entre la pauta de variabilidad de una variable aleatoria y los valores de una o más variables (aleatorias o no) de las que la primera depende, o puede depender. El análisis de los datos se lleva a cabo mediante el establecimiento de un modelo cuyos parámetros recogen y cuantifican los efectos que se pretende estudiar. Dichos parámetros se estiman a partir de los datos disponibles, utilizando eficientes procedimientos estadísticos y analizándose su significación mediante las correspondientes técnicas de inferencia.

Así, con el fi de poder cuantificar los efectos de las diferentes variables o factores considerados en el estudio sobre el rendimiento de los alumnos de Ingeniería Técnica en Informática, se ha planteado en el trabajo un modelo de regresión lineal múltiple ponderada.

Estudio sobre el rendimiento medio:

Modelización: El modelo presenta como variable dependiente, *Y*, el rendimiento de los alumnos, calculado según la expresión referida anteriormente en el trabajo. Como variables explicativas del rendimiento se han considerado las tres siguientes: año de acceso de los alumnos, procedencia (Formación Profesional o COU) y nota de entrada de los alumnos en la titulación.

El modelo de regresión incluye un termino de segundo grado con el fin de captar mejor la naturaleza del efecto del tiempo o año de acceso, sobre el rendimiento promedio de los alumnos (variables T y T^2). Para facilitar la interpretación de los parámetros del modelo, la variable T se corresponde con el año de acceso - 1993. Por tanto, si el año de acceso es 1993 (el primero considerado en el análisis) la variable T=0, si el año de acceso es 1994 entonces T=1 y así sucesivamente.

Respecto a la procedencia de los alumnos, se ha incluido una variable Adummy≅ (*FP*) que toma el valor 1 si los alumnos proceden de Formación Profesional y 0 si proceden de COU. Además, el modelo plantea la posibilidad de que el efecto de la procedencia de los alumnos sobre el rendimiento promedio pueda ser diferente en el tiempo. Por ello se debe incorporar en la ecuación del modelo la interacción *(FP)*T* .

El modelo también incluye el posible efecto de la variable nota de entrada de los alumnos en la EUI sobre el rendimiento medio de los mismos. Para ello se ha añadido una nueva variable "dummy" denominada *N5_7*, que toma el valor 1 si la nota de entrada es menor que 7 y toma el valor 0 si la nota de entrada es mayor o igual a 7. El modelo contempla la posibilidad de que el efecto de la nota de entrada sea diferente según el alumno proceda de Formación Profesional o COU, incluyendo la interacción *(N5_7)*FP*. Finalmente, la interacción *(N5_7)*T* plantea la posible diferencia del efecto de la nota de entrada sobre el rendimiento promedio de los alumnos según el año de ingreso de los mismos en la EUI.

En consecuencia, el modelo de regresión puede escribirse de la siguiente manera:

$$E(Y) = \beta_0 + \beta_1 T + \beta_2 T^2 + \beta_3 FP + \beta_4 (N5\_7) + \beta_5 (FP)*T + \beta_6 (N5\_7)*FP + \beta_7 (N5\_7)*T ,$$

en el que los parámetros β_i recogen y cuantifican los efectos que pretendemos medir.

Estimación: A partir de los datos disponibles (en total 20 valores del rendimiento medio obtenidos en las condiciones definidas por las

variables explicativas) se han estimado los valores de los parámetros β_i . Para ello se ha utilizado el paquete estadístico *STATGRAPHICS Plus v4.1*, realizando el análisis de regresión ponderada por el número de alumnos en cada caso (nótese que el rendimiento medio puede obtenerse a partir de un número muy diferente de alumnos en cada caso). La siguiente salida del programa proporciona las estimaciones de los parámetros β_i, permitiendo cuantificar los efectos de las variables consideradas en el análisis.

La columna *"Estimate"* de la salida de regresión, proporciona las estimaciones de los parámetros βi del modelo, mientras que la última columna, *"P-Value"*, indica el nivel de significación de los mismos. Como se desprende de la información contenida en esta última columna, prácticamente todos los efectos contemplados en el modelo son altamente significativos (valores inferiores a 0.05). La bondad o calidad del ajuste realizado se puede obtener a partir del Coeficiente de Determinación, "*R-squared"*.El valor obtenido, 0.976306, muy próximo a 1, corrobora la adecuación del modelo para explicar la naturaleza de la relación existente entre el rendimiento medio de los alumnos y las variables consideradas en el mismo.

Interpretación de los resultados: A partir de la salida del programa se obtiene que el rendimiento medio de los alumnos procedentes de COU con nota de acceso ≥7, que se incorporaron a la EUI en 1993 fue de 62.3708 puntos. Este valor correspondería a la estimación de β_0 .

Multiple Regression Analysis

Dependent variable: RTOMEDIO

Parameter	Estimate	Standard Error	T Statistic	P-Value
CONST	62,3708	2,14406	29,0901	0,0000
T	-4,89839	1,45357	-3,36991	0,0056
T^2	0,79771	0,26994	2,95508	0,0120
FP	-35,0089	2,41783	-14,4795	0,0000
N5_7	-28,9308	2,16304	-13,3751	0,0000
(FP*T)	2,15827	0,93871	2,29918	0,0403
(N5_7*FP)	18,5272	2,32869	7,95605	0,0000
(N5_7*T)	0,55298	0,95923	0,57648	0,5749

R-squared = 97,6306 percent
R-squared (adjusted for d.f.) = 96,2485 percent
Standard Error of Est. = 21,1005

El valor del coeficiente de T , la estimación de β1 , es negativo (- 4.89839) y significativo, lo que indicaría que el rendimiento medio de los alumnos ha ido disminuyendo desde 1993. Sin embargo el coeficiente de T^2 es positivo (0.79771) y significativo, deduciéndose que el rendimiento medio empeoró, pero sobre todo durante los primeros años del análisis.

En cuanto a la procedencia de los alumnos, puede observarse como la estimación de β_3 es muy grande en valor absoluto y por tanto un parámetro muy significativo. Ello indica que existe una diferencia muy relevante entre los rendimientos medios de los alumnos que llegan a la EUI procedentes de Formación Profesional y los que proceden de COU. Esta diferencia es de 35 puntos en el inicio del estudio; es decir, los alumnos de Formación Profesional en 1993 obtuvieron un rendimiento medio inferior en 35 puntos al de los alumnos procedentes de COU. Esta diferencia ha ido disminuyendo ligeramente con el paso del tiempo (observese como el coeficiente de la interacción (FP)*T ha resultado significativo y positivo).

El efecto de la nota de entrada con la que se incorporan los alumnos es muy relevante. El coeficiente de la variable N5_7, β_4 , ha resultado significativo y negativo (-28.9308). Esto indica que para los alumnos procedentes de COU, hay una diferencia estimada en el rendimiento medio de 28.9308 puntos entre los alumnos que acceden con nota ≥7 y los que acceden con nota <7. Con respecto a los alumnos procedentes de FP, el efecto de la nota de acceso es también relevante, pero menos marcado que en el caso anterior al resultar significativa y positiva la interacción (N5_7)*FP . Así, para los alumnos procedentes de FP, la diferencia en el rendimiento medio entre los alumnos con nota de acceso ≥7 y los alumnos con nota <7 es de 10.4036 puntos (es decir, -28.9308 + 18.5272).

El coeficiente de la interacción (N5_7)*T , esto es β_7 , no ha resultado significativo (nótese que es el único parámetro del modelo no significativo) en el análisis deduciéndose pues que el efecto de la nota de entrada sobre el rendimiento medio se ha mantenido constante a lo largo del tiempo.

Estudio sobre la desviación típica del rendimiento.

Modelización. Se ha elaborado un nuevo modelo, considerando como variable dependiente a explicar la desviación típica de los valores del rendimiento en cada grupo considerado (variable DETIPRTO). El objetivo es analizar si alguno de los factores estudiados influye sobre la presencia de una mayor o menor dispersión en los rendimientos de los alumnos de un determinado tipo. Como variables explicativas se han considerado las mismas que en el estudio sobre el rendimiento medio.

Estimación: El ajuste se ha realizado mediante la técnica de regresión "stepwise", en su variante "backward", en la que se acaban reteniendo en el modelo sólo aquellas variables que resultan significativas estadísticamente. Los resultados obtenidos en el ajuste se recogen en la tabla que presentamos a continuación. Como se observa, el ajuste obtenido explica el 73% de la variabilidad constatada en la variable DETIPRTO, lo que puede considerarse bastante aceptable, aunque no sea tan bueno como el hallado para el rendimiento medio.

Multiple Regression Analysis

Dependent variable: DTIPRTO

Parameter	Estimate	Standard Error	T Statistic	P-Value
CONST	19,248	0,298448	64,0493	0,0000
T^2	- 0,130413	0,039160	-3,33025	0,0042
FP	1,59721	0,517693	3,08524	0,0071
(N5_7*FP)	-4,4143	0,710970	-6,20878	0,0000

R-squared = 73.0273 percent
R-squared (adjusted for d.f.) = 67.9699 percent
Standard Error of Est. = 9.74088

Interpretación de los resultados: El coeficiente negativo encontrado para la variable T^2 indica que la dispersión en los valores del rendimiento ha disminuido ligeramente a lo largo del tiempo.

El coeficiente significativo positivo encontrado para la variable *FP*, indica que para los alumnos con nota de entrada alta la dispersión del rendimiento es algo mayor en los alumnos procedentes de Formación Profesional que en los de COU. Ello se debe, probablemente, a que los alumnos con notas altas de COU tienen en general buenos rendimientos todos, mientras que entre los alumnos con notas altas de Formación Profesional hay algunos cuyo rendimiento no ha sido bueno.

Sin embargo, cuando las notas de entrada son bajas, la diferencia en dispersión entre Formación Profesional y COU es -4.414 + 1.597 = -2.817, lo que indica que hay menos dispersión en los rendimientos de los de Formación Profesional que en los de COU. Ello se debe, muy probablemente, a que los alumnos procedentes de Formación Profesional y con notas bajas de acceso tienen casi todos un bajo rendimiento académico en el primer curso de la universidad, habiendo más disparidad de resultados en los alumnos con notas bajas procedentes de COU.

5. Conclusión

Sobre el aumento paulatino en el "nivel" de los alumnos de FP, que creemos es el resultado más relevante de este trabajo, podría explicarse su mejoría relativa al compararlos con los de COU, presuponiendo unos conocimientos básicos cada vez mayores. Si nos fijamos en otros estudios llevados a cabo sobre los conocimientos básicos de los alumnos al ingresar en la EUI [8], esta explicación no parece corroborar la hipótesis apuntada. En un test de conocimientos básicos pasado al inicio del curso 1996-97 en la EUI, los alumnos procedentes de COU mostraron unos resultados espectacularmente superiores a los de los alumnos procedentes de FP, tanto en Matemáticas como en Física, lo que induce a pensar que la positiva evolución de los resultados de los alumnos de FP no es debida a la mejoría en conocimientos básicos, sino a otros factores. En concreto pensamos que dos son los factores fundamentales que pueden ayudar a explicar este comportamiento:

- En primer lugar, la paulatina disminución en el número de alumnos procedentes de FP que ingresa en la EUI con nota inferior a 7, y el correspondiente incremento de aquellos con nota superior a 7. Y este hecho, que no está avalado por un incremento correspondiente de conocimientos básicos con los que acceden a la EUI, sí que refleja, en nuestra

opinión, una motivación y predisposición al aprendizaje que posibilita unos resultados mejores que los de alumnos con conocimientos básicos objetivamente superiores.

Así, podría interpretarse que los conocimientos básicos de los alumnos no es el único factor a la hora de predecir el éxito/fracaso de un alumno, y que habría otros como la motivación, los hábitos de trabajo, la capacidad de autoaprendizaje, y seguramente alguno más que tendrían una influencia determinante sobre los resultados académicos; y que, a igualdad de procedencia, la nota de acceso es un indicador en el que "confluyen" tanto los conocimientos básicos como los otros factores apuntados.

- En segundo lugar, pensamos que también la implantación de grupos especiales para alumnos procedentes de FP, que la EUI comenzó en el curso 96-97, para intentar paliar esos deficientes conocimientos básicos, ha tenido mucho que ver en el repunte de los resultados de estos alumnos. Y en esta línea, a pesar de todas las reticencias muchas veces manifestadas, y de las precauciones con que se debe acometer, creemos que es positiva la política de hacer grupos de alumnos más homogéneos que los actuales, con la intención de dar una docencia específica a cada uno de ellos, aunque los objetivos sean homogéneos para todos los alumnos, independientemente de sus características.

Agradecimientos

A 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] Añó, A., Benlloch, J.V., y otros. *Estudio comparativo del rendimiento académico de estudiantes procedentes de COU y de FP en Ingenierías Técnicas en la UPV*. Libro de resúmenes de las II Jornadas Nacionales de Innovación en las enseñanzas de las Ingenierías. UPM. 1996.

[2] 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.

[3] B.O.E. de 17 de Enero de 1997

[4] González, R. M. *Rendimiento académico en la UPM: estudio longitudinal en primer ciclo, Vol. 1 y 2*. I.C.E. de la UPM, 1993.

[5] Más, J. *Conocimientos básicos de los alumnos de nuevo ingreso en las escuelas universitarias. Evolución en el tiempo.* Enseñanza de las Ciencias. Nº extra V Congreso internacional sobre investigación en didáctica de las ciencias. Intercambios y pósters, 251-253. Murcia. 1997.

[6] Más, J. *Estudio sobre el rendimiento académico de los alumnos de la EUI de la UPV*. Libro de resúmenes de las IV Jornadas de enseñanza universitaria de Informática. Andorra. 1998.

[7] 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.

[8] 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.

[9] Más, Meseguer y otros. *La asignatura de Física frente al nivel de conocimientos básicos de los alumnos de nuevo ingreso en la universidad.* Libro de resúmenes del I Encuentro ibérico para la enseñanza de la Física. Valladolid 1991.

[10]P.I.D. 9052 de la UPV. Libro de resúmenes de los P.I.D. de la UPV. 2001.

Análisis de la integración del uso de aplicaciones de apoyo a la docencia universitaria en Internet

Piedad Garrido Picazo
Dpto. de Informática e Ing. de Sistemas
Área de Lenguajes y Sistemas Informáticos
Universidad de Zaragoza
44003 Teruel
e-mail: piedad@posta.unizar.es

Fernando Naranjo Palomino
Centro de Cálculo
Área de Ordenadores Personales
Universidad de Zaragoza
44003 Teruel
e-mail: fnaranjo@posta.unizar.es

Sergio Albiol Pérez
Fco. J. Martínez Domínguez
Dpto. de Informática e Ingeniería de Sistemas
Área de Arquitectura de Computadores
Universidad de Zaragoza
44003 Teruel
e-mail: salbiol@posta.unizar.es
e-mail: f.martinez@posta.unizar.es

Resumen

La implantación de las nuevas tecnologías de la información en el mundo universitario, aparte de tener unas evidentes ventajas, trae consigo una serie de problemas asociados que es necesario solventar.

El presente documento analiza dicha problemática, y propone una serie de alternativas para la puesta en marcha de un sistema de información en el ámbito docente, trasladando las características de calidad tradicionales al nuevo entorno.

Una de las alternativas, consiste en el uso de herramientas expertas (Dreamweaver, Funnel Web) como apoyo al desarrollo del sistema de información por parte del personal docente. La otra posibilidad consiste en la adquisición e implantación de sistemas propietarios integrales (WebCT, Blackboard).

Finalmente se estudia cada una de las posibilidades, en base a la experiencia adquirida por nuestro grupo, al estar inmersos actualmente en la puesta en marcha del anillo digital docente de la Universidad de Zaragoza.

1. Introducción

Encontrándonos en los albores del siglo XXI y con los rápidos avances que se han obtenido en las nuevas tecnologías de la información y la comunicación, no se ha hecho esperar demasiado la introducción de éstas en el mundo de la docencia, siendo esta incorporación especialmente relevante en la última década.

La progresiva integración de Internet en el mundo docente, se pone de manifiesto hoy en día en numerosos ejemplos que van desde el simple uso de Internet como nuevo medio de comunicación para la comunidad docente, por ejemplo el correo electrónico, hasta su utilidad como nuevo canal de distribución a través del cual ofertar los servicios educativos tradicionales, con un nuevo formato como es la web.

Además de los servicios tradicionales, aparece una nueva línea de aplicación que está cobrando poco a poco gran importancia. Esta línea pretende integrar a través de Internet a distintos colectivos universitarios (profesorado y alumnado).

De esta forma, además de abrir nuevas vías de comunicación entre profesor y alumno a servicios

tradicionales ya existentes (apuntes, etc.) sirve de soporte a nuevos servicios sólo accesibles gracias al uso de Internet (evaluaciones on-line, foros, chat, semi-presencialidad,...).

¿Qué debe hacer el mundo educativo para responder al reto de Internet? Lo que sí está claro es que el concepto de docencia on-line empieza a formar parte de nuestras vidas, y con él se requiere un cambio en el enfoque pedagógico de las materias, ya que disciplinas como las Telecomunicaciones, las Tecnologías de la Información, y la Informática empiezan a jugar un papel importante como herramientas de apoyo a la docencia.

La gran cantidad de oportunidades formativas que ofrece la formación on-line, poniendo los contenidos de una materia a disposición de un gran número de alumnos desde diferentes lugares, proporciona una ventaja clara sobre los formatos tradicionales, y supone un reto para la comunidad universitaria.

2. Problemática actual

Cuando el docente desea poner a disposición de los alumnos la información relativa a una asignatura haciendo uso de Internet, debe tener en cuenta que el medio de comunicación es bastante distinto a lo que ya estaba acostumbrado.

Nuevos objetivos plantean nuevos retos, y la puesta en práctica de las nuevas estructuras no resulta trivial en la mayoría de los casos. El principal problema radica en que, aunque exista la tecnología necesaria para la aplicación de las nuevas tecnologías a la docencia, las herramientas que lo pueden hacer posible tampoco han evolucionado lo suficiente como para que su aplicación por la comunidad docente sea sencilla.

Se requiere un esfuerzo en formación y práctica en algunas técnicas que en muchos casos son desconocidas para la mayoría del personal docente. En la actualidad, la construcción de una página web de una asignatura, a la que podríamos considerar el núcleo de servicios docentes ofrecidos a través de Internet, no resulta accesible para una gran parte del profesorado debido a la cantidad de conocimientos específicos del área informática que se requieren. Incluso la simple colocación de apuntes en un determinado servidor web suele conllevar el aprendizaje de alguna herramienta nueva (por ejemplo un cliente FTP). Como es de suponer, esta problemática se hace patente especialmente en los grupos de profesorado de áreas no técnicas, teniendo una más difícil solución en centros pequeños como los que existen en Teruel, debido a la escasez de personal.

Todos estos problemas anteriormente comentados estarían estrechamente ligados a la formación técnica del profesorado en cuanto a la implementación del sistema de apoyo a la docencia. Sin embargo, otro de los problemas que suelen aparecer, sobretodo una vez superado el primer escollo, es que aun teniendo unos ligeros conocimientos acerca de cómo implementar un pequeño sistema, el resultado no suele ser convincente. Si bien las carencias en cuanto a tecnología se refiere suelen ser grandes por parte de la mayoría del profesorado, las carencias relativas en cuanto a los nuevos Sistemas de Información (organización, recuperación, interfaz, etc.) suelen ser todavía mayores.

Otro de los problemas que suele darse con mayor frecuencia es que, una vez implantado el sistema, no se realiza una evaluación continuada y un mantenimiento de los servicios que ofrece para intentar mejorar o ampliarlos.

La evaluación, aunque debiera ser realizada en todos los aspectos de la enseñanza, en el campo de la teleformación o información on-line suele ser dejado de lado cuando quizá es más fácil y requiere menos tiempo, pues la propia tecnología puede recoger de forma automática diversos indicadores del grado de aceptación de nuestro trabajo dentro de la comunidad universitaria.

En cuanto al mantenimiento, cabe destacar que, unida a la preocupación por la construcción de un sistema de información, debe tenerse en cuenta su posterior mantenimiento, una fase de la cual nadie suele ocuparse en un principio, pero que a medio plazo puede proporcionar más de un quebradero de cabeza y una considerable pérdida de tiempo.

El mantenimiento, quizá suele ser un trabajo bastante duro, puesto que se debe realizar a la vez que el mantenimiento de los servicios llamados "tradicionales". Quizá ese sea el gran error, separar los servicios y por tanto tener que duplicar los esfuerzos.

En la actualidad, en muchos centros universitarios la responsabilidad del desarrollo y mantenimiento de los sistemas documentales correspondientes recae en personal docente, lo que debido a los problemas anteriormente expuestos, provoca una falta de integración general de este tipo de entorno en la universidad.

Por otra parte, si hablamos del concepto de calidad en la enseñanza, y nos empeñamos en introducir mecanismos que nos la aseguren, es lógico hacer extensibles esos requerimientos de calidad a estas nuevas aplicaciones de Internet en la docencia. Pues bien, es aquí donde resultan más obvios los problemas de formación a los que nos referíamos, dado que si ya puede resultar complicada la construcción de una página web docente, mucho más puede serlo si se quiere hablar de su construcción en términos de calidad.

3. El nuevo rol del profesor

El papel del docente ha cambiado sustancialmente con la aparición del nuevo paradigma centrado en el aprendizaje y en el que aprende, frente al paradigma planteado hasta la fecha actual (enseñanza vs. profesor).

Por lo general, el profesor suele limitarse a transmitir y a evaluar conocimientos. Es indudable que el buen uso de los medios actualmente disponibles en Internet puede suponer un cambio radical en las relaciones enseñanza-aprendizaje. El profesor debería convertirse en un facilitador de la información, analista crítico de áreas de conocimiento, guía de estudio, revisor y evaluador de la capacitación del alumno.

Por otra parte, el alumno debería empezar a ser consciente de su papel esencialmente activo en el proceso de aprendizaje, como miembro de una comunidad virtual de personas con unos intereses de formación compartidos.

Para poder definir el nuevo rol del profesorado, la ISTE (International Society for Technology in Education) propone una serie de habilidades de gran interés [5], que pueden servir de base para diseñar un programa de aprendizaje para los docentes.

Conseguir todas estas destrezas y habilidades no es algo inmediato, exige una planificación detenida y progresiva del aprendizaje, y una predisposición por parte del profesorado de hacerse partícipe de los nuevos propósitos, características, metas, exigencias y estilo de esta nueva tipología de formación, y que deben orientar la actividad docente, discente y administrativa de quienes componen la institución en todo momento.

En definitiva, tendría que producirse un cambio en la mentalidad del profesorado, modificando las relaciones enseñanza-aprendizaje y convirtiéndose en el facilitador de la información en aquellos niveles de conocimiento propios.

4. Posibles soluciones

Por tanto, se hace necesaria la adopción de soluciones que permitan la progresiva implantación de los nuevos servicios basados en Internet para la comunidad universitaria. Sin embargo, la subcontratación de la construcción y sobre todo del mantenimiento de páginas web exige un alto coste y una falta de independencia que no suele ser asumible.

Es de suponer que con el tiempo, la tecnología avanzará hasta ir minimizando la distancia entre lo que se quiere hacer y lo que se puede hacer con mínimos conocimientos, pero la comunidad docente no puede esperar y los servicios deben ser ofrecidos sin demora. Todo esto, nos lleva a dos posibles planteamientos:

- Optar por la propia creación de nuestro nuevo sistema docente orientado a la web y hacer uso de las nuevas herramientas expertas tales como FunnelWeb o WatchPoint, para la gestión, mantenimiento y

optimización a lo largo del ciclo de vida de un web-site formativo para que sea robusto y de calidad.

- En contraposición, tenemos la posibilidad de usar soluciones propietarias que se encuentran en el mercado tales como WebCT, que integran todo en un único producto. Estas herramientas ya proveen de las herramientas necesarias para la creación rápida de una experiencia docente interactiva, quedando de parte del profesorado la gestión de contenidos y administración.

En base a nuestra experiencia en el Campus de Teruel y tras una estrecha colaboración entre PAS y PDI, no se deben cerrar las posibilidades que nos ofrece hoy en día el mercado tecnológico, sino que hay de tratar de encontrar soluciones personalizadas, y adaptarlas a cada uno de los usuarios. Como afirman McGreal, Gram y Marks el problema vendrá determinado por qué herramientas serán adecuadas para conseguir unos objetivos educativos específicos. [7]

Sería necesario intentar encontrar un entorno eficaz, sencillo y a la vez amigable, para que al docente le supusiera el mínimo esfuerzo y dedicación a la hora de crear sus propias webs docentes.

5. Análisis comparativo

5.1. Herramientas expertas

La primera de las soluciones propuestas conlleva la realización del esfuerzo por parte del personal en la creación las web docentes y los servicios ofrecidos a través de ellas. Hoy en día ya existen herramientas orientadas al diseño visual bastante conocidas (FrontPage, Page Mill,

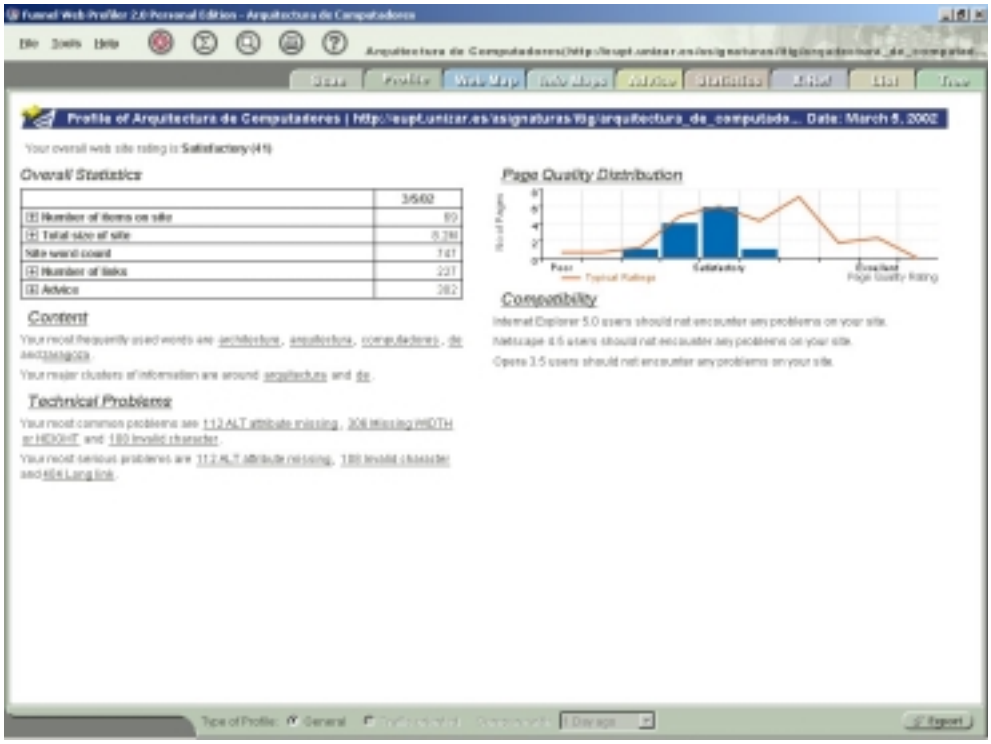


Figura 1. Perfil de asignatura generado por Funnel Web

Dreamweaver,...) que facilitan y limitan el trabajo a realizar de cara al diseño de páginas en sí.

Esto acerca a la comunidad docente la oportunidad de ofrecer varios de los servicios tradicionales en la docencia a través de Internet, como la disponibilidad de apuntes a través de la web, la comunicación vía correo electrónico, horarios de tutorías, anuncios y noticias breves, información institucional de asignaturas y el acceso a enlaces y bibliografía de interés.

Sin embargo, la posibilidad de ofrecer servicios más avanzados tales como foros, chat, evaluaciones interactivas y seguimiento personalizado a través de estas herramientas es todavía complicado para el personal docente con poca formación técnica. Además, en cualquier caso estas herramientas suelen estar dirigidas a la implementación de los sitios, pero no a su evaluación, concepto que es imprescindible abordar si se quieren introducir características de calidad en dichas webs.

De esta manera, llegamos a la conclusión de que no sólo debemos abordar la construcción de sistemas docentes basados en web, sino que también deberemos ocuparnos de evaluarlos para su optimización, mantenimiento y gestión, que redundará en un mejor servicio al alumno como usuario final de dichos sistemas.

Es aquí donde se introducen las herramientas expertas de última generación que, incorporando conocimiento experto técnico en el área que nos ocupa, son capaces de suplir la falta de formación de la persona para evaluar, señalar problemas y en última instancia recomendar las acciones correctoras apropiadas sin necesidad de que el diseñador o administrador del sitio disponga de conocimientos técnicos profundos, además de ahorrar el considerable coste empleado en la recopilación de datos del sitio y en su posterior análisis.

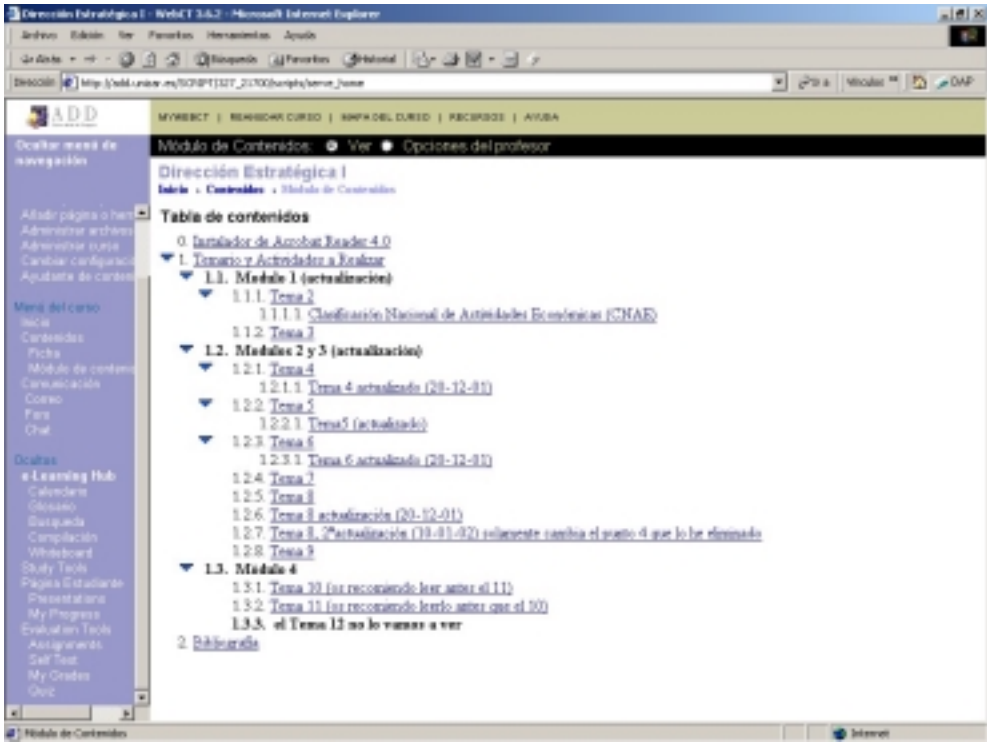


Figura 2. Módulo de contenidos de una asignatura (WebCT)

Nuestra opinión, apoyada por nuestra experiencia, es que en el escenario en el cual el personal docente implementa y mantiene estructuras web docentes, el uso de estas herramientas cada vez resulta más recomendable.

Como ejemplo planteémonos una de las experiencias abordadas por nuestro equipo, en la cual se hace uso de una de estas herramientas expertas para la evaluación de una determinada web de asignatura, dentro de su fase de mantenimiento y gestión. En este caso, usaremos la herramienta "Funnel Web Profiler" de la empresa Quest Software para la evaluación, gestión y mantenimiento de la página web de una asignatura impartida en la Escuela Universitaria Politécnica de Teruel (EUPT):

1. En un principio, la configuración de la herramienta para evaluar nuestra página o sito web es tan sencillo como indicarle la URL de la página principal. (http://eupt.unizar.es/asignaturas/itig/arquitectura_de_computadores/index.shtml)

2. Una vez realizada la evaluación, la herramienta nos proporciona una serie de pestañas categorizadas en las cuales se nos indican:

   - Características intrínsecas a nuestro sitio web. (ver figura 1)
   - Aviso de problemas de sintaxis, usabilidad y compatibilidad, con la solución recomendada.
   - Marcación de enlaces rotos y páginas inaccesibles.
   - Acceso a la estructura del sitio web para su gestión remota.

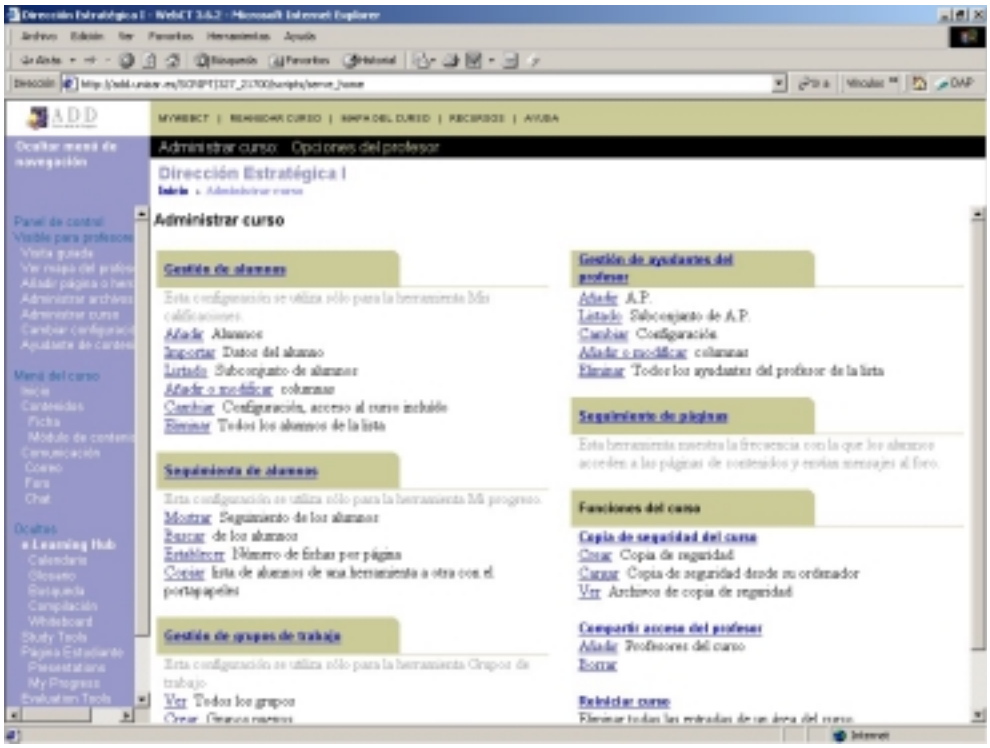


Figura 3. Gestión de una asignatura con WebCT

- Distribución de la información en el sitio web.

En resumen, mediante el uso de estas herramientas se simplifica el mantenimiento y la gestión de las páginas web dedicadas a la docencia, además de proporcionar una buena base para dotar de mayor calidad a su estructura general, todo esto con un coste reducido y minimizando el conocimiento a priori necesario para realizar tales tareas, tal y como se demanda en la comunidad docente. Además de la referida FunnelWeb Profiler, cabe destacar dentro de este tipo de herramientas InFocus, WebTrends Analisys o Web SAT.

En el lado negativo, cabe destacar que, aunque la distancia entre el mundo docente y el tecnológico se reduce con el uso de estas herramientas, siguen siendo necesarios unos ciertos conocimientos técnicos , ya que la resolución de problemas y el mantenimiento último sigue siendo manual. Por último, las servicios accesibles a ser ofertados por la comunidad docente no pueden verse ampliados por el uso de estas herramientas, dado que para los más avanzados (chat, foros, autoevaluaciones,...) se hacen imprescindibles conocimientos técnicos más profundos.

5.2. Soluciones propietarias

Como alternativa a la aproximación anterior puede adoptarse como solución la adquisición e implantación de soluciones propietarias integrales. Esto es, sistemas que integran las herramientas necesarias para ofrecer todos los servicios docentes requeridos vía web, y que intentan que, siendo el personal docente el encargado de administrar y diseñar los sistemas docentes orientados a web, estas tareas puedan ser asumidas con una formación y conocimientos informáticos mínimos.

Estas soluciones no sólo proveen de los servicios conocidos como tradicionales en el mundo docente a través de web, sino que intentan ir un poco más allá y ofrecer una serie de servicios adicionales específicamente dirigidos a la integración lo más profunda posible de la experiencia docente entre alumno y profesor a través de la web.

Como ejemplo de estos sistemas podemos destacar WebCT, usada en nuestra experiencia como plataforma soporte de toda una titulación universitaria semi-presencial en el Anillo Digital Docente de la Universidad de Zaragoza (http://add.unizar.es).

Mediante WebCT, se definen cursos y asignaturas que pueden ser diseñadas y gestionadas en su totalidad por el profesorado sin necesidad de conocimientos técnicos profundos. En este sistema, ya vienen incorporadas las siguientes herramientas docentes, que sólo deben ser configuradas adecuadamente por los profesores para los alumnos matriculados en cada curso o asignatura:

- Contenidos de asignatura. (ver figura 2)
- Listas de correo, foros y chat.
- Exámenes realizables a través de la red.
- Seguimiento del alumno.
- Auto-evaluaciones en línea.
- Grupos de trabajo y asignación de trabajos.
- Gestión de alumnado de asignatura.
- Calendario.

Las ventajas de estas soluciones son indudables en cuanto que están dirigidas específicamente a la docencia a través de Internet, aportando ya todas las herramientas implementadas y con unas labores de diseño y gestión bastante guiadas por parte del profesorado.

De esta forma se consigue una integración total en los servicios ofrecidos. Por el contrario, en la parte negativa debemos destacar el alto coste de adquisición de estos sistemas, más el coste de las máquinas donde se ubiquen .También podemos constatar su rigidez, su falta de acoplamiento con el resto de sistemas docentes a través de Internet que existan anteriormente. Por último, no podemos olvidar la dependencia que se crea entre el centro decente y una empresa externa a través de este tipo de sistemas.

6. Conclusiones

Cuando se trata de llevar adelante una innovación educativa, que exige cambios profundos en la estructura organizativa de los centros, la gestión del aula, la función de los profesores y la manera predominante de aprender de los alumnos, se deben atajar una serie de problemas que surgen en la actualidad.

La Educación todavía no encuentra su camino hacia el ciberespacio. Los tecnofóbicos y los escépticos siguen estando presentes en nuestros centros educativos, existiendo un claro rechazo a una serie de tecnologías capaces de aportar unos grandes avances en la formación universitaria y mejorando de está forma la calidad en la enseñanza.

Para ello, es necesario que aparezcan soluciones complejas que resuelvan el problema de manera sencilla, clara y concisa. Donde los usuarios no necesiten un elevado tiempo de formación, pero que los resultados sean profesionales.

Desgraciadamente, esto está aún por llegar, por lo que proponemos el uso de sencillos editores de páginas apoyados con herramientas expertas de evaluación y mantenimiento, en el caso de resolución de necesidades puntuales y cuando no se pretenda ofrecer gran cantidad de servicios, mientras que cuando queramos realizar sistemas de información integrados y completos que incluyan servicios avanzados, se debería considerar la inversión que supone la adquisición de herramientas propietarias para la docencia ya desarrolladas y comercialmente aceptadas, como pueden ser WebCT, LearningSpace o Blackboard.

Referencias

[1] Anillo digital docente de la Universidad de Zaragoza. http://add.unizar.es/
[2] Blackboard. http://www.blackboard.com
[3] Funnel Web Profiler. http://www.quest.com
[4] InFocus. http://www.ssbtechnologies.com/
[5] ISTE (International Society for Technology in Education). http://www.iste.org/
[6] LearningSpace. http://lotus.com/home.nsf/welcome/learnspace
[7] McGreal, R, Gram, T. Y Marks, T. *A Survey of New Media Development and Delivery Software for Internet-Based Learning*. http://teleeducation.nb.ca/content/media/environment/
[8] Universidad Nacional de Educación a Distancia. http://www.uned.es/
[9] Web Course Tools. http://www.webct.com
[10] WebSAT. http://zing.ncsl.nist.gov/WebTools/WebSAT/
[11] WebTrends Analisys. http://www.webtrends.com/products/as/

Algunas consideraciones sobre el léxico utilizado en la docencia de la Informática

Alberto Prieto
Dept. de Arquitectura y Tecnología de Computadores
Universidad de Granada
E 18071 Granada
e-mail: aprieto@ugr.es

Antonio Cañas
Dept. de Arquitectura y Tecnología de Computadores
Universidad de Granada
E 18071 Granada
e-mail: acanas@atc.ugr.es

Gregorio Fernández
Dpto. de Ing. de Sist. Telemáticos
E.T.S.I. Telecomunicación
Univ. Politécnica de Madrid
E 28040 Madrid
e-mail: gfer@dir.upm.es

Resumen

Este trabajo trata de poner de manifiesto la importancia que debe tener para un profesor de Informática la utilización de un léxico adecuado, lo cual requiere una sensibilidad especial para buscar con cierto rigor la traducción de los numerosos términos científicos y técnicos que continuamente aparecen, de origen fundamentalmente anglosajón. Se muestra una relación de vocablos incluidos recientemente en el diccionario de la RAE así como un Apéndice con una propuesta de traducción de términos que no suelen traducirse de forma unificada por su dificultad.

1. Motivación

La Informática, como la mayor parte de las ramas de la Ciencia y de la Técnica, está generando constantemente conceptos, que, para un avance adecuado de la misma, es necesario definir o establecer con exactitud, y, lo que a veces es más difícil, darles el nombre que los represente correctamente. Los profesionales y usuarios de un determinado dominio se entenderán tanto mejor cuanto más precisa y universalmente aceptada sea la terminología que usen. Viene al caso recordar que en aras de lograr esos objetivos la Medicina utilizó durante muchos siglos el Latín como fundamento de su terminología.

Es razonable reconocer la necesidad en los ámbitos científicos y técnicos de unificar la terminología para lograr un correcto entendimiento entre los que se pretenden comunicar. Obviamente este rigor formal no se suele requerir en otros ámbitos o en la conversación espontánea.

En la Informática concurren las especiales circunstancias de su rápida evolución, y de que ésta se produce en distintos países, fundamentalmente anglosajones. Estos hechos obligan a tener que incorporar en nuestro idioma numerosos neologismos, la mayoría de las veces a una velocidad vertiginosa, sin tiempo para una mínima maduración.

Uno de los objetivos fundamentales de la Universidad es la difusión del conocimiento (Artículo 1 de la LOU), por lo que sin duda los profesores universitarios tenemos una especial responsabilidad en preocuparnos por utilizar una terminología lo más correcta posible, en el sentido de que los términos se adecuen al concepto que quieren representar.

En el resto de esta comunicación tratamos en primer lugar (Sección 2) de hacer unas consideraciones sobre el proceso de traducción. Posteriormente (Sección 3) comentaremos distintos vocablos, la mayoría de ellos anglicismos, relacionados con la Informática que aparecen en las últimas ediciones del diccionario de la Real Academia Española (RAE). Al problema de la formación del plural de los anglicismos, que con frecuencia presenta dificultades, le dedicaremos la Sección 4. Por último incluimos un apartado (Sección 5) que trata de difundir neologismos que se están utilizando en las últimas ediciones de libros relacionados con la Informática [1-8], en la mayoría de las que nos hemos visto involucrados directamente como autores o traductores [1-4, 7-8].

2. Consideraciones sobre el proceso de traducción de vocablos

Como "las palabras se las lleva el viento" los mayores esfuerzos en la utilización de vocablos adecuados se han realizado en los documentos escritos, fundamentalmente libros de texto o para profesionales, y a ellos nos vamos a referir.

A inicios de la década de los 80 distintas editoriales toman conciencia del interés comercial de traducir, además de libros de texto, distintos libros de informática dedicados a temas tales como lenguajes de programación (BASIC y Pascal), sistemas operativos (CP/M y MS-DOS, por ejemplo), manuales de referencia, aplicaciones en distintos campos (juegos para ordenador, por ejemplo) etc. En esta etapa resaltan los esfuerzos de la empresa McGraw-Hill, que acometió la tarea de editar una gran cantidad de obras informática de distinta naturaleza traducidas al castellano del inglés. Los profesores Antonio Vaquero, Luis Joyanes y Sebastián Dormido actuaron como consultores editoriales y revisores técnicos de gran cantidad de obras. Especial mención merece el glosario de términos y siglas publicado por las dos primeras personas referenciadas anteriormente [9]. Posteriormente otras editoriales como Prencice Hall, Anaya y Thomson-Paraninfo siguieron los pasos de la anteriormente citada.

La actitud de un profesor o un traductor ante un término en un idioma extranjero, que debe expresar en castellano puede ser una de las siguientes:

1. *Traducción directa.* Un ejemplo sería el término *low-level language*, que sin dificultad se traduce por *lenguaje de alto nivel.* La traducción directa es muy peligrosa, ya que no debe acometerse sin conocer el significado del término. Esta idea conduce a que la traducción de textos deba ser realizada por profesionales del campo, y no por meros traductores profesionales. Sin duda, la no utilización de este principio provocó que el título de un capitulo de un libro de informática traducido en la década de los 70 fuese *Lenguaje de Asambleas,* y al inicio del capítulo se "aclaraba" el concepto más o menos de esta guisa: "El *lenguaje de Asambleas o Congresos* es un lenguaje de programación de bajo nivel en el que los códigos de operación están representados por símbolos nemotécnicos ..."
2. *Traducción por significado.* A veces la búsqueda en un diccionario de la acepción más adecuada para una palabra no resulta fructífera. En estos casos la primera solución es buscar un vocablo castellano que recoja el significado del concepto, sin tener en cuenta la traducción directa del mismo. Un ejemplo es la palabra *pin*, en el contexto de los circuitos integrados. Ninguna de las acepciones del diccionario como traducción de *pin* parece razonable (alfiler, perno, chaveta, clavija, polo, etc.); pero teniendo en cuenta el concepto se acostumbra a traducir por *patilla* o *terminal,* vocablos que dentro de su contexto expresan claramente la idea en cuestión. No obstante lo anterior la palabra *pin* ha sido incluida recientemente en el diccionario de la RAE con el significado anteriormente indicado[1].
3. *Castellanización.* Existen términos cuya acepción castellana es difícil de encontrar o, si se ha encontrado, no se han difundido lo suficiente o en el momento oportuno dentro de la comunidad informática. En estos casos los vocablos originales se suelen incorporar a nuestro lenguaje habitual en forma de anglicismos y, si su uso es sancionado por la opinión pública, habitualmente acaban siendo aceptados por la RAE. Ejemplos de estos vocablos son: *escáner, disquete,* etc. A este grupo de palabras le dedicamos la Sección 3 de esta comunicación.
4. *Inhibición.* Esta opción es la más cómoda, ya que consiste en no esforzarse en buscar una traducción del término o concepto en cuestión. Un ejemplo sería decir lo siguiente: "El *stack pointer* del procesador contiene la dirección de memoria ...".

Consideramos que un profesional responsable, y sobre todo dedicado a la enseñanza, debe realizar el esfuerzo de no optar por la cuarta alternativa, no dejándose llevar por lo fácil.

Con mucha frecuencia, encontrar la palabra adecuada que exprese una idea (a pesar de la gran riqueza del castellano) no es tarea fácil. Además

[1] ***Pin***. 1.m Electr. Cada una de las patillas metálicas de un conector multipolar.

existen conceptos que pueden representarse por varios vocablos, y cuando surgen y se opta por uno de estos puede ser que, con el tiempo, acabe imponiéndose otro de los alternativos. Así, en un Congreso de Informática de la década de los 70 se utilizaba *silo* como traducción de la palabra *stack;* como es bien conocido, con posterioridad se impuso la traducción por *pila.* Consideramos muy meritorio el esfuerzo de los autores de la comunicación del congreso citado por traducir el término, aunque finalmente la comunidad optase por otro. Este mismo problema en ocasiones aparece en la nación de origen, ya que a veces cuando aparece un concepto nuevo no siempre se le da una denominación única; así ocurrió, por ejemplo, con el concepto de *bus*, que inicialmente era denominado por algunos autores como *highway* [10].

3. Incorporación de términos informáticos en el Diccionario de la Real Academia Española

La Real Academia Española en las últimas ediciones de su diccionario ha realizado un esfuerzo notable para incorporar nuevos vocablos relacionados con el mundo de la Informática. Bien es verdad que en numerosas ocasiones las definiciones de los términos no están hechas con rigor, pero esto es justificable pensando que frecuentemente incluso los profesionales no nos ponemos de acuerdo en la definición de conceptos. Así, por ejemplo, del diccionario de la RAE se deduce que no pueden existir (por definición) ordenadores con pequeña o media capacidad de memoria[2]. Otra definición sorprendente de la RAE es la del término *web*[3], que en este caso peca de falta de concreción.

Consideramos que una persona culta debe conocer la terminología informática aceptada por la RAE [11], y, con este objetivo incluimos en la Tabla 1 algunos ejemplos.

Tabla 1. ***Algunos términos informáticos incluidos en el diccionario de la RAE***

Bit	
Byte	Fax
Casete	Hardware
CD	Indexar
Chip	Interfaz *(femenino)*
Colgarse (bloquearse)	Módem
Disquete	Pin
Disquetera	Píxel
Display	Software
DVD	Tóner
Escáner	Web

4. Formación del plural de nuevos términos y de siglas

La formación del plural de algunos de los nuevos términos ofrece cierta dificultad ya que en castellano el plural de las palabras terminadas en consonante se forma añadiéndoles *-es* (salvo que tengan dos o más sílabas y acaben en *s* o *x*), mientras que las voces extranjeras forman su plural añadiéndoles una *-s*. Considerando esta norma, existe la controversia sobre si ha de utilizarse *bits* o *bites* (en caso de utilizar bites, como hace el diccionario de la RAE, en el lenguaje hablado no se diferencia del plural de *byte* pronunciado en castellano), *módems* o *módemes*, *chips* o *chipes*, etc. No es común encontrar la forma del plural *bites* (buscando mediante un conocido buscador "32 bites" aparecen 144 resultados en páginas en español, mientras que para "32 bits" los resultados son 35900), *módemes* (181 frente a 37600 *módems*), y mucho menos *chipes*, que nunca se emplea. No obstante, sí es más frecuente encontrar el plural de otros términos acabado en *-es* que en *-s*, como *píxeles* o *escáneres*.

También conviene recordar que el plural de las siglas se construye haciendo variar las palabras que las acompañan [12]. Es correcto por tanto *los PC con varias CPU*, y no *los PCs con varias CPUs*.

[2] *Ordenador.* Máquina electrónica dotada de una memoria de gran capacidad y de métodos de tratamiento de la información, capaz de resolver problemas aritméticos y lógicos gracias a la utilización automática de programas registrados en ella.

[3] *Web.* Red informática

4. Propuesta de traducción de algunos términos relacionados con la informática

En el Apéndice incluimos una lista que contiene términos que suelen presentar dificultad en su traducción. Esta terminología es usada en libros de texto recientemente editados en castellano. En alguno casos se muestran varias opciones, a utilizar dependiendo del contexto o indicando que no todos los traductores han estado de acuerdo en la utilización del mismo vocablo castellano.

Obviamente, siempre debe ponerse cuidado en utilizar las palabras con rigor, y sin caer en errores provocados por la proximidad fonética de palabras en los idiomas de origen y de destino. Así, errores muy corrientes son traducir *library* por *librería* (en vez de por biblioteca) y *command* por *comando*[4] (en lugar de por orden o mandato). En la Tabla 2 incluimos una lista de este tipo de errores, que, utilizando un termino familiar entre los actores de teatro, podríamos denominar *morcillas.*

La expresión *cache memory*, debería traducirse con toda propiedad y rigor como *memoria oculta,* y así se ha realizado en varios textos (ver, por ejemplo, [4]). En efecto, el termino es de origen francés: *caché*[5]*,* y su traducción al castellano es clara. No obstante, se ha impuesto el término original, como galicismo, y en este caso lo lógico es escribirla y pronunciarla como en el original francés (con tilde en la última vocal). La traducción de este término sigue siendo muy polémica.

En otro orden de cosas, hay que resaltar el loable trabajo realizado por diversas personas que incluyen pequeños diccionarios o glosarios de términos informáticos en la web, así como foros de discusión. En la Tabla 3 incluimos una lista de direcciones web que consideramos útiles.

Tabla 2. ***Algunos términos que con frecuencia se traducen fonéticamente (incorrectamente)***

Vocablo original	*"Morcilla"*	*Traducción adecuada*
Actual	Actual	Verdadero, real
Allocate	Alocar	Reservar
Command	Comando	Orden
Constraint	Constricción	Restricción
Encript	Encriptar	Cifrar
Eventual	Eventual	Definitivo, final
Font	Fuente	Tipo (de letra)
Indent	Indentar	Sangrar
Instance	Instancia	Ejemplo, elemento
Library	Librería	Biblioteca
Location	Localización	Ubicación, posición
Paper	Paper	Artículo
Performances	Performances	Prestaciones
Privacy	Privacidad	Intimidad
Remove	Remover	Borrar, eliminar
Report	Reporte	Informe
Requirements	Requerimientos	Requisitos

5. Conclusiones

Entre los objetivos básicos de la Universidad se encuentra el de la transmisión de la ciencia, la técnica y la cultura, y aquella se efectúa fundamentalmente a través del lenguaje. En consecuencia, todo profesor universitario debería tener una especial sensibilidad y espíritu crítico hacia el lenguaje, y en particular hacia la terminología especializada que utiliza.

La mayoría de los conceptos que surgen son complejos, y con frecuencia su representación por medio de vocablos no lo es menos. No obstante, es conveniente hacer el intento de proponer términos adecuados en castellano, que puedan ser sometidos a la crítica y a la sanción pública, que es quien definitivamente seleccionará la acepción a seguir, por ser la más universal.

Sin lugar a dudas, algunas de las traducciones de los términos que aparecen en el Apéndice pueden ser polémicas, sobre todo porque muchas veces los usuarios y los profesionales se adelanta aceptando el vocablo extranjero. En estos casos resulta arduo lograr que otras opciones más lógicas y más en consonancia con nuestra lengua lleguen a prevalecer sobre las originales. No obstante, consideramos que el esfuerzo de buscar vocablos castellanos precisos merece la pena ya que sin duda la clarificación de la terminología utilizada es un paso imprescindible para el correcto avance de una ciencia y técnica en constante evolución, como es la Informática.

[4] Según el diccionario de RAE, *comando* es :1. mando militar; 2. pequeño grupo de tropas de choque, destinado a hacer incursiones ofensivas en terreno enemigo; 3. Grupo armado de terroristas

[5] Según el diccionario francés de Larousse, *caché* es: tapado, cubierto, oculto.

Tabla 3. ***Direcciones web relacionadas con la traducción de terminología informática***

Foros de discusión sobre traducción y empleo de términos relacionados con la informática

Lista de distribución *spanglish* (foro de debate y análisis de términos de informática).
http://majordomo.eunet.es/listserv/spanglish/

Términos debatidos en la lista *spanglish*.
http://www.gsi.dit.upm.es/~gfer/spanglish/

Foro TIC del Centro Virtual Cervantes (espacio de debate sobre los aspectos terminológicos de la informática y las telecomunicaciones).
http://cvc.cervantes.es/foros/foro_tic/

Diccionarios de términos relacionados con la informática, con sugerencias sobre traducción

Manuel Alfonseca: *Diccionario Unificado de Términos Informáticos*. Escuela Técnica Superior de Informática. Universidad Autónoma de Madrid.
http://www.ii.uam.es/esp/alumnos/terminologia_informatica.html

Proyecto ORCA: herramientas de ayuda para los traductores y productores de software libre en español (incluye un glosario de términos de informática, cuyo objetivo es dar una lista de sugerencias para su traducción al español).
http://quark.fe.up.pt/orca/index.es.html

Ángel Álvarez: *Basic Computer Spanglish Pitfalls (Errores habituales de Spanglish de los informáticos... y también de los no informáticos)*.
http://maja.dit.upm.es/~aalvarez/pitfalls/

Pedro José Sampedro Losada: *Anglicismos, barbarismos, neologismos y «falsos amigos» en el lenguaje informático*: http://www.ati.es/gt/lengua-informatica/externos/sampedr1.html

Carmen Ugarte: http://usuarios.tripod.es/karmentxu/glosario.html

Otros diccionarios y glosarios de términos relacionados con la informática

Rafael Fernández Calvo: *Glosario básico inglés-español para usuarios de Internet.* Cuarta edición.
http://www.ati.es/novatica/glointv2.html

Ciber-Léxico comparativo inglés - castellano, de Telefónica
http://www.telefonica.es/fat/lex.html

José Manuel Martín Nieto: *Glosario de términos informáticos Inglés-Español.*
http://www.geocities.com/Athens/2693/glosario.html

Javier Díez: Spanglish (página de enlaces a diversos glosarios).
http://www.ia.uned.es/~fjdiez/spanglish/

Diccionario Interdic: Informática e Internet.
http://pagina.de/interdic

English-Spanish Dictionary of Common Computing Terms. Universidad de Londres.
http://www.css.qmul.ac.uk/foreign/eng-spanish.htm

NETGLOS (glosario —en inglés— de términos relativos a Internet, cada uno de ellos enlazado con su traducción a varios idiomas, entre ellos el castellano).
http://wwli.com/translation/netglos/glossary/glossary.html

Dedicatoria y agradecimientos

Este trabajo está dedicado a Antonio Vaquero, con nuestro reconocimiento a la sensibilidad que siempre ha tenido hacia los problemas relacionados con la incorporación al castellano de nuevos términos en el campo de la Informática.

Referencias

[1] A. León-García, I. Widjaja. *Redes de comunicación. Conceptos fundamentales y arquitecturas básicas;* McGraw-Hill, 2001.

[2] A.Prieto, A.Lloris, J.C.Torres. *Introducción a la Informática.* 3ª Edc., McGraw-Hill. 2002.

[3] B.A.Forouzan. *Comunicación de datos y redes.* 2ª Edc. McGraw-Hill, 2002.
[4] G.Fernández. *Conceptos básicos de arquitectura y sistemas operativos;* Sistemas y Servicios de Comunicación, 1998.
[5] J.Carretero, F.García, P. de Miguel, F.Pérez. *Sistemas Operativos.*McGraw-Hill, 2001.
[6] W.Stallings. *Sistemas Operativos,* 3ª Edc., Prentice Hall, 2001.
[7] W.Stallings. *Comunicaciones y redes de computadores*. 6ª Edc. Prentice Hall, 2000.
[8] W. Stallings. *Organización y arquitectura de computadores*. 5ª Edc. Prentice Hall, 2000
[9] A.Vaquero, L.Joyanes. *Informática. Glosario de términos y siglas.* McGraw-Hill, 1985.
[10] D. Lewin. *Theory and design of digital computers.* Nelson London, 1973.
[11] Real Academia Española. *Diccionario de la Lengua Española,*22ª Edc. Editorial Espasa Calpe, 2001.
http://buscon.rae.es/drae/drae.htm
[12] Real Academia Española. *Ortografía de la lengua española*. Espasa Calpe, 1999.

Apéndice
Términos que con frecuencia se presentan difíciles de traducir

Applet	miniprograma, programilla
Backbone	red troncal
Background	segundo plano, entre bastidores
Backtracking	retroceso
Backup	copia de seguridad
Benchmark	programas de prueba
Big-endian (criterion)	criterio del extremo mayor
Bootstrap loader	cargador inicial
Browser	navegador/visualizador de web
Buffer	memoria (o zona) temporal
Bug	gazapo, error
Burst	ráfaga
Cache (memory)	memoria caché u oculta
Command	orden
Crosstalk	diafonía
Datapath	camino de datos, ruta de datos
Deadlock	interbloqueo,, abrazo mortal
Debug	depurar, corregir errores
Default	implícito, por omisión
Dirty bit	bit de modificación
Disable	inhibir, deshabilitar
Dispacher	distribuidor
Display	visualizador, pantalla, display
Drop	conexión
Dummy:	ficticio
Enable	habilitar
Encrip	cifrar
Error trapping	captura de errores
Flag	indicador, señalizador
Flip-flop	biestable
Font	tipo de letra
Foreground (in)	en primer plano, en escena
Frame	marco, trama
Garbagge	información no valida, basura
Garbagge colector	recogemigas
Gateway	pasarela
Handshake	saludo, dialogo, conformidad
Hash function	función de dispersión
Host	anfitrión, computador central
Housekeeping	operaciones de servicio
Interleaving	entrelazado
Interlock	interbloqueo
Internet	*Internet*
Jumper	puente
Latch	cerrojo
Layout	trazado, disposición, diseño
Library (programs)	programas de biblioteca
Link	enlace
Linker	montador de enlaces
Little-endian (criterion)	criterio del extremo menor
Lock	bloqueo
Login (remote)	conexión remota
Login	entrada de identificación
Mail(ing)	correo, envío por correo
Map	correspondencia, proyección aplicación , traducción
Memory mapping	asignación de memoria
Miss	fracaso
Motherboard	placa base
Offset	sesgo, desplazamiento
Overflow	desbordamiento
Overhead	suplementario
Overlay	superposición
Overload	sobrecarga
Password	contraseña
Path	trayecto, camino
Performances	prestaciones
Pipe	cauce, conducto
Pipeline	cauce segmentado, cadena
Pipelining	segmentación de cauce, encadenamiento
Pixel	elemento de imagen, píxel
Plotter	registrador gráfico
Plug-in unit	unidad enchufable
Polling	consulta
Prompt	invitación
Random scan	barrido a la medida
Raster scan	barrido sistemático
Rate	tasa
Reset	reiniciación
Round robin (priority)	prioridad circular, turno rotatorio
Router	encaminador, enrutador
Routing	encaminamiento
Scheduler	planificador
Scratch-pad memory	memoria de anotaciones
Script	guión
Set	establecer, inicializar, activar
Slot	ranura
Socket	conector, *socket*
Spliting	división
Stand-alone	autónomo
Stream	flujo
Swapper	intercambiador
Thrasing	vapuleo
Thread	hebra, hilo
Throughput	rendimiento, productividad
Time stamp	fecha, marca de fecha
Timeout	fuera de plazo, expirado
Token	testigo, valor simbólico
Trace	rastreo, seguimiento
Trap	intercepción, interrupción interna
Underflow	agotamiento
Update	actualizar
User friendly	cómodo, amigable, convivencial
Utility	herramienta
www	web

Evaluación del alumnado

Métodos alternativos de evaluación basados en el sistema de honor

Emilio Camahort, Francisco Abad
Departamento de Sistemas Informáticos y Computación
Universidad Politécnica de Valencia
46022 Valencia
e-mail: {camahort,fjabad}@dsic.upv.es

Resumen

Una de las tareas más importantes y difíciles del profesor de universidad es la evaluación de los alumnos y la producción de notas objetivas. El sistema clásico de evaluación mediante exámenes contrasta con las nuevas tendencias de seguimiento del alumno y evaluación continua. Existen métodos alternativos de evaluación y la mayoría se pueden combinar con el objetivo de producir un resultado adecuado. En este artículo presentamos varios métodos alternativos de evaluación utilizados en Estados Unidos y basados en el sistema de honor. Los métodos propuestos son fundamentalmente de aplicación a los últimos cursos de carrera, donde los alumnos tienen una mayor responsabilidad.

El sistema de honor impone a los alumnos ciertas condiciones de ética y honradez a la hora de realizar tests, trabajos, prácticas, exámenes y otros ejercicios sujetos a evaluación. El sistema de honor se combina con el escalado de notas y su ajuste a la distribución normal. El escalado de notas asigna la nota máxima al mejor alumno y escala el resto de notas de acuerdo con el escalado de la nota máxima. Este ajuste permite repartir las notas finales de forma que su histograma siga una distribución normal. De este modo se promueve la competitividad entre los alumnos y se intenta garantizar que los resultados sean independientes de las incidencias particulares del curso y el sistema de evaluación. Los métodos que proponemos se basan en este principio y en el sistema de honor. Nuestra presentación incluye ejemplos reales de aplicación de los métodos de evaluación propuestos.

1. Introducción

Uno de los problemas más importantes con que se enfrenta el profesor de universidad de hoy día es la evaluación de los alumnos. Atrás han quedado los tiempos en que la solución pasaba por los estrictos e impersonales exámenes. Métodos de evaluación más recientes sugieren el seguimiento del alumno utilizando medios más amenos que los exámenes. Por ejemplo, se pueden utilizar combinaciones de trabajos, prácticas, tests, presentaciones y otros ejercicios para fomentar el seguimiento continuo de una asignatura.

Métodos de evaluación como las prácticas y los ejercicios tienen, sin embargo, un problema. Si se asignan a alumnos por separado, es común que se junten para trabajar en grupo. Cuando los alumnos trabajan en grupo es fácil que sólo uno de ellos haga el trabajo y el resto se copie los resultados. Esta situación surge también cuando alumnos repetidores utilizan trabajos de años anteriores para desarrollar los trabajos de este año.

Este problema es de difícil solución en los primeros cursos de carrera. El elevado número de alumnos y los problemas de coordinación de grupos y profesores limitan el uso de métodos de evaluación alternativos al examen. En cursos superiores, sin embargo, los grupos suelen ser más reducidos y los alumnos más responsables. Es en estos grupos donde se pueden aplicar métodos alternativos basados en el sistema de honor.

El sistema de honor es un sistema utilizado en todos los ciclos universitarios de Estados Unidos, aunque su aplicación es más útil en los ciclos segundo y tercero. La frase sistema de honor viene del inglés *honor system*, que a su vez está basado en un código de honor o *honor code*. El código de honor es un código ético que los alumnos deben

cumplir en la realización de trabajos, prácticas y otros ejercicios objeto de evaluación. La traducción más adecuada de estos términos sería código ético, pero en este artículo preferimos usar sistema y código de honor para ser fieles al origen de ambos términos. El objetivo de este artículo es presentar varios métodos alternativos de evaluación basados en el sistema de honor.

Primero introducimos el sistema de honor y describimos los medios empleados para garantizar su cumplimiento. Luego presentamos una serie de métodos de evaluación utilizados en EE. UU. y experimentados por los autores. Finalmente, complementamos nuestra presentación con unos ejemplos prácticos de asignaturas evaluadas de acuerdo con el sistema de honor.

2. El sistema de honor

El sistema o código de honor establece normas de comportamiento ético en la realización de trabajos, ejercicios, prácticas, presentaciones, exámenes y tests. Estas normas son de obligatorio cumplimiento y su incumplimiento puede llevar desde el suspenso en el ejercicio correspondiente hasta la apertura de expediente y expulsión de la universidad.

La mayoría de las universidades estadounidenses funcionan bajo los principios del sistema de honor. Algunos ejemplos de códigos de honor se pueden encontrar en las referencias [1], [2], [3] y [4]. Estas referencias corresponden a universidades y facultades de Derecho y de Ciencias Económicas y Empresariales. En [5] se pueden ver las normas de conducta de un Departamento de Informática. Nótese que en este caso se definen las reglas de comportamiento de los alumnos, así como las responsabilidades de los profesores en cuanto a la evaluación y el tratamiento de los alumnos.

Comenzamos explicando con detalle el código de honor de la Facultad de Ciencias Económicas y Empresariales de la Universidad de Texas, en Austin [2]. En primer lugar definimos los conceptos de *estándares, mentir, robar* y *copiar.*

Estándares. Los *estándares* son normas generales de conducta que deben seguirse durante el desarrollo de la actividad académica en la universidad. Estos estándares vienen establecidos por las normas de la universidad, en general, y por las normas de evaluación de cada asignatura, en particular. Los estándares de conducta prohíben el comportamiento deshonesto y la realización de actividades delictivas. Entre estas actividades se definen las tres siguientes.

Mentir. Se considera *mentir* el intento deliberado de confundir o engañar a alguien utilizando una representación falsa o parcialmente falsa de la verdad. Mentir incluye cualquier omisión de información destinada a confundir o engañar. Algunos ejemplos de mentiras son: dar una excusa falsa al entregar un trabajo tarde o no decir al profesor que se incumplen los prerrequisitos de acceso a una asignatura.

Robar. Se considera *robar* la apropiación, obtención, retención, daño o destrucción de la propiedad de otra persona bajo cualquier circunstancia. Por propiedad se entienden tanto objetos físicos, dinero, apuntes, libros y otros artículos, como servicios y propiedad intelectual. Ejemplos de robo son la sustracción de prácticas del casillero de un profesor, la ocultación de las notas de un tablón de anuncios, el hurto de objetos dejados en clase durante el descanso y la fotocopia masiva de artículos, revistas y libros con objetivos extra-curriculares. También se considera robo el subrayado y marcado de libros de la biblioteca, pues de ese modo se daña la propiedad de la universidad.

Copiar. Se considera *copiar* la obtención o aprovechamiento de información privilegiada con el objeto de alcanzar mejor calificación que los compañeros de asignatura. Ejemplos de copia son la obtención ilegal de preguntas de exámenes y tests, y la utilización de medios no autorizados para la resolución de ejercicios y prácticas. Los medios autorizados serán aquellos especificados en las normas de la asignatura. Por ejemplo, utilizar código de un programa bajado de Internet suele estar permitido cuando se cita la fuente. También se considera copiar el plagio del trabajo de otro alumno, del mismo curso o de cursos anteriores, y el uso de información obtenida a espaldas de una conversación entre profesores.

Todos los alumnos están moralmente obligados a adherirse al sistema de honor y, por extensión, a informar de cualquier incumplimiento que detecten. El sistema de honor se basa no sólo en el cumplimiento de sus normas, sino también en la responsabilidad de informar de su incumplimiento. Para reforzar esta idea las

universidades suelen exigir a los alumnos que se adhieran al sistema por escrito. El compromiso resultante suele resumirse en una frase:

"El alumno se compromete a no mentir, copiar o robar trabajos, objetos, conceptos o ideas y a informar de aquellos casos de incumplimiento de esta norma que descubra."

El código de honor, además, explica qué hacer si un alumno se encuentra con un caso de incumplimiento. El primer paso consiste en dirigirse al alumno sospechoso para cerciorarse de que se ha cometido una falta. De ser así, el alumno sospechoso deberá comunicar la falta al profesor de la asignatura o al Vicedecano o Subdirector de Alumnado. De no hacerlo así, el alumno que detectó la falta está moralmente obligado a comunicarlo al profesor o al Vicedecano o Subdirector de Alumnado. Aunque esta obligación no se exige en el código de honor, su incumplimiento fomenta futuras infracciones del sistema. Y ésta es la idea fundamental del código de honor: permitir que otros lo incumplan sólo perjudica a aquellos que lo respetan.

3. Cumplimiento del sistema de honor

Una vez definido el sistema de honor, el problema que aparece es cómo garantizar su cumplimiento. En los primeros cursos académicos las asignaturas suelen tener tantos alumnos que es necesario dividirlos en grupos con distintos horarios y profesores asignados. Como resultado, es casi imposible la aplicación del sistema de honor, pues resulta difícil identificar copias en trabajos, prácticas, ejercicios e incluso en exámenes y tests. Esto es debido a que el trabajo de corrección está distribuido entre diferentes profesores. Parece razonable, por lo tanto, mantener los exámenes en los primeros cursos de carrera y utilizar métodos alternativos a partir del tercer o cuarto curso y, por supuesto, en el tercer ciclo.

Para fomentar el cumplimiento del código de honor se utlizan dos herramientas adicionales: el escalado de notas y su ajuste a la distribución normal. El escalado de notas asigna la nota máxima al alumno con la mejor nota, y escala el resto de notas proporcionalmente a dicha nota máxima. Esta técnica se puede complementar con el ajuste de las notas de forma que su histograma siga una distribución normal.

Estas técnicas tienen las siguientes ventajas. Por un lado, evitan que circunstancias distintas de los conocimientos de los alumnos influyan en la evaluación de la asignatura. Entre esas circunstancias se pueden incluir la presencia de un profesor nuevo con falta de experiencia, la confección de un examen más difícil de lo habitual, o la selección de una fecha inadecuada para el día de la evaluación final. Por otro lado, ambas técnicas promueven la competitividad entre los alumnos y, por lo tanto, fomentan la aplicación del sistema de honor. La asignación relativa de notas demuestra al alumno que la única forma de alcanzar una nota es rendir comparativamente más que el resto de sus compañeros. Esto evita la tentación de compartir información y resultados cuando las normas de la asignatura lo prohíben.

No obstante sus ventajas, para que el escalado y el ajuste de notas sean justos, deberán cumplirse dos condiciones. La primera es que dos alumnos que demuestren el mismo nivel de conocimientos reciban la misma nota. La segunda es que el ajuste a la distribución normal no reduzca la nota numérica de ningún alumno. Es decir que el ajuste no podrá dar un 6.5 a un alumno con un 7.2. De este modo se garantiza que en un grupo con alumnos excepcionales, la media sea más alta que la habitual para esa asignatura.

4. Métodos alternativos de evaluación

Hemos visto las características más importantes del sistema de honor. En esta sección revisamos distintos métodos de evaluación que se pueden utilizar cuando existan garantías de cumplimiento. Para cada uno de los métodos proponemos una serie de normas de aplicación basadas en el sistema de honor.

4.1. Ejercicios propuestos

Los ejercicios propuestos son problemas a realizar en casa durante una o dos semanas, y a entregar para su evaluación durante el curso académico. Su objetivo es el seguimiento continuado de la asignatura por parte de los alumnos. Los ejercicios propuestos se aplican a asignaturas teóricas o teórico/prácticas y pueden consistir en cuestiones

cortas o un problema largo de desarrollo. Ejemplos de cuestiones cortas son la resolución de problemas sencillos, la prueba de teoremas y el diseño y validación de algoritmos.

Las normas de realización de ejercicios propuestos pueden ser de dos tipos, dependiendo del nivel de interacción que permitan a los alumnos. Si no se permite que los alumnos comenten los ejercicios entre sí, entonces las normas son las mismas que las de los exámenes para llevar. Estas normas están explicadas en el Apartado 4.6. Si, por el contrario, se permite la discusión de los problemas entre los alumnos, entonces las normas son las siguientes. Se recomienda a los alumnos que comenten los problemas con sus compañeros, pero se exige que cada alumno escriba la solución por su cuenta. Si algún alumno utiliza alguna idea fundamental proporcionada por otro alumno, entonces ésta deberá ser referenciada como comunicación personal del otro alumno.

Otra cuestión que aparece en la realización de ejercicios propuestos es el uso de referencias externas. ¿Puede un alumno acercarse a la biblioteca y resolver el problema con la ayuda de un libro o un artículo? ¿Y buscar la solución en Internet? La respuesta dependerá de los criterios de evaluación del profesor. Se puede permitir el uso únicamente de apuntes, o de apuntes, libros y otras referencias externas. En este último caso, toda la información obtenida deberá ser referenciada apropiadamente. En general, se asume que el alumno ha sido lo bastante inteligente para identificar y aplicar al problema una solución ya existente. De ser así, la referencia y la forma de aplicar la solución al problema son suficientes para evaluar positivamente al alumno.

4.2. Prácticas

Las prácticas consisten en la realización de una serie de ejercicios con la ayuda de equipamiento de laboratorio o de campo, y con el objetivo de aplicar los contenidos de las clases de teoría. Las prácticas son similares a los ejercicios propuestos porque se realizan en pocas semanas y conllevan la entrega de una memoria de resultados. Su realización fomenta el seguimiento continuado de la asignatura y proporciona un método adicional de evaluación.

Las normas de realización de prácticas suelen exigir que el alumno realice todo el trabajo por su cuenta. En prácticas que requieran programación, por ejemplo, el código entregado por cada alumno debe ser substancialmente distinto del código entregado por los demás. Además, para facilitar la comparación y evaluación de resultados, se exige que el código entregado por los alumnos esté en formato electrónico. En la mayoría de los casos está prohibido el uso de código bajado de Internet y, de permitirse el uso, la fuente debe aparecer claramente referenciada en la memoria de la práctica. Por lo demás, está permitido el uso de referencias como libros, apuntes y artículos como ayuda a la implementación, siempre y cuando no se utilicen programas ya escritos.

4.3. Trabajos

Los trabajos de una asignatura pueden ser de dos tipos, trabajos cortos o trabajos finales. Los trabajos cortos requieren la presentación de una memoria breve, de una a cuatro páginas, y contienen el estudio de un tema relacionado con la asignatura, complementado con un análisis del alumno. Estos trabajos se llaman también papeles o artículos, y pueden utilizarse más de una vez para evaluar una asignatura. Los trabajos cortos son similares a los ejercicios propuestos y, por lo tanto, se suelen regir por las mismas normas. Deben escribirse por separado y toda información externa debe estar referenciada. La interacción entre alumnos puede o puede no estar permitida, pero lo más común es que no esté permitida.

Los trabajos finales son trabajos que abarcan gran parte del contenido de la asignatura. Normalmente combinan parte de estudio con la aplicación de los conocimientos adquiridos en la asignatura. El ejemplo típico son los proyectos de programación. Este tipo de proyectos pueden ser individuales o colectivos, y pueden ir acompañados de una presentación en clase. Los trabajos o proyectos individuales suelen regirse por las mismas normas que las prácticas. Si se permite la colaboración entre alumnos, entonces la utilización de las ideas de otro alumno debe ser referenciada.

En los trabajos colectivos se exige a los alumnos que especifiquen claramente las tareas realizadas por cada uno de los miembros del grupo. Además, se puede solicitar que ellos

mismos puntúen el trabajo realizado por sus compañeros dentro del grupo. Esto permite al profesor juzgar de una forma mejor informada el trabajo realizado por cada uno de los miembros del grupo. En algunos casos se pedirá además que los alumnos realicen una presentación del trabajo ante el resto de la clase. En ese caso será conveniente que cada miembro del grupo presente su trabajo y que la evaluación se complemente con las opiniones del resto de alumnos de la clase, como se indica en el apartado siguiente. En cualquier caso, siempre son de aplicación las normas respecto a información extraída de libros, apuntes, artículos e Internet. Esta información debe aparecer referenciada al final del trabajo.

4.4. Presentaciones

Las presentaciones en clase ofrecen la posibilidad de que el alumno demuestre lo que ha aprendido durante clase o en la realización de un trabajo. Este último caso es el más común y puede acompañar a un trabajo individual o colectivo. Si el trabajo es colectivo, la presentación deberá hacerse, si es posible, por todos los miembros del grupo. El reparto del contenido de la presentación corresponderá hacerlo a los integrantes de cada grupo. Las presentaciones deberán ser atendidas por todos los alumnos matriculados en la asignatura. Todos ellos deberán hacer preguntas y, si es posible, evaluar el trabajo de sus compañeros.

Las preguntas en el sistema estadounidense suelen hacerse durante la presentación y no al final de la misma. Esto permite que las dudas sean resueltas en el momento más adecuado. No obstante, preguntas que no estén relacionadas directamente con el entendimiento de la charla deberán ser relegadas al final para evitar interrupciones innecesarias. La tarea de tomar este tipo de decisiones corresponde al profesor de la asignatura.

4.5. Tests cortos

Los tests cortos (llamados *quizzes* en inglés) consisten en un conjunto de preguntas que se contestan en un corto espacio de tiempo durante las clases normales de la asignatura. Un test corto suele durar media hora y puede contener dos tipos de preguntas, preguntas de tipo test y cuestiones cortas. Su objetivo principal es fomentar el seguimiento continuo de la asignatura y, por ello, no es extraño que se realicen sin avisar.

Las reglas de realización de tests cortos son las más estrictas. No se permite ningún tipo de colaboración entre los alumnos y la utilización de apuntes, libros y otra información está prohibida. Como los tests cortos se hacen durante las horas de clase es fácil garantizar que las normas de realización se cumplan.

4.6. Exámenes

Los exámenes son el método de evaluación más común y existen muchas formas de realizarlos. En este artículo describimos sólo los ejemplos más inusuales en el sistema español. Los exámenes pueden ser con apuntes, con apuntes y libros o sin ninguna ayuda. En este último caso se puede permitir al alumno que traiga una hoja a modo de chuleta con la información más relevante para el examen. Estas hojas suelen contener fórmulas.

Los exámenes pueden realizarse en un aula o en casa. Los exámenes que se hacen en casa se llaman *exámenes para llevar* y pueden ser de dos tipos, de un día para otro o de más de un día. Los exámenes de más de un día duran entre una y dos semanas. El uso de apuntes está permitido y el uso de libros depende del criterio de evaluación del profesor.

En todos los tipos de examen la colaboración entre alumnos está prohibida. En exámenes para llevar esta prohibición es tan estricta que ni siquiera permite que los alumnos comenten con sus compañeros por dónde van en la resolución del examen. Es en estos exámenes en los que el sistema de honor resulta más difícil de aplicar. No obstante, los exámenes de un día para otro suelen requerir tal volumen de trabajo, que es imposible que los alumnos colaboren durante la realización. La mayor parte del tiempo es necesaria para el desarrollo y la redacción de las respuestas, sin tiempo para discutirlas con los demás.

El éxito de los exámenes de más de un día depende del contenido del examen y de la competitividad entre los alumnos. El contenido del examen debe de ser tal que sea fácil identificar los casos de colaboración. La competitividad entre los alumnos se fomenta con el ajuste de notas descrito en la Sección 3 de este artículo. Existen universidades en EE. UU. donde no es necesario

ni siquiera vigilar los exámenes en clase. Los alumnos tienen tal sentido de la competitividad que no se copian cuando el profesor los deja solos en el aula hasta la hora de recoger el examen al final del tiempo asignado.

5. Ejemplos de aplicación

Hemos visto el sistema de honor que se utiliza en EE. UU. para garantizar un comportamiento ético en la realización de ejercicios sujetos a evaluación. Hemos visto que dicho sistema posibilita la utilización de métodos de evaluación alternativos al examen final. Nos quedan por ver algunos ejemplos de asignaturas reales que utilizan o han utilizado combinaciones de los métodos de evaluación propuestos.

Los primeros ejemplos son los más completos y los que requieren más trabajo de corrección. Recordemos que en EE. UU. existe la figura del ayudante, llamado *teaching assistant* o *TA*. Su principal tarea es la de corregir ejercicios, prácticas, trabajos e incluso exámenes. Sin ellos serían difíciles de implementar algunos de los sistemas de evaluación que describimos a continuación. Por otro lado, hemos incluido también tres ejemplos curiosos de criterios de evaluación poco habituales. El objetivo de estos tres ejemplos es complementar a los anteriores y demostrar que no hay nada como la imaginación a la hora de evaluar a los alumnos.

5.1. Diseño y análisis de algoritmos

La asignatura de diseño y análisis de algoritmos es una asignatura semestral impartida por la profesora Vijaya Ramachandran a alumnos de tercer y cuarto cursos de la carrera de Informática. La evaluación de la asignatura se hace mediante ejercicios propuestos, dos tests cortos y un examen final. Los ejercicios propuestos constan de cinco problemas teóricos y se realizan durante las semanas que no hay tests.

Los tests cortos se realizan en clase y constan de tres cuestiones teóricas un poco más sencillas que los ejercicios propuestos. Cada test dura una hora y quince minutos y cubre un tercio del contenido de la asignatura. El examen final cubre toda la asignatura y es para llevar. Consta de cinco preguntas largas y se dan dos semanas para resolverlo. El *TA* corrige los ejercicios propuestos y la profesora los tests cortos y el examen final. Para el cálculo de la nota final se asignan los siguientes pesos: 30% a los ejercicios propuestos, 20% a cada test y 30% al examen final.

Las normas para cada caso son las siguientes. En los ejercicios se recomienda a los alumnos que comenten los problemas entre sí, pero se requiere que cada alumno escriba las soluciones por su cuenta. Se pueden utilizar fuentes de información externas, pero todo trabajo ajeno debe ser referenciado. Los tests cortos son sin apuntes ni libros y no se permite la comunicación entre los alumnos. En el examen para llevar se permite el uso de apuntes y el libro de la asignatura, pero no se permite el uso de ninguna fuente más, ni la colaboración entre los alumnos.

5.2. Informática gráfica avanzada

La asignatura de informática gráfica avanzada es una asignatura semestral impartida por los profesores Don Fussell y Nina Amenta en el tercer y cuarto cursos de Informática. La evaluación se hace mediante tres prácticas y un examen final. Cada práctica es individual y consiste en la implementación de un programa relacionado con la teoría impartida hasta el momento. Para las prácticas los alumnos disponen de una serie de ayudas, como librerías, plantillas de programas y manuales. Los alumnos sólo tienen permitido el uso de esas ayudas, los apuntes y el libro de la asignatura. La colaboración no está permitida en la realización de prácticas.

Al finalizar cada práctica se entregan una memoria de unas cinco páginas y el código fuente con las instrucciones de compilación. La entrega se hace mediante un programa que genera un registro con la fecha y hora de entrega. Ese registro se utiliza para penalizar las prácticas entregadas con retraso. El *TA* se encarga de evaluar las prácticas siguiendo los criterios especificados por el profesor en una hoja de evaluación. La evaluación incluye la prueba de los programas con un conjunto de datos de test.

El examen final es un examen sin libros ni apuntes que se realiza durante el último día de clase. El examen dura una hora y quince minutos y consta de cuatro o cinco cuestiones cortas. No se permite ninguna colaboración entre los alumnos durante el examen. El profesor corrige el examen

y calcula la nota final mediando el conjunto de las prácticas al 50% con el examen final.

5.3. Análisis y métodos numéricos

Análisis y métodos numéricos se imparte semestralmente en tercero y cuarto de Informática por el profesor Alan Cline. La evaluación se hace mediante ejercicios propuestos, un examen parcial y un examen final. Hay dos tandas de ejercicios propuestos, antes y después del parcial. Cada tanda tiene cinco ejercicios que pueden ser cuestiones teóricas o problemas prácticos. Los problemas prácticos necesitan, en ocasiones, la implementación de un pequeño programa. Los exámenes son de cuestiones teórico/prácticas y se hacen para llevar. El examen parcial dura una semana y el final dos.

La colaboración está prohibida en ejercicios y exámenes y el uso de información está limitado a los apuntes de la asignatura y las ayudas proporcionadas por el profesor en clase y en horario de consultas. En este caso, las ayudas del profesor son importantes a la hora de completar los exámenes para llevar. La evaluación de la asignatura la hace enteramente el profesor (no hay *TA*) y el valor de cada parte en la nota final es: 20% los ejercicios propuestos, 35% el examen parcial y 45% el examen final.

5.4. Semántica y verificación formal

Semántica y verificación formal se imparte cada dos semestres por el profesor Allen Emerson en cuarto y quinto de Informática. La evaluación se hace mediante ejercicios propuestos y un trabajo de la asignatura. Los ejercicios propuestos son semanales y constan de cinco problemas teóricos cada uno. Su peso en la nota final es del 50%. Cuando los ejercicios se entregan, el *TA* elige cinco alumnos para corregirlos. Cada alumno se asigna la nota máxima en su ejercicio y corrige los de los demás. De este modo la evaluación la hacen los compañeros, como ocurre en congresos y revistas de investigación. Las quejas de los alumnos son resueltas por el *TA* en primera instancia y por el profesor en segunda. Las normas de realización de ejercicios permiten la colaboración y la utilización de referencias, pero requieren que cada alumno escriba las soluciones por separado.

El trabajo de la asignatura se hace en grupos de dos y tiene una pequeña componente de investigación. Cada grupo elige un tema y varias referencias relacionadas, y produce una memoria y una presentación. La colaboración está permitida incluso entre grupos con temas afines. La memoria es de unas 10 a 15 páginas y la presentación de una hora. La presentación se divide en partes iguales entre los miembros de cada grupo. Todas las presentaciones se hacen al final del semestre y el profesor invita a pizza a los alumnos que asisten a ellas. La evaluación de la memoria, la presentación y la participación en clase la realiza el profesor. La nota resultante cuenta un 50% de la nota final.

5.5. Sistemas operativos

Sistemas operativos es una asignatura semestral destinada a alumnos de tercer y cuarto cursos de Informática e impartida durante algunos años por el profesor Avi Silberschatz. La evaluación se hacía mediante una presentación en clase. Los alumnos elegían un tema relacionado con la asignatura y preparaban una presentación de media hora. Las presentaciones se hacían al final del semestre, y el profesor se dedicaba a interrogar a los alumnos hasta el punto de ridiculizarlos si no conocían muy bien el tema. Después del mal trago, el profesor asignaba la nota máxima a todos los alumnos, sin excepción.

5.6. Cálculos y el diseño de demostraciones

Cálculos y el diseño de demostraciones es una asignatura que fue impartida durante muchos años por el profesor Edsger W. Dijkstra. La asignatura se impartía cada dos semestres y estaba dirigida a alumnos de cuarto y quinto cursos de Matemáticas e Informática. La evaluación se hacía al final del semestre y consistía en mantener una conversación de media hora con el profesor Dijkstra. El tema estaba normalmente relacionado con la asignatura y la nota se asignaba según la impresión causada al profesor.

5.7. Comentario

De las seis asignaturas descritas en este artículo, cinco fueron cursadas por uno de los autores durante la realización de sus estudios de

máster y doctorado. Su experiencia demuestra que en la mayoría de los casos se cumplió el código de honor. Esto fue debido a dos motivos: el respeto a las normas, más arraigado en las universidades estadounidenses de prestigio, y la competitividad entre los alumnos. Aún así existieron casos de incumplimiento, pero estos no fueron comunes.

6. Conclusiones

El problema de la evaluación es uno de los más complejos con que se enfrenta el profesor de universidad. Con el objetivo de ofrecer soluciones hemos presentado algunos ejemplos inusuales de métodos de evaluación y de su aplicación a asignaturas en EE. UU. Los ejemplos presentados están basados en el sistema de honor, un sistema que impone a los alumnos unas condiciones de comportamiento ético en la realización de ejercicios sujetos a evaluación. Para que estas condiciones se cumplan es conveniente que los profesores utilicen técnicas como el escalado de notas y su ajuste a la distribución normal.

Pensamos que este sistema no debe ser aplicado a primeros cursos de carrera. Los alumnos no tienen la suficiente responsabilidad y su número es tan grande que el incumplimiento es difícil de detectar. Esto no significa que el código se respete en los últimos cursos de carrera. En EE. UU. es común encontrarse con instancias de plagio y omisión de referencias. Puesto que la colaboración suele estar permitida, los problemas aparecen a la hora de referenciar el trabajo de los demás. El cumplimiento también depende de la universidad. Los alumnos de universidades de prestigio suelen ser tan competitivos que evitan que sus compañeros se copien o plagien su trabajo. Sin embargo, es común encontrar casos donde las copias se han conseguido ilegalmente como, por ejemplo, entrando en los directorios personales de los demás alumnos.

Sugerimos, por lo tanto, que como experimento se intente la aplicación de estos métodos a los últimos cursos de carrera. Somos conscientes de que muchos de ellos son difíciles de aplicar en España, donde el ideal de compañerismo entre los alumnos es más importante que el cumplimiento de unas normas de evaluación. Aun así, pensamos que el sistema de honor se puede utilizar como referencia y que se puede esperar que, con el tiempo, se utilice para simplificar y mejorar el proceso de evaluación.

Agradecimientos

Este trabajo ha sido financiado en parte por el proyecto TIC99-0510-C02-01, por la Facultad de Informática de Valencia y por ayudas para la realización de estudios en el extranjero de La Caixa y del antiguo Ministerio de Educación y Ciencia. Además queremos mostrar nuestro agradecimiento a los revisores por su ayuda y por sus sugerencias para mejorar el contenido de este artículo.

Referencias

[1] *The Honor Code,* Judicial Affairs, Stanford University, Stanford CA, EE. UU., http://www.stanford.edu/dept/vpsa/judicialaffairs/honor_code.htm

[2] MBA Program Office, *The Honor Code,* McCombs School of Business, The University of Texas at Austin, Austin TX, EE. UU., http://texasmba.bus.utexas.edu/students/academics/honor/index.asp

[3] MBA/MSIA Student Handbook, *Honor Code,* Graduate School of Industrial Administration, Carnegie-Mellon University, Pittsburgh PA, EE. UU., http://student.gsia.cmu.edu/studenthandbook/honorcode.html

[4] Law School Student Guide, *Law School Honor Code,* School of Law, University of Washington, Seattle WA, EE. UU., http://www.law.washington.edu/lawschool/genbul/honorcode.html

[5] Elaine Rich, *The Computer Sciences Department Rules to Live By,* Dept. of Computer Sciences, The University of Texas at Austin, Austin TX, EE. UU., http://www.cs.utexas.edu/users/ear/CodeOfConduct.html

SCRAE'Web: Sistema de Corrección y Revisión Automática de Exámenes a través de la WEB

Nieves Pavón, José Ramón Cano, Francisco Márquez, Alfredo Sainz
Dpto. de Ingeniería Electrónica, Sistemas Informáticos y Automática
Universidad de Huelva
21007 Palos de la Frontera (Huelva)
e-mail: {npavon, jose.cano, alfredo.marquez, alfredo.sainz}@diesia.uhu.es

Resumen

En este artículo se presenta SCRAE'Web: un Sistema de Corrección y Revisión Automática de Exámenes a través de la WEB. Este entorno de corrección de exámenes vía Internet proporciona un gran número de ventajas tanto para los alumnos y alumnas como para los profesores y profesoras. Permite la realización de exámenes tipo test frente a un ordenador con conexión a Internet de modo que, una vez terminado el examen, pueda ser entregado mediante el envío de las respuestas y datos personales a un script CGI alojado en un servidor seguro para la corrección del mismo. Los resultados son mostrados al instante con lo que la revisión del ejercicio es inmediata. El diseño del sistema, así como las conclusiones y líneas futuras hacia las que nos encaminamos, se describen con detalle a continuación.

1. Introducción

En asignaturas como Programación II [1][3] (programación lógica y funcional), del plan de estudios de Ingeniero Técnico en Informática de la Universidad de Huelva, tanto la parte teórica como la parte práctica de la materia tienen el mismo peso.
En la mayoría de las ocasiones, el problema de la calificación de las prácticas puede resolverse de varias maneras:

- Mediante la realización de un examen en el laboratorio, al final del cuatrimestre.
- Mediante la realización durante el cuatrimestre de una práctica extensa con una defensa final.

En ambos casos, la automatización del proceso de calificación es prácticamente imposible. Sin embargo, la parte teórica puede ser evaluada mediante un examen escrito compuesto por ejercicios de diferentes características o bien por un test que permita probar que los alumnos tienen tanto conocimientos teóricos como prácticos de la resolución de pequeños enunciados o problemas.
En este último caso, la automatización del proceso de corrección es perfectamente posible. Es más interesante aún, si además de la corrección del test, se proporciona la posibilidad de automatizar la revisión del examen, así como el desarrollo final de una lista de resultados en el mismo instante de finalización de la prueba, todo ello en un entorno seguro mediante el uso de Internet.
En este artículo se presenta un método para implementar esta idea que presenta algunos inconvenientes pero, indudablemente, un gran número de ventajas.

2. Objetivos

Los objetivos que se persiguen con la implantación de esta metodología se enumeran a continuación:

- Anulación del tiempo de espera de resultados del examen.
- Automatización tanto de la corrección como de la revisión del examen de forma individual en un entorno completamente seguro.

- Generación automática de listas de notas en formato HTML adaptable con cualquier formato compatible.

3. Metodología

La experiencia del uso de software remoto para realizar la corrección de exámenes tipo test se viene utilizando en los tres últimos años en diversas asignaturas del plan de estudios de Ingeniero Técnico en Informática y de Ingeniero Técnico Industrial [2]. En la primera de las titulaciones la experiencia ha tenido lugar en el tercer curso y en la segunda, la experiencia se ha llevado a cabo en la especialidad de Electrónica de primer curso.
Evidentemente, aunque en el primer caso el número de alumnos implicados es menor que en el segundo, en ambos es necesario definir un conjunto de elementos comunes:

- Cómo se va a implementar el programa de corrección remoto así como la generación de listas de resultados.
- Cómo se va a establecer una correcta política de seguridad que evite las copias y la suplantación asegurando un funcionamiento del sistema fiable.
- Cómo va a ser el formulario presentado en el cliente.

3.1. Esquema genérico del sistema global

El sistema automatizado de realización y corrección de exámenes tipo test se muestra en la Figura 1.
Podemos diferenciar, claramente, dos partes, aquella que se ejecuta en el cliente y la que se ejecuta en el servidor.
En el cliente tenemos los siguientes elementos:

- Una copia impresa del examen en cuestión.
- Un formulario plantilla con una tabla de elementos de selección para realizar la elección de la respuesta adecuada a cada pregunta. Este formulario dispone de elementos internos HTML que permiten la consumación de una petición mediante el método POST de la ejecución de un script CGI alojado en una máquina remota UNIX que redirecciona los resultados del proceso de corrección a la cuenta concreta del profesor que realiza el examen.

En el servidor disponemos de un script CGI que permite la ejecución del programa de corrección en la carpeta del profesor que realiza la prueba.

3.2. Detalles del Formulario para el llenado de las respuestas del test

El formulario diseñado para el llenado de las respuestas del test se muestra en la Figura 2.
Como se observa, se trata de una simple plantilla con tres partes diferenciadas:

- Dos zonas de texto para introducir las claves.
- Conjunto de cajas de texto para introducir los datos personales del alumno o alumna.
- Tabla con un número de entradas coincidente con el número de preguntas del test, y un número de columnas coincidente con el número de respuestas por pregunta.

Se dispone, al final del formulario, de un botón que permite la ejecución de una petición de llamada a un script CGI que recibe los datos mediante el protocolo de intercambio POST.
La pulsación del botón permitirá que en la ejecución de dicha llamada se genere una secuencia de caracteres que podrá ser leída por el script a través de la entrada estándar.
Los datos enviados al script se detallan a continuación:

- Cada uno de los elementos dinámicos del formulario posee un nombre y un valor concreto. Por ejemplo, en el caso de una caja de texto, el nombre podría ser *apellido1* y el valor "*PULIDO*", en el caso de un elemento de selección el nombre podría ser *p001* y el valor "*a*" o "*b*"o, en general, la letra de la respuesta seleccionada.
- El conjunto de pares *nombre-valor* es enviado al script mediante una cadena de texto ASCII con la forma:

```
nombre1=valor1&nombre2=valor2
```

3.3. Detalle del funcionamiento del script CGI corrector para el cálculo de la nota

El script CGI activado por la petición tras la pulsación del botón de envío de las respuestas del examen se encuentra situado en el directorio c*gi-bin* de un servidor UNIX. Este script realiza una llamada a un ejecutable con permisos de ejecución

ubicado en la carpeta del profesor que realiza el examen.
El ejecutable comienza a leer de la entrada estándar la cadena de datos proveniente de la petición WEB. Internamente, formatea dichos datos y ejecuta un algoritmo que calcula la nota en función de las respuestas dadas por el alumno.
El algoritmo de cálculo de notas tiene en cuenta las respuestas erróneas para calcular el resultado final adaptándose a los estudios estadísticos realizados al respecto, así la ecuación que permite el cálculo justo de la nota final es:

```
(aciertos-fallos/N)*(10/M)
```

donde N es el número de respuestas por pregunta y M el número de preguntas totales.

3.4 Esquemas de seguridad implementados en el sistema de corrección

Dado que se trabaja en un entorno UNIX el éxito del esquema de seguridad depende, en parte, del éxito del esquema global de seguridad diseñado para el servidor que contiene los scripts. En nuestro caso, trabajamos sobre la máquina *urium.uhu.es* que hace las veces de servidor de correo y servidor WEB en la Universidad de Huelva. A nivel global, las políticas de seguridad vienen impuestas por el Servicio Central de Informática de la Universidad de Huelva, sin embargo, estas políticas de seguridad no son suficientes para evitar problemas de copia o suplantación por lo que, a nivel del cliente y del programa corrector, se ha diseñado un esquema que minimiza al máximo estos riesgos.
Se puede observar que en el formulario del cliente los dos primeros campos de texto están diseñados para introducir dos claves. Si cualquiera de las claves no son correctas, el examen no puede ser entregado, por tanto, cuando un alumno o alumna termina su ejercicio no puede pulsar el botón de envío directamente sino que debe llamar al profesor para que éste valide los datos personales y apunte las claves correctas.
La primera clave es de seguridad y la segunda es el nombre del fichero donde se encuentra una ristra de caracteres ASCII que componen las respuestas verdaderas del examen.
Dado que el fichero de soluciones tiene un nombre que sólo conoce el profesor y además se encuentra alojado en una carpeta protegida por la clave de la cuenta de dicho profesor es, prácticamente imposible, hacerse con las soluciones del examen a priori.
El script CGI evalúa la clave de seguridad y en caso de que sea incorrecta envía una respuesta al cliente. Si la clave de seguridad es correcta pero no lo es el nombre del fichero de soluciones, el script envía el mensaje adecuado al cliente. El cliente, en este caso, puede realizar un nuevo intento.

4. Conclusiones

La principal ventaja que se obtiene con este método es la disminución del tiempo de corrección del examen.

- El alumno puede saber su nota al instante.
- El alumno hace la revisión del examen tras la entrega de su ejercicio.
- El sistema es flexible y está adaptado para que funcione con un número variable de preguntas de test.
- El tiempo que el profesor ahorra en corregir exámenes lo puede emplear en tutorías o en mejorar el sistema de corrección de prácticas.
- El sistema aplica las normas estadísticas adecuadas para este tipo de exámenes.

Los principales inconvenientes son:

- Sólo funciona con exámenes tipo test.
- Cuando el número de alumnos es grande y el laboratorio es pequeño es necesario establecer varios turnos, ya que cada alumno necesita obligatoriamente un ordenador que disponga, al menos, de una conexión a Internet.

Por otro lado, los alumnos coinciden casi al 100% que este método es mucho más adecuado que los métodos tradicionales de corrección ya que les permite saber los resultados obtenidos en el instante y así realizar una mejor planificación de su tiempo.

5. Líneas Futuras

Las líneas futuras que pretendemos seguir se exponen a continuación:

- Mejorar los esquemas de seguridad para que la entrega del examen sea más cómoda tanto para el alumno/a como para el profesor/a.

- Implantar un entorno de desarrollo de exámenes de corrección automática que integre todos los pasos de diseño desde la presentación del formulario hasta la generación de código corrector.
- Ampliar su uso al máximo número de asignaturas posibles.

Referencias

[1] Actas del Jenui 2001. Universidad de las Islas Baleares, 2001.
[2] Actas del Jenui 2000. Universidad de Alcalá, 2000.
[3] Actas del Workshop en Docencia en I.A. CAEPIA 2001. Universidad de Oviedo, 2001.

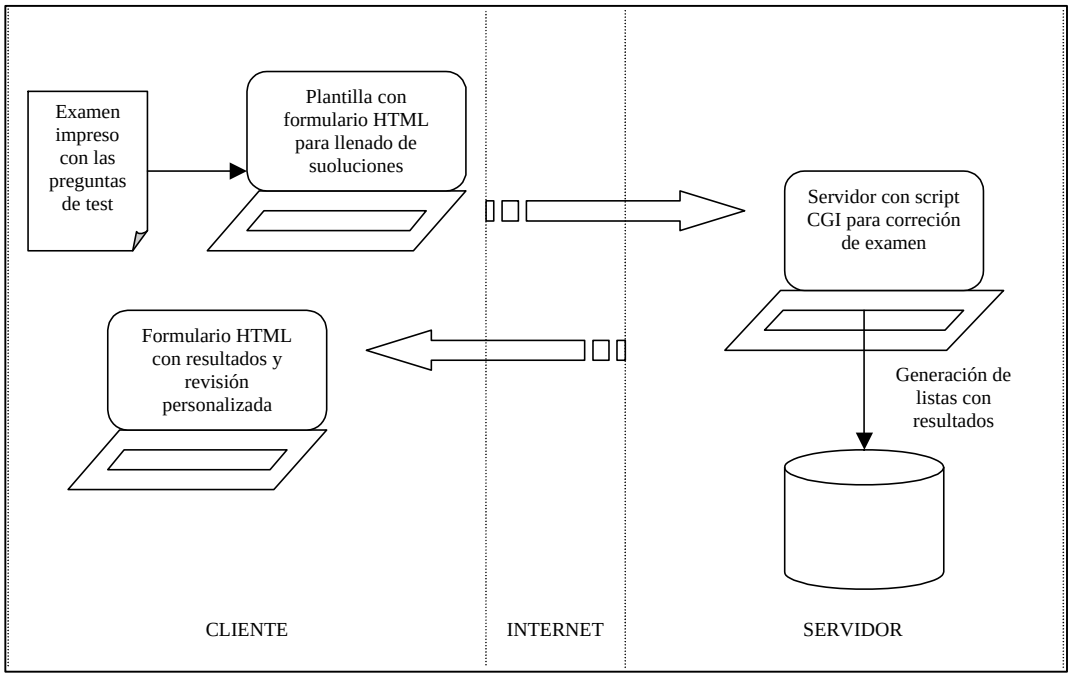


Figura 1. Esquema de funcionamiento del sistema de corrección

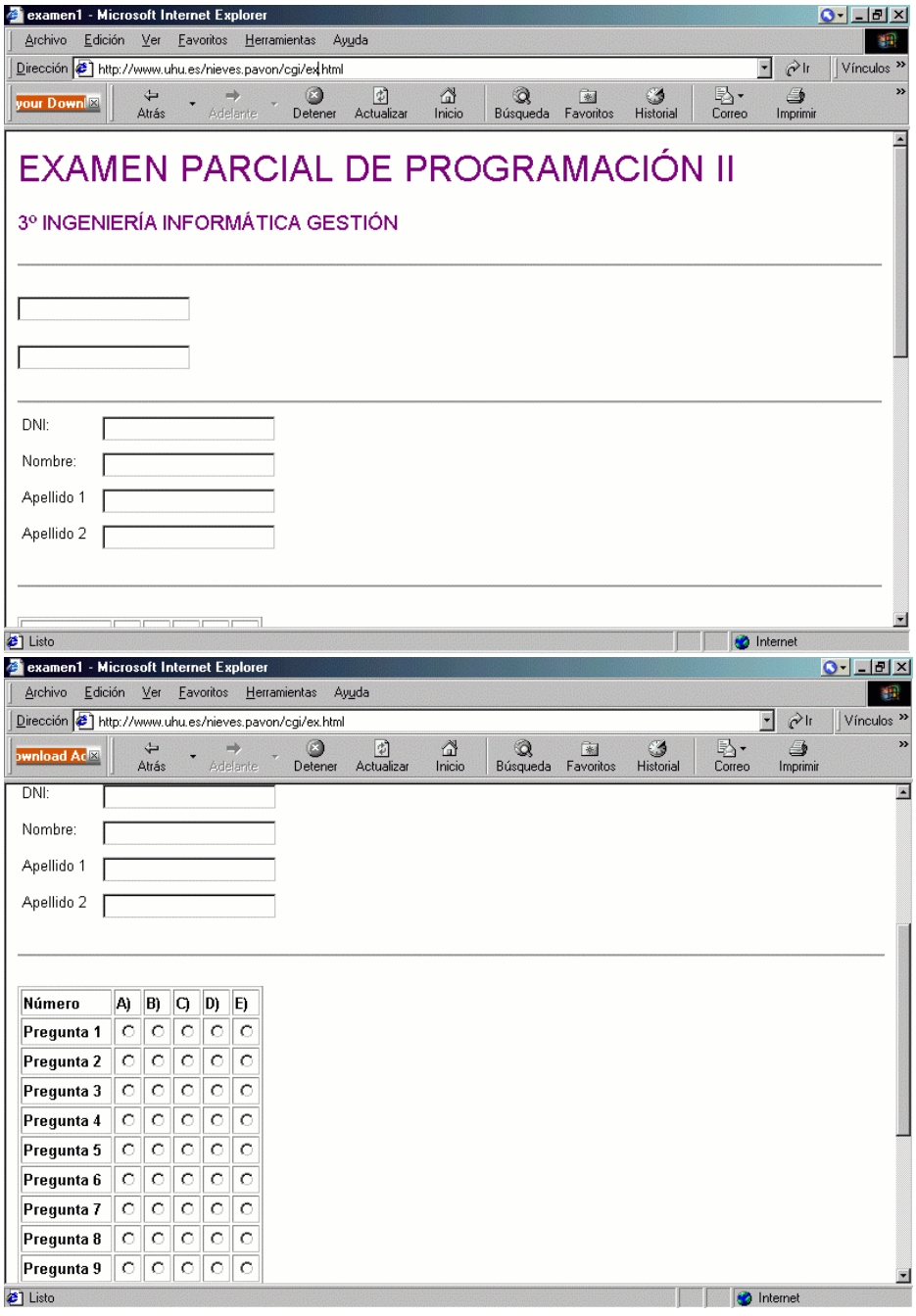


Figura 2. Detalle del formulario para marcar las respuestas del examen

Informática en otras carreras

Docencia sobre Internet en la Diplomatura de Estadística de la Universidad de Zaragoza

Ángel de Miguel Artal, Jorge Lloret Gazo
Dept. de Informática e Ingeniería de Sistemas
Universidad de Zaragoza
50009 Zaragoza
e-mail:{admiguel | jlloret}@posta.unizar.es

Resumen

En este artículo se presentan los contenidos y la experiencia docente en la impartición de la asignatura Configuraciones y Equipos de la Diplomatura de Estadística de la Universidad de Zaragoza.

1. Introducción

En el año 1969 se produjo la que podría considerarse la primera comunicación entre dos ordenadores, situado el primero de ellos en la Universidad de California en Los Ángeles y el segundo en el Instituto de Investigaciones de Stanford [1]. A partir de esta primera comunicación se realizaron multitud de pruebas e investigaciones que condujeron en 1983 al nacimiento de Internet, momento en el que el protocolo TCP/IP se estableció como lenguaje universal de comunicación entre ordenadores.

Desde entonces, el grado de evolución de Internet en los países del mundo ha sido desigual. Centrándonos en España, desde 1983 hasta los primeros años noventa, el uso de Internet estuvo limitado a centros de investigación y universidades.

Aproximadamente en el año 1993, Internet empieza a ofrecer un gran número de posibilidades y servicios y se extiende su uso a gran parte de la población. Entendemos que la Universidad, en cumplimiento de su función docente, debe adaptar sus enseñanzas a las necesidades que surgen en la sociedad conforme aparecen en ella avances tecnológicos que la modifican.

Siendo conscientes de esa necesidad, en la Diplomatura de Estadística de la Universidad de Zaragoza hemos modificado el contenido de la asignatura Configuraciones y Equipos Informáticos con el fin de ofrecer en ella conocimientos y habilidades sobre Internet. Al tiempo que realizamos esos cambios de contenido, hemos decidido aplicar métodos de enseñanza que faciliten a los alumnos la asimilación de las ideas en las que se basan las nuevas tecnologías.

Este trabajo describe los objetivos que se pretenden alcanzar con esta asignatura y qué contenidos y métodos de enseñanza se han utilizado para lograrlos. Para ello, en la sección 2 se presenta el contexto en el que se imparte la asignatura; en la sección 3 se detallan los contenidos teóricos; la sección 4 presenta las prácticas que realizan los alumnos; la sección 5 describe la metodología utilizada; la sección 6 expone la interacción con los alumnos; en la sección 7 se comentan las referencias bibliográficas utilizadas a lo largo del curso; y, finalmente, la sección 8 presenta las conclusiones.

2. Contexto

La Diplomatura de Estadística impartida por la Universidad de Zaragoza tiene el objetivo de formar estadísticos con una alta intensificación en conocimientos informáticos.

Antes de cursar la asignatura de Configuraciones y Equipos Informáticos de la Diplomatura, el alumno ya ha adquirido conocimientos básicos que serán parte fundamental para el seguimiento de la asignatura.

Así, en su formación como usuarios, la asignatura de Fundamentos de Informática, de carácter obligatorio e impartida en el primer curso de la Diplomatura, les habrá proporcionado conocimientos de manejo de

herramientas ofimáticas (procesadores de texto, hojas de cálculo), acceso y navegación por Internet e, incluso, conceptos sobre bases de datos y su utilización.

La asignatura Programación I, también de carácter obligatorio y del primer curso, les habrá proporcionado sólidos conocimientos sobre programación estructurada, formándolos como desarrolladores de aplicaciones.

Por tanto, el alumno que llega a la asignatura de Configuraciones y Equipos Informáticos está familiarizado con aplicaciones informáticas y de desarrollo y es ya capaz de interactuar con Internet, siempre y cuando se encuentre en un entorno preparado y en el que no deba preocuparse de, por ejemplo, configuraciones iniciales de los accesos o de instalación de aplicaciones.

La elección de los contenidos de Configuraciones y Equipos la hemos hecho tenido en cuenta el contexto en el que se encuentra, es decir, qué conocimientos han adquirido los alumnos a través de las asignaturas que acabamos de describir y cuál es el nivel de nuevos conocimientos que deseamos que estos adquieran.

Sobre ese punto, debemos tener en cuenta que en Informática y, en particular, en el mundo de las redes, y las comunicaciones, los contenidos son muy cambiantes. Hoy en día están obsoletos o en desuso conocimientos de hace apenas cinco años. Sin embargo, existen una serie de nociones básicas que permanecen y que ayudarán a los alumnos a comprender la evolución constante de las tecnologías. Esas nociones básicas son las que explicaremos en la parte teórica de la asignatura.

3. Contenidos teóricos

La asignatura de Configuraciones y Equipos Informáticos se estructura alrededor de cuatro temas: fundamentos generales de redes, nociones de Internet, diseño de sitios web y desarrollo de sitios web.

En el primer tema, el alumno encontrará una definición general de red y sus funciones primordiales, nociones básicas como protocolos de comunicaciones o hardware de red y una clasificación de esquemas de comunicación entre ordenadores, que abarca desde la comunicación directa entre dos ordenadores, pasando por las redes de área local hasta llegar a Internet.

Respecto a las redes de área local, nos centramos en que los alumnos conozcan el hardware específico que es necesario instalar para crear una red de área local así como el software que implementa los protocolos de comunicaciones. Describimos también los sistemas operativos de red y, finalmente, nos referimos a la forma en que se comunican entre sí las redes de área local y cómo se comunican con el exterior, especialmente con Internet.

En lo que refiere a Internet (segundo tema), hemos observado que los alumnos reconocen muchos de los términos asociados a la red pero no su significado. TCP/IP, servicios Internet, sitios web o URL son palabras habituales en nuestro vocabulario y el medio que nos rodea, pero si se pregunta por lo que significan es difícil encontrar una respuesta con un mínimo de contenido. En lugar de presentar estas nociones de manera aislada, las presentamos mediante la descripción de un proceso en el que todas ellas aparecen involucradas: el proceso de conexión desde un navegador a un sitio web.

Una vez que este vocabulario está fijado, el alumno debe percibir los elementos necesarios para establecer una conexión a Internet, ya sea desde el domicilio particular, ya sea desde una empresa u organización, así como las herramientas para acceder a la información. Sobre la conexión, describimos el módem y sus características fundamentales. A continuación, introducimos la noción de proveedor de servicios Internet y hacemos un detallado recorrido por las prestaciones que dichos proveedores ofrecen. Finalmente, explicamos las posibilidades de conexión actuales en el mercado (RTB, RDSI, ADSL, módem cable). Respecto a las herramientas, se presentan los servicios que en estos momentos ofrece Internet y las herramientas que deben emplearse para hacer uso de esos servicios.

En este punto, el alumno dispone ya de los conocimientos teóricos básicos sobre los que se sustenta la comunicación a través de Internet y durante las prácticas de este tema, ha reforzado mediante la interacción con una gran variedad de sitios web los conocimientos básicos de

usuario que ya poseía. Sin embargo, todavía desconoce cómo pueden ser construidos. En los temas tercero y cuarto del curso se ofrece a los alumnos los conocimientos necesarios para diseñar y desarrollar sitios web.

El tercer tema comienza insistiendo a los alumnos en que para la creación de un sitio web no basta conocer las herramientas, ya sea aplicaciones de desarrollo como Frontpage, ya sea el lenguaje HTML. Por el contrario, la creación de un sitio web contiene también una parte muy importante de diseño, proporción que día a día se hace más patente e importante en Internet. Se muestra a los alumnos que se trata de una actividad difícil ya que involucra personas de muy distinta formación como, por ejemplo, programadores, redactores o diseñadores gráficos, y es necesario coordinar sus esfuerzos para lograr un sitio de calidad.

Teniendo en cuenta lo anteriormente expuesto, en este tema ofrecemos una serie de recomendaciones que deben seguirse para conseguir que el sitio web cumpla la finalidad para la que haya sido diseñado. Las recomendaciones se ilustran mediante sitios web que actualmente están en funcionamiento en Internet y de ellos destacamos aquellas características que, a nuestro juicio, ayudan a que el usuario de un sitio web interactúe con él con facilidad así como aquellas otras que consideramos dificultan esa interacción. Las recomendaciones ofrecidas tienen la intención adicional de fomentar el espíritu crítico del alumno con lo que observa y con sus propias creaciones.

Una vez realizado el diseño es necesario realizar el desarrollo del sitio web creando las páginas que forman parte del sitio. Por ello, en el cuarto tema se le introduce en los cimientos sobre los que se soporta toda la estructura de información disponible en Internet: el lenguaje de marcas HTML.

Sobre el lenguaje HTML se detalla su sintaxis básica, las marcas utilizadas para describir una página web y sus atributos. Aprenden a formatear párrafos, a hacer más atractivas las páginas con la inclusión de imágenes, hipervínculos, listas, marcadores y mapas de imágenes y a organizar la información en marcos y tablas.

Sin embargo, el lenguaje HTML tiene limitaciones, que deben ser transparentes a un usuario básico de Internet, pero no deben serlo para un alumno de esta asignatura, orientado a adquirir conocimientos de nivel medio. Por ejemplo, si bien con HTML se pueden construir páginas con formularios que la dotan de cierta interactividad con el usuario, es también cierto que se pueden crear sobrecargas de trabajo en el servidor. Además, el lenguaje HTML no está bien adaptado para realizar diseños más actuales, atractivos e innovadores.

Por ello, al alumno se le presenta el lenguaje Javascript como un lenguaje que extiende las posibilidades del lenguaje HTML. Una vez introducido en las nociones básicas de programación en Javascript -variables, asignación, sentencias de control, funciones - será capaz de crear páginas web que realicen en el cliente, entre otras cosas, validación de datos de entrada, cálculos sencillos, animaciones como el efecto 'roll over' y responder a los eventos que sobre ella se produzcan.

Todos estos contenidos teóricos forman al alumno en una doble vertiente tan habitual en el mundo de hoy: por un lado, el alumno es un usuario más de un mundo poblado de redes que interconectan multitud de ordenadores, cada uno de ellos repleto de información; por otro, es capaz de ser no sólo usuario, sino creador de algunas de estos almacenes de información a través de los cuales interactuar, comunicarse y ofrecer al mundo aquello de lo que sea capaz.

4. Contenidos prácticos

Los contenidos prácticos de la asignatura de Configuraciones y Equipos Informáticos se centran en Internet e ilustran en la parte práctica dos de los tópicos tratados en la parte teórica: servicios de Internet y desarrollo de sitios web.

Las prácticas sobre servicios Internet se realizan con Netscape Communicator. Puesto que en la asignatura de Fundamentos de Informática han realizado prácticas básicas de navegación y de correo electrónico, hemos preparado unas prácticas básicas complementarias sobre servicios de Internet. Vamos a detallar aquí la que se dirige a una de las posibilidades de las que los alumnos pueden

extraer muy buen partido en su futuro profesional: la descarga e instalación de programas. Esta práctica permite, además, introducir a los alumnos en tres tipos de aplicaciones (antivirus, compresión / descompresión y lectura de documentos) que completan su formación básica y le proporcionan conocimientos imprescindibles para un usuario de Internet.

La práctica comienza con un ejercicio para descargar vía ftp un antivirus en formato comprimido con el objetivo de instalarlo en el ordenador. El proceso de descarga sirve para recordar al alumno los problemas de seguridad que pueden producirse cuando se descargan programas que pueden no ser seguros y pueden causar efectos maliciosos en su ordenador. Puesto que el antivirus llega a nuestro ordenador comprimido, es necesario hacer ejercicios en los que los alumnos usan una aplicación de compresión/descompresión para descomprimir ficheros así como para crear sus propios ficheros comprimidos. En particular, se descomprime e instala el antivirus que ha sido descargado.

Otro uso muy frecuente de la descarga de ficheros es la descarga de documentos. La práctica introduce dos de los formatos de documentos que se encuentran con frecuencia en Internet (pdf y ps) y se plantea el ejercicio de descargar e instalar un visualizador de documentos pdf y, a continuación, se propone a los alumnos que descarguen y visualicen documentos en formato pdf que puedan serles interesantes. Para finalizar la práctica, ejercicios adicionales proponen la búsqueda, descarga e instalación de un visualizador de ficheros ps, una aplicación de ping, y un editor de HTML.

Respecto al desarrollo de sitios web y tal y como hemos apuntado en la parte teórica puede hacerse bien utilizando una herramienta visual, bien utilizando un editor de HTML. Las prácticas que van a realizar permiten que desarrollen ambas capacidades.

Así, las primeras prácticas se realizan utilizando la herramienta visual FrontPage 2000 y en ellas el alumno desarrolla desde cero un sitio web de una librería que incluya todo lo explicado sobre el lenguaje HTML. En particular, el sitio web deberá contener formularios para darse de alta en la librería y para realizar compras de libros. La construcción de sitios web con FrontPage no necesita que el usuario conozca el lenguaje HTML. Antes bien, a través de un menú, el usuario introduce los elementos de las páginas web y FrontPage crea automáticamente el código HTML, que en ocasiones no es óptimo.

Esta falta de optimización permite proponer ejercicios que consisten en mejorar y hacer más legible este código y, de esta forma, los alumnos tienen oportunidad de trabajar directamente con HTML.

Una nueva práctica propone la creación por imitación de páginas escribiendo directamente el código HTML. Esas páginas han sido extraídas de sitios web actualmente en funcionamiento. Por ejemplo, proponemos construir la página de la revista Desnivel (www.desnivel.com) que se muestra en la Figura 1. Esto supone que el alumno utilice tablas para organizar la información e integre en ellas una gran variedad de imágenes y contenidos.

Con el uso de un navegador el alumno comprueba a lo largo de las prácticas que el resultado es el esperado y percibe cómo lo que diseñó (página web) y luego describió con un lenguaje textual (código HTML) es lo que finalmente obtiene en pantalla. Si eso no sucede, el alumno debe repasar el código HTML y encontrar y corregir aquellos errores que existan.

Para introducir al alumno en la programación con Javascript se le plantean ejercicios para manejar las sentencias básicas de este lenguaje construyendo programas sencillos. Ello permite que los alumnos se familiaricen con la sintaxis del nuevo lenguaje de programación y con sus peculiaridades (los datos de entrada son siempre cadenas de caracteres, por ejemplo).

Figura 1. Página que los alumnos deben construir

A continuación, y utilizando los formularios que se han incluido en el sitio web de la librería, se hacen ejercicios en los que Javascript se usa para la validación de datos de entrada (por ejemplo, para comprobar que una dirección de correo electrónico tiene exactamente una arroba y al menos un punto) o para realizar el cálculo del valor de determinados productos que se venden en el sitio web. Estos programas se incrustan en el código HTML de las páginas del sitio donde se comprueba su correcto funcionamiento.

La práctica final explica cómo publicar el sitio web en un servidor (por ejemplo, Yahoo!) en el cual los alumnos se habrán dado previamente de alta.

A través de estos contenidos prácticos, el alumno afianza todos aquellos conceptos teóricos que se le han proporcionado y aprende no sólo a interactuar con Internet, sino a ser parte y contribuir a ella.

5. Metodología y desarrollo de las clases

La asignatura de Configuraciones y Equipos Informáticos es una asignatura de seis créditos repartidos entre tres créditos de práctica y tres de teoría. Distribuidos a lo largo de quince semanas de un cuatrimestre, el alumno recibe semanalmente dos horas de teoría y dos horas de práctica.

Los contenidos teóricos se desarrollan siguiendo el método tradicional de exposición en pizarra, apoyado en transparencias para ilustrar alguno de los conceptos como, por ejemplo, las recomendaciones acerca del buen diseño de sitios web.

Un aspecto metodológico importante se refiere a los lenguajes HTML y Javascript. En la explicación de estos lenguajes, hemos utilizado un método de enseñanza basado en mostrar ejemplos en los que sea necesario utilizar las distintas marcas de HTML. Sólo una vez presentadas dichas marcas en ejemplos concretos, es cuando damos la sintaxis general. Para ello, hemos seleccionado páginas de una gran variedad de sitios web que actualmente están en funcionamiento y que ofrecen ejemplos perfectos para ilustrar desde los conceptos básicos hasta los avanzados del lenguaje de marcas HTML. Por ejemplo, para la clase teórica de HTML básico hemos utilizado la página de la Asociación de Libreros de Viejo (www.libris.es) que se muestra en la Figura 2, ligeramente modificada con objeto de lograr mayor eficacia en la transmisión de conceptos.

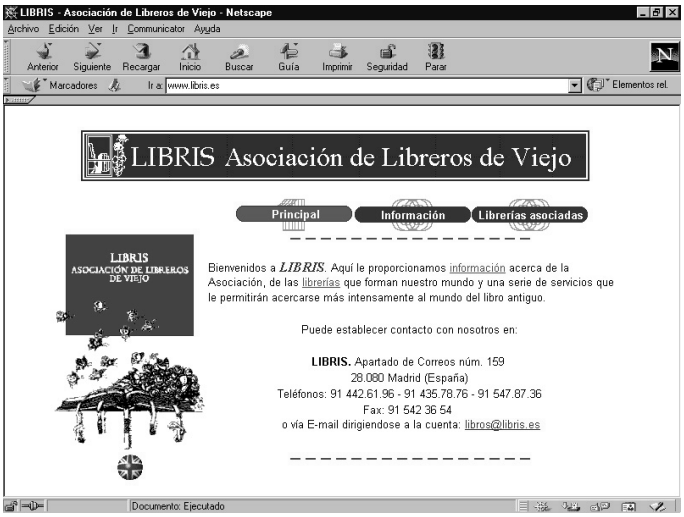


Figura 2. Página web con elementos básicos de HTML

Una vez mostrada la página web, se explican las marcas y atributos del código HTML que soportan esa página y analizamos la función de cada una de ellas. Los conocimientos de Programación que los alumnos han adquirido en otras asignaturas de la Diplomatura, les ayudan en la asimilación de las explicaciones sobre HTML. A partir del ejemplo que hemos presentado, explicamos en general el significado de todas las marcas y atributos de la página así como los posibles valores de los atributos.

Obsérvese que con esta forma de presentación mostramos los elementos HTML en el lugar donde se utilizan y a través de ejemplos. Una vez que el alumno conoce el uso de los elementos HTML gracias a los ejemplos escogidos, es cuando consideramos adecuado dar la sintaxis general de los mismos.

Las clases prácticas tienen lugar en la sala de ordenadores. Se forman grupos de dos personas porque consideramos adecuado que las personas del grupo puedan colaborar cuando en la realización de ejercicios surjan dificultades. Cada práctica trabaja sobre un tema particular de los explicados en la parte teórica, tratándolos como unidades de conocimiento. Transferencia de ficheros o creación de formularios para páginas web mediante FrontPage son algunas de las unidades desarrolladas en las prácticas para afianzar los contenidos teóricos recibidos.

El aspecto metodológico destacable en la parte práctica es el tipo de ejercicios que proponemos. Se trata de ejercicios absolutamente guiados en los que se indica a los alumnos los pasos que deben dar para alcanzar un determinado fin. La mayor parte de ellos incluyen una sección de ayuda a la que los alumnos pueden recurrir si no saben completar el ejercicio. Además, las prácticas contienen una sección 'Acerca de' que refresca y abunda en los contenidos teóricos necesarios para realizar los ejercicios prácticos.

6. Interacción con los alumnos

En este apartado vamos a detallar la interacción con los alumnos de la asignatura Configuraciones y Equipos Informáticos así como los trabajos que obligatoriamente deben realizar para lograr aprobar la asignatura.

La experiencia en la interacción con los alumnos ha sido muy positiva. La asistencia a clases prácticas ha sido muy elevada (lo cual no siempre es fácil de conseguir) y también ha sido muy importante el interés demostrado por los alumnos. Nuestra experiencia es que las

prácticas guiadas, que conducen al alumno a un determinado objetivo facilitan enormemente la asimilación de conceptos. Además aquellos ejercicios incluidos en las prácticas en los que la guía desaparece suponen para el alumno un reto que le atrae y que se esfuerza por vencer. No cabe duda, no obstante, que en esta asignatura el interés del alumno por la materia (punto clave para que el alumno aprenda y estudie) es fácil de conseguir ya que el bombardeo continuo en los medios de comunicación acerca de la importancia presente y futura de Internet nos facilita el camino en ese aspecto.

La evaluación de la asignatura se encuentra totalmente integrada en esta interacción asignatura/alumno. Además de un examen de tipo tradicional con preguntas de tipo teórico y práctico, los alumnos deben realizar una serie de trabajos cuya finalidad es formativa y evaluadora.

A lo largo de las prácticas del curso los alumnos han ido adquiriendo las habilidades necesarias para la creación de sitios web, pero no han realizado trabajos donde deban integrar todas esas habilidades. Con los trabajos de evaluación propuestos el alumno se forma, con la ayuda del profesor, en la utilización coordinada de estas habilidades para aprender, entre otras cosas, a crear sitios web reales. Los trabajos tienen también función evaluadora ya que es necesario que los alumnos los entreguen para ser evaluados de la asignatura.

Así, un primer trabajo debe mostrar que los alumnos son capaces de realizar el uso más generalizado de Internet: la búsqueda de información. Si bien esta actividad es considerada dentro de los conocimientos básicos y que se supone ya posee al llegar a la asignatura, el alumno debe demostrar que la realiza de forma ordenada, organizada y con objetivos claros y bien definidos. La propuesta es realizar varias búsquedas para las cuales deben detallar cuál ha sido el resultado, el camino seguido y las distintas fuentes de información utilizadas. Así, dada la propuesta de encontrar hoteles de un precio asequible en Londres, los alumnos deben explicar cuál ha sido su inicio en la búsqueda, el camino seguido y si han sufrido alguna pérdida en el camino hasta encontrar lo propuesto. Esta práctica no sólo tiene la función de mostrarles y que demuestren que saben encontrar información sino también que sepan utilizar el vocabulario adecuado para explicar su interacción con la red.

Los siguientes trabajos se orientan a desarrollar sus capacidades como constructores de sitios web. Así, los alumnos deben crear una página web, usando la aplicación FrontPage, partiendo de un boceto y de un conjunto de ficheros de imágenes que, teóricamente, les ha entregado un diseñador gráfico. Las páginas que se entregan son similares a la de la revista Desnivel que se muestra en la Figura 1.

En el siguiente trabajo de evaluación, al alumno se le da la libertad de que diseñe el sitio web de su interés, aplicando en él las recomendaciones explicadas en teoría acerca del diseño de sitios web.

Como trabajo final de la asignatura, los alumnos aplican sus conocimientos sobre Javascript para la validación de datos y la realización de cálculos en varios formularios.

De esta forma y progresivamente, los alumnos habrán empezado como simples usuarios de Internet hasta llegar a ser capaces de desarrollar sus propios sitios web.

La experiencia de este año en la impartición de la asignatura nos lleva a proponer para años futuros la realización por parte de los alumnos de trabajos sobre temas relacionados con el mundo de Internet y de las comunicaciones. Estos trabajos, que serían realizados en grupo, implicarían a los alumnos aún más si cabe en el desarrollo de la asignatura, y les llevaría a interesarse por temas que de otra forma quedarían fuera de los contenidos de la misma. Además, ofrecen la ventaja adicional de obligarles a desarrollar su capacidad de comunicación y de presentación en público. Puesto que el número de alumnos matriculados en la asignatura es elevado, estos trabajos se propondrían solamente para aquellos alumnos que desearan alcanzar la calificación máxima. Algunos de los temas que podrían ofrecerse son las tecnologías 'wireless', los teléfonos móviles, la tecnología UMTS o el lenguaje XML.

7. Revisión de referencias bibliográficas

En este apartado vamos a hacer un breve comentario de las fuentes bibliográficas que

hemos utilizado para elaborar los contenidos de esta asignatura.

Un libro clásico de redes es *Comunicaciones y redes de computadores* [2] donde se pueden encontrar las definiciones básicas de redes, sí bien su nivel teórico es bastante elevado. Cuando nos referimos a redes de ordenadores desde una perspectiva de usuario es interesante el libro *Introducción a las redes locales* [3] que explica de una manera sencilla el funcionamiento, normas, arquitecturas etc. de una red de área local. Para una introducción a Internet, recomendamos el libro *Internet, Edición 2000* [4].

El libro *Edición de páginas web* [5] detalla las fases de construcción de un sitio web y la obra *Diseño de páginas web* [6] lo complementa ofreciendo recomendaciones para la construcción de sitios web así como excelentes ejemplos de sitios web bien construidos. *Internet & World Wide Web. How to program* [7] es un libro de propósito general sobre programación en Internet y cuyos contenidos dan una amplia panorámica de herramientas y tecnologías para la programación en Internet. Para finalizar, *Programación en Javascript* [8] es una obra en la que se realiza una presentación sencilla y bien estructurada del lenguaje de programación Javascript, necesario para la parte final del curso.

8. Conclusiones

Varias son las conclusiones que podemos extraer del desarrollo del curso en esta asignatura y la primera y principal es el alto nivel de motivación presente en los alumnos. El contenido de la asignatura ha sido altamente motivante para el alumnado, teniendo un seguimiento de las clases por encima de la media de otras asignaturas de la Diplomatura.

De igual forma, no sólo el contenido, sino la metodología utilizada ha contribuido a aumentar este interés del alumno en la asignatura. Los contenidos teóricos tienen una aplicación inmediata en su parte práctica y el alumno percibe la utilidad de lo que se le está enseñando, siendo él mismo el que progresivamente utiliza los conocimientos adquiridos y obtiene resultados que le resultan atractivos.

Esto ha traído consigo que los alumnos en su gran mayoría han asimilado los conceptos impartidos, como prueba el alto índice de aprobados de la asignatura y nos conduce a pensar que el método práctico, progresivo y guiado es eficaz.

Pese a todo y dado que el planteamiento de esta asignatura tal y como se presenta en este trabajo, ha sido llevado a la práctica por primera vez durante el curso 2000/2001, es difícil considerar las conclusiones extraídas de un único año de impartición como definitivas.

No obstante, la experiencia obtenida junto con estas primeras conclusiones nos llevan a seguir en la línea metodológica utilizada, reelaborando los contenidos que consideremos necesario modificar y aquellos que, sin duda, será necesario incorporar a medida que el avance tecnológico continúe.

Referencias

[1] J. Ranz. *Breve historia de Internet.* Anaya Multimedia, 1997.
[2] William Stallings. *Comunicaciones y redes de computadores.* Prentice Hall, 1998.
[3] José Félix Rábago. *Introducción a las redes locales*. Anaya Multimedia, 2000
[4] Carlos Essebag, Julián Martínez. *Internet Edición 2000*. Anaya Multimedia, 2000
[5] Óscar Peña. *Edición de páginas web.* Anaya Multimedia, 2000.
[6] Jack Davis, Susan Merrit. *Diseño de páginas web*. Anaya Multimedia, 1999
[7] Deitel, Deitel & Nieto. *Internet & World Wide Web. How to Program.* Prentice Hall, 2000.
[8] José Manuel Alarcón. *Programación en Javascript.* Anaya Multimedia, 2000.

Aplicando Técnicas de Mejora de la Enseñanza de la Inteligencia Artificial en la Licenciatura en Documentación

Carlos Carrascosa, Vicente Julián, Miguel A. Salido
Departamento de Sistemas Informáticos y Computación
Universidad Politécnica de Valencia
46020 Valencia
e-mail: {carrasco, vinglada, msalido}@dsic.upv.es

Resumen

La mejora en la enseñanza es un aspecto que todos los docentes se plantean. La posible mejora en la impartición de una asignatura concreta puede ser obtenida por diversos cauces. En la asignatura Sistemas de Representación y Procesamiento Automático del Conocimiento (SRP) de la Licenciatura en Documentación de la Facultad de Informática en la Universidad Politécnica de Valencia (UPV) se ha previsto un programa de actividades para replantear los métodos docentes de la asignatura. El contenido fundamental de esta asignatura consiste en una vista panorámica donde presentar diferentes tópicos de la Inteligencia Artificial como métodos de representación y procesamiento automático del conocimiento.

1. Introducción

Debido a la juventud de la licenciatura en Documentación es imposible hablar de contenidos clásicos dentro de las asignaturas que conforman dicha titulación. Sin embargo, en el caso de la asignatura Sistemas de Representación y Procesamiento Automático del Conocimiento (SRP), teniendo en cuenta la orientación hacia el campo de la inteligencia artificial que se le ha dado, sí que es posible el estudio de esta materia desde una orientación puramente informática. No hay que olvidar que en general el bagaje de conocimientos previos con que el alumno accede a dicha asignatura puede ser muy limitado, debido al hecho de que la carrera en la que se encuentra esta asignatura no corresponde con una de corte puramente informático. De esta manera, la orientación de esos temas no puede ser exactamente la misma que la que se utilizaría en ese otro tipo de asignaturas, ni la forma de impartirla, puesto que la base teórica de estos alumnos es limitada en este campo.

Un aspecto a destacar de la Universidad Politécnica de Valencia (UPV) es el esfuerzo realizado en los últimos años en analizar los aspectos más deficitarios del sistema educativo vigente y tomar medidas orientadas a fomentar y subsanar, en lo posible, estos déficits. Entre ellos destaca la puesta en marcha de un ambicioso proyecto, conocido como proyecto EUROPA (Una Enseñanza ORientada al APrendizaje), para la mejora integral de la enseñanza y el aprendizaje. Este proyecto es fruto de la evolución de proyectos anteriores. La forma de abordar el cometido del proyecto se concreta en cinco programas de actuación, entre los que se encuentra el programa de Ayuda a la Mejora de la Enseñanza (AME), para favorecer sinergias entre cada centro y los departamentos que imparten docencia en él. Dentro de este proyecto existen distintos subproyectos, entre los que se encuentran:

- AME-2 (Nuevos métodos de enseñanza-aprendizaje): el cual trata de incentivar la implantación de métodos de enseñanza-aprendizaje orientados a una mayor participación activa del alumnado

Técnica	Actividad del profesor	Actividad del alumno
Lección Magistral	Planificación Desarrollo estructurado Evaluación	Atención en clase Participación
Método de la pregunta	Planificación Estudio del momento adecuado	Participa Se convierte en un elemento activo
Resolución de problemas en grupo	Planificación previa Presentación del problema Orientación Seguimiento Evaluación	Discute Trabaja en grupo
Contrato de aprendizaje	Presenta objetivos Orienta trabajo Seguimiento Técnicas de evaluación	Discute Trabaja en grupo
Tutorización	Planificación y estructuración de las tutorías Seguimiento continuo Evaluación personalizada	Planifica Participa y pregunta
Prácticas de laboratorio	Realización de boletines Realización de seminarios de iniciación a la tecnología necesaria Evaluación continuada de las prácticas	Participación Planificación a medio plazo de su trabajo Trabajo en grupo

Figura 1: Técnicas y actividades de mejora planificadas

- AME-3 (Mejora de los sistemas de evaluación): en este caso consiste en incentivar la implantación de sistemas de evaluación del rendimiento del alumnado que se aproxime a la evaluación continua multicriterio. Teniendo en cuenta todas las actividades que lleva a cabo.

Para el caso que nos ocupa, esto es, la asignatura SRP de la Licenciatura en Documentación de la Facultad de Informática en la UPV, se ha desarrollado un proyecto, concedido recientemente, encaminado a mejorar todos los aspectos previstos en estos dos subproyectos. Básicamente los objetivos que se persiguen son los siguientes:

- Reestructuración de las clases magistrales estudiando la incorporación de técnicas alternativas como resolución de problemas en pequeños grupos, empleo del método de la pregunta, debates, pequeños foros, estudio de pequeños casos, etc.
- Reestructuración de las prácticas de modo que fortalezcan en mayor medida el proceso de aprendizaje ya que se ha detectado cierto distanciamiento entre las prácticas y la teoría. Para ello se plantea el dar un peso importante en la evaluación a las nuevas prácticas a desarrollar.
- Empleo de nuevas metodologías en el proceso enseñanza-aprendizaje como la metodología de aprendizaje autodirigido de los contratos de aprendizaje, mediante el desarrollo de un pequeño proyecto de trabajo guiado a lo largo de casi la totalidad del curso realizado por los alumnos en grupo.
- Utilización del acción tutorial como mecanismo de seguimiento de los diferentes trabajos, de tal forma que se pueda emplear la tutoría como docencia en sí.

- Cambios en el sistema de evaluación dando un mayor peso al trabajo en grupo realizado por el alumno. Se establece por tanto la necesidad de técnicas para medir el grado de esfuerzo y dedicación del alumno en el grupo para poder ser evaluado correctamente.

En la figura 1 se recogen el conjunto de diferentes técnicas a aplicar para la consecución de los objetivos anteriores. Para el desarrollo de este proyecto de mejora en el proceso de enseñanza-aprendizaje y de los métodos de evaluación se ha pensado en un plan a corto y medio plazo para la incorporación de todos los elementos necesarios que permitan el cumplimiento de los objetivos marcados. De esta forma, desde un punto de vista temporal, se plantea lo siguiente:

- Estudio y aplicación de la técnica de contrato de aprendizaje en el actual curso académico.
- Estudio e incorporación gradual a lo largo de como mínimo dos cursos académicos de nuevas técnicas alternativas a la lección magistral en las clases de teoría, estableciendo mecanismos de evaluación de las mismas al finalizar el primer curso académico.
- Reestructuración para el siguiente curso académico del programa de prácticas, para ello se necesitarán recursos humanos y técnicos.
- Incorporación de las tutorias de grupo e individualizadas de forma más pausada, realizando una prueba este curso académico para evaluar el método (fundamentalmente estudiar la respuesta del alumno) e incorporándolo totalmente el siguiente curso académico.

A continuación se presenta el contenido que se le ha dado a la asignatura siguiendo las pautas comentadas en el proyecto de mejora del aprendizaje.

2. Enfoque de la asignatura SRP

Teniendo en cuenta la orientación que se pretende dar a la asignatura en función de su contexto y el carácter introductorio de sus contenidos, los objetivos principales de la asignatura son los siguientes:

- Introducción a la disciplina de la Inteligencia Artificial, su historia, su presente y su futuro, qué se puede esperar de ella, así como la relación de la inteligencia artificial con otras disciplinas.
- Estudio introductorio de las técnicas básicas de IA como métodos de representación y procesamiento automático del conocimiento.
- Uso práctico de la IA, queriendo orientar estos conocimientos al campo de la Documentación.

En los siguientes puntos se presenta la propuesta de contenidos de la asignatura (teóricos y prácticos) ajustándose a los objetivos de la misma y a los del proyecto de mejora.

3. Teoría

La parte de teoría de la asignatura “Sistemas de Representación y Procesamiento Automático del Conocimiento” se distingue entre dos partes bien diferenciadas en contenidos, forma de trabajo y evaluación. La primera de ellas es la que imparten los profesores de la asignatura como clases de teoría, con los contenidos que se detallarán a continuación y cuya evaluación se realizará por medio de un examen final. La segunda de ellas es la formada por un pequeño proyecto de trabajo guiado realizado por los alumnos en grupo (usando la metodología de aprendizaje autodirigido de los contratos de aprendizaje). El contenido de estos trabajos versará sobre la aplicación de alguna técnica de IA en el campo de la documentación, y cuya evaluación dará lugar al 50% de la nota de teoría (la nota del examen anterior da el otro 50%).

3.1. Clases de Teoría

A continuación se presenta una descripción de cada uno de los temas que constituyen esta parte de la asignatura. Los contenidos de las clases de teoría se dividen en dos bloques: el primero se centra en los conceptos clásicos de la Inteligencia Artificial, mientras que el segundo se dedica a la exposición de diferentes técnicas actuales de la Inteligencia Artificial, orientado a su uso en la documentación.

Para un mayor detalle de los contenidos referirse a [Carrascosa 2001].

3.1.1. La Inteligencia Artificial Clásica

Esta primera parte del bloque de clases de teoría presenta una introducción a un conjunto de métodos de representación del conocimiento y resolución de problemas considerados como clásicos en la Inteligencia Artificial. En concreto, esta parte está formada por los siguientes temas:

- Tema 1: Introducción a la Inteligencia Artificial:
  Breve introducción al concepto de Inteligencia Artificial junto con un repaso a la historia de la IA. El tema concluye con la exposición de las diferentes áreas en las que se divide la IA, junto a los diferentes tipos de aplicaciones de las técnicas de la IA haciendo especial hincapié en aquellas dentro del ámbito de la documentación.
- Tema 2: Representación del Conocimiento en Inteligencia Artificial:
  Este tema es el núcleo de esta parte de la asignatura por contenidos y por duración. En él se presenta la problemática de la representación del conocimiento junto con un conjunto de aproximaciones en grado creciente de complejidad:
  - La lógica (proposicional y de predicados).
  - Los sistemas basados en reglas (viendo distintas configuraciones según la forma de representar los hechos).
  - Los sistemas de representación estructurados (redes semánticas y *frames*).
- Tema 3: Sistemas Basados en el Conocimiento: Sistemas Expertos:
  Introducción a los Sistemas Basados en el Conocimiento y los Sistemas Expertos, prestando especial atención a las características funcionales y estructurales de los mismos. También se incluye en este tema una breve descripción de metodologías de desarrollo de Sistemas Expertos, así como algunas áreas de aplicación tales como planificación, diagnosis, control, ...

3.1.2. Ejemplos de Técnicas de IA de Relevancia Actual

Este tema comprende un conjunto de introducciones a una serie de técnicas avanzadas de Inteligencia Artificial de aplicación al campo de la documentación. De esta manera el tema se estructura en cuatro apartados. Cada uno de estos apartados se centra en una de estas técnicas.

Sistemas de *Blackboard*

Se presenta este modelo de resolución de problemas como una generalización de los sistemas de producción. Además se describe su estructura y diversas variaciones de acuerdo a las distintas arquitecturas de control posibles. Posteriormente se presenta un ejemplo de una arquitectura de Sistemas de *Blackboard* (BB1). El apartado concluye con la presentación del diseño de una aplicación meta-buscador usando el modelo de *blackboard*.

Agentes Inteligentes

Se introduce el concepto, diversas arquitecturas o aproximaciones distintas y algunas aplicaciones, prestando especial interés en aquellas de carácter documental o de gestión de información.

Redes Neuronales Artificiales

Se introduce este modelo de computación subsimbólica, indicando sus principales características y múltiples aplicaciones en multitud de campos (incluido el de la Documentación).

Algoritmos Genéticos

Se presenta este modelo de computación subsimbólica, indicando su esquema general y principales aplicaciones en el campo de la Documentación.

3.2. Trabajos Guiados

Una de las partes más importantes de la asignatura (como se refleja en el porcentaje de la nota final que le corresponde) la comprende el trabajo final que deben realizar los alumnos. Este trabajo se realiza durante gran parte de la mitad del cuatrimestre, y su objetivo es ver casos reales (en aplicaciones o líneas de investigación abiertas) de la aplicación de técnicas de IA al campo de la Documentación.

En la realización de este trabajo se trata de establecer un aprendizaje autodirigido [Knowles 1975], el cual es un método en el que el estudiante asume la iniciativa en el diagnóstico de sus necesidades de aprendizaje, la formulación de los objetivos, la elección y búsqueda de los recursos humanos y materiales para el aprendizaje, selecciona las estrategias para aprender mejor y evalúa los resultados obtenidos con la ayuda del profesor que actúa como proveedor y recurso de este tipo de aprendizaje.

Una de las técnicas de aprendizaje autodirigido es la de los *contratos de aprendizaje*. Esta técnica ayuda a organizar de forma más eficiente el aprendizaje, induce a los alumnos a ser más creativos a la hora de identificar los recursos de aprendizaje y desarrollar estrategias de aprendizaje, y los fuerza a obtener mayores evidencias de sus logros. Además, se puede utilizar en cualquier área de conocimiento. El contrato de aprendizaje es un medio de reconciliar los requerimientos impuestos por las instituciones y la sociedad con la necesidad de los que aprenden de realizar dicho proceso de forma autodirigida. Esta técnica les permite mezclar estos requerimientos con sus propios objetivos personales para elegir sus propios caminos para lograrlos, y medir su propio progreso. El contrato de aprendizaje hace visible por tanto las responsabilidades mutuas del estudiante, el profesor y la institución.

De esta manera, en una sesión a mediados de cuatrimestre, se les propone a los alumnos que formen grupos de un máximo de 5 personas y se les presenta una serie de posibles temas para los trabajos. Ésta no es una lista cerrada, sino que se les da la opción a los alumnos para que propongan ellos un tema para su trabajo (que los profesores de la asignatura deben juzgar si es adecuado o no a la temática de la misma). En la presentación de los trabajos propuestos se les entrega a los alumnos un documento con una breve explicación de cada uno junto a una pequeña bibliografía.

Una vez transcurridas unas 4 semanas desde que se decidió la composición de los grupos y se repartieron los temas de los trabajos, se realiza una sesión en la que los grupos presentan un índice preliminar del trabajo a realizar, y discuten con los profesores de la asignatura el enfoque a dar a dicho trabajo. En ese momento se establece el contrato entre los alumnos y el profesor. En el contrato se debe dejar claro los componentes del grupo de trabajo y cuáles van a ser los objetivos concretos del trabajo a realizar (mediante un proceso de negociación con el profesor). Además, hay que describir cómo se planea obtener esos objetivos, especificando los recursos a utilizar (a lo que el profesor también puede ayudar).

Por último, la/s última/s sesión/es de la asignatura se dedica/n a la exposición oral de los trabajos. Para ello, los profesores de la asignatura deciden in-situ el orden y la/s persona/s dentro del grupo que debe/n exponer el trabajo.

HOJA EVALUACIÓN TRABAJO

Título:

Componentes:

Memoria entregada: SI NO / Disco entregado: SI NO

Concepto	*Comentario*	*Nota*	*Peso*
Calidad Presentación HTML			0.05
Ajuste al tema			0.1
Calidad Presentación Oral			0.15
Calidad Presentación Trabajo escrito			0.05
Calidad Información Contenida			0.1
Calidad Estructuración			0.1
Claridad en las explicaciones			0.1
Bibliografía			0.1
Aportaciones Propias			0.15
Valoración general			0.1

NOTA FINAL

Figura 2: Hoja de Evaluación del Trabajo en grupo.

Después de cada exposición la audiencia puede realizar las preguntas que se consideren oportunas.

Para evaluar estos trabajos se utiliza una hoja que recoge todos los factores que se tienen en cuenta (figura 2):

4. Prácticas

El planteamiento de las prácticas en la asignatura tiene en cuenta los siguientes puntos:

- Dar a conocer una herramienta como KAPPA, cuyas características resultan bastante novedosas al tipo de alumno al que va dirigido.
- Que el alumno practique el desarrollo de problemas de representación cuya explicación se ha realizado en teoría.

- Demostrar la utilidad y aplicación de los conceptos teóricos mediante el desarrollo de un problema relativamente complejo cercano a la realidad.

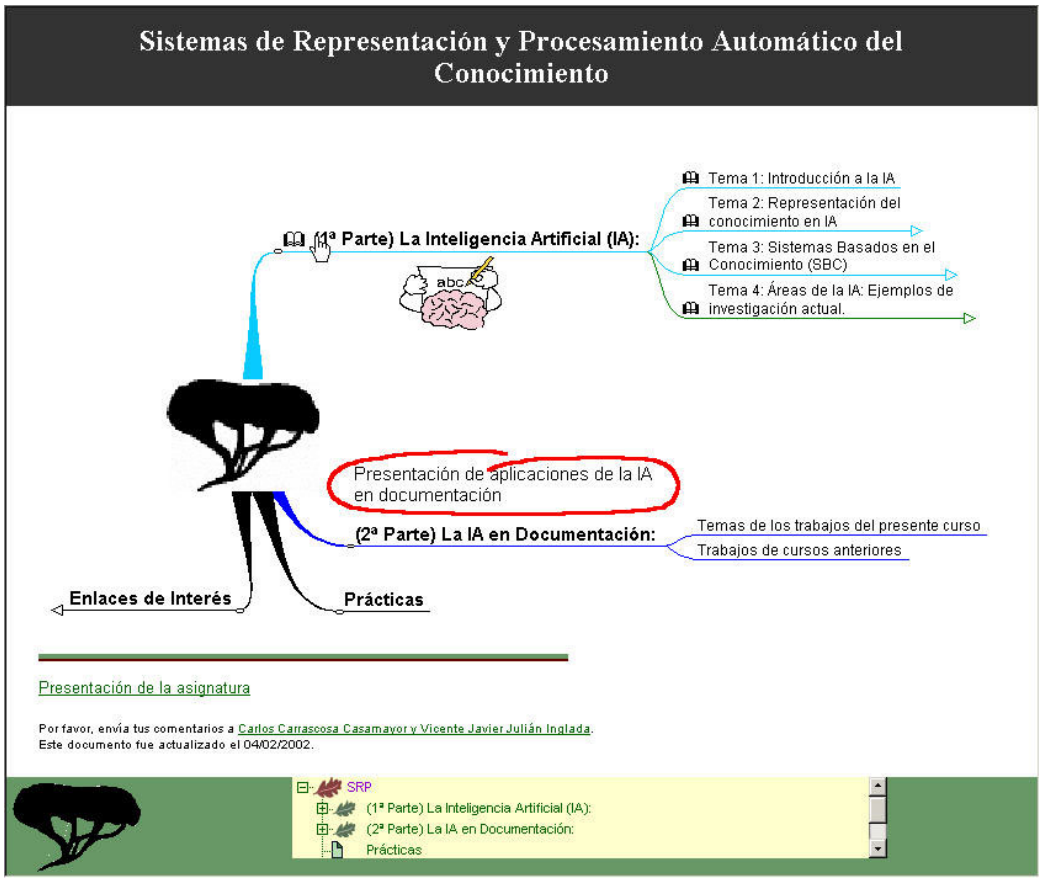


Figura 3: Página web principal de SRP

El programa de prácticas de la asignatura se organiza en tres grandes bloques de sesiones (cada sesión equivale a dos horas de laboratorio):

- 1er Bloque: Iniciación a la herramienta KAPPA. Sesiones donde se pretende que el alumno se inicie en el uso de dicha herramienta. El objetivo principal de este bloque de prácticas es que el alumno pueda disponer a través de una serie de sesiones de la base necesaria para poder entender y utilizar KAPPA. Este bloque se estructura como si fuese un seminario, donde se plantean ejemplos sencillos y se plantean al alumno pequeños problemas a resolver, de tal forma que se habitúen al empleo de la herramienta.
- 2º Bloque: Desarrollo de sistemas de marco (frames). En este bloque se plantea un problema más complejo al alumno de representación de información. Mediante el empleo del modelo de frames visto en

teoría y la herramienta KAPPA los alumnos, en grupos de máximo dos personas, deben entregar una implementación de una estructura de frames que de solución al problema planteado. Este bloque es evaluado y la nota obtenida cuenta para la evaluación final.

- 3er Bloque: Desarrollo de un pequeño sistema experto. De forma similar al bloque anterior, el alumno debe realizar una implementación en KAPPA de un pequeño sistema experto relacionado con el área documental, desarrollando reglas y métodos que crean oportunos, también se les exige una interfaz gráfica empleando las utilidades que ofrece KAPPA en este aspecto. Este bloque también es evaluado y su peso es mayor que el bloque anterior.

El motivo de emplear KAPPA como herramienta en el curso es debido a su sencillez de uso y a su interfaz gráfico, el cual facilita notablemente la visualización de las estructuras de frames.

5. Página Web de SRP

La asignatura SRP tiene disponible una página web donde el alumno durante el curso dispone de toda la información necesaria de la asignatura. Además la pagina web dispone de un link donde se encuentra un histórico de los trabajos que antiguos alumnos llevaron a cabo en cursos anteriores para tener una base de referencia para sus trabajos. Esta página tiene una estructura clara y ordenada con el objetivo de que el alumno encuentre la información que necesite, como los contenidos de la asignatura por temas, con referencias útiles para cada tema, información de los profesores que imparten la materia, etc. En la Figura 3 se presenta el portal de entrada de la asignatura.

6. Futuros Cambios

Lo expuesto hasta aquí corresponde a los contenidos de la asignatura para el actual curso académico. Para el siguiente curso se tiene previsto realizar una serie de modificaciones que afectarán básicamente a dos aspectos:

- Clases de teoría:
  Está previsto realizar los cambios oportunos al temario para dar cabida a una introducción al *aprendizaje* y al *data mining*.
- Prácticas:
  Aunque el contenido de la parte teórica seguirá siendo muy superior al de la parte práctica, se tratará de realizar importantes ajustes al contenido de esta última parte entre los que destacan el añadir / cambiar la herramienta a utilizar, para lo cual se está estudiando el empleo de la herramienta ZEUS. Ésta es una herramienta creada por British Telecom para el desarrollo de sistemas multiagente que ya se utiliza para presentar diversos ejemplos en varios temas de las clases de teoría.

Bibliografía

[Knowles 1975] Knowles, M. S. “Self-Directed Learning: A Guide for Learners and Teachers”. N. Y. Cambridge Book Company. (1975).

[Carrascosa 2001] C. Carrascosa, V.J. Julián, V. Botti. La Inteligencia Artificial en la Licenciatura en Documentación. Actas del Encuentro sobre Docencia en Inteligencia Artificial, Gijón. p. 131-138. 2001.

Propuesta para la asignatura de Introducción a los Computadores de la ETSII de la UPV

M. Carmen Juan Lizandra José Antonio Gil Gómez
Dept. Sistemas Informáticos y Computación
Universidad Politécnica de Valencia
46022 Valencia
e-mail: mcarmen@dsic.upc.es e-mail: jgil@dsic.upv.es

Resumen
Las asignaturas de informática son indispensables en cualquier titulación. En la Escuela Técnica Superior de Ingenieros Industriales de la Universidad Politécnica de Valencia, se imparten varias asignaturas de informática, la más básica es Introducción a los Computadores. Con el objetivo de actualizar sus contenidos, se ha realizado un estudio de distintos planes de estudio de universidades extranjeras y españolas y considerando este estudio y otros aspectos, se ha realizado una propuesta de cambio.

1. Introducción

Los conocimientos básicos de informática son indispensables para cualquier titulado y en el caso de que estos conocimientos no los hayan adquirido en su formación anterior, como suele ser el caso, en el primer curso de carrera deberían adquirirlos. Con este objetivo se imparte la asignatura de Introducción a los Computadores (ICO) en la Escuela Técnica Superior de Ingenieros Industriales (ETSII).

La mayoría de las titulaciones de Ingeniero Industrial o Ingeniero Técnico Industrial suelen incluir asignaturas básicas de informática en sus planes de estudio, como troncales, obligatorias, optativas o de libre elección. En estas asignaturas se imparten, como su nombre indica, conocimientos básicos de informática y en ellas también se suele incluir el estudio de algún lenguaje de programación (C, Pascal, etc.). Los conocimientos básicos que se suelen impartir se podrían agrupar del siguiente modo:

1. Conocimientos básicos sobre Software y Hardware.
2. Conocimientos básicos de Sistemas Operativos a nivel teórico y en algunos el manejo de alguno de ellos.
3. Ofimática. Enseñando el manejo de: Procesadores de texto, Hojas de cálculo, Programas de presentación, Programas de dibujo, Navegadores, Entornos de programación.
4. Programación. En la mayoría de los casos se enseña la sintaxis de un lenguaje de programación.

En este artículo se presenta el temario actual de ICO, un estudio de distintos planes de estudios de universidades extranjeras y españolas, y se propone un nuevo temario.

2. Temario actual de ICO

ICO es una asignatura optativa, de primer curso, primer cuatrimestre, la dotación de teoría y laboratorio es de 2 y 1.75 créditos, respectivamente. Su temario es el siguiente:

Parte 1: *Introducción al Entorno de Programación de Turbo C++*.

Parte 2: MS-DOS. Puesta en marcha desde Windows. Símbolo del sistema. Formato general de una orden. Modificadores. Órdenes más usuales. Comodines. Redirecciones y tubos.

Parte 3: *Windows*. ¿Qué es Windows?. ¿Cómo entrar y salir de Windows? (Versión Oficial (PCs normales). Aula Informática de la ETSII). El Escritorio. Iconos. Ventanas y Controles (El Portapapeles. Elementos de Windows. El Explorador de Archivos. El Sistema de Ayuda. Configuración del Sistema. Utilidades (ScanDisk. Defragmentador. VirusScan. WinZip)).

Parte 4: *Informática y Ordenadores Personales*. Informática. El Ordenador (HW) (La Unidad Central de Proceso. La Memoria Central. Dispositivos de Entrada y Salida). Los Programas (SW) (Creación de un programa. Tipos de Programas (universales y localizados)).

Parte 5: ***Conceptos de Sistemas Operativos.*** ¿Qué es un Sistema Operativo?. Componentes del Sistema. Programas del Sistema. Intérprete de Órdenes.

Las prácticas actuales son las siguientes:

Práctica 1. El IDE de Turbo C++. Cuentas y entrada al sistema. Descripción del Aula de Informática. Puesta en marcha de Turbo C++. Ejemplos y ejercicios.

Práctica 2. MS-DOS. Puesta en marcha de MS-DOS. Órdenes. Ejercicios.

Práctica 3. Introducción a Windows 95. Tutorial. El sistema de Ayuda. El Explorador de Windows.

Práctica 4. Configuración del sistema. WordPad. Asociación de programas. Configuración del sistema. WordPad. Asociación de tipos de archivo a programas.

Práctica 5. Paint. Utilidades. Paint. WinZip. ScanDisk. VirusScan.

Práctica 6. World Wide Web. Internet Explorer. El servidor de la asignatura. Navegación.

3. Estudio de planes de estudios de asignaturas de informática básica en Ingeniero Industrial

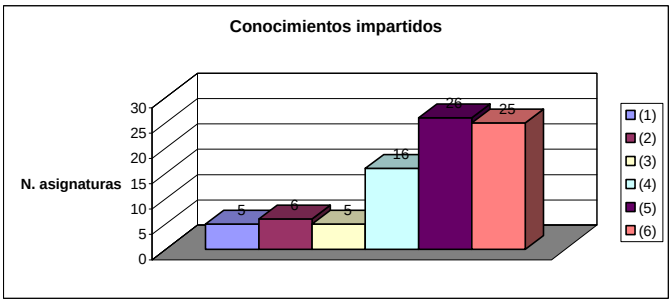
En este trabajo se ha realizado un estudio comparativo de las enseñanzas de informática básica en la titulación de Ingeniero Industrial en distintas universidades extranjeras y españolas. Concretamente 3 americanas, 9 europeas y 21 españolas. Se han consultado todas las universidades españolas que disponían de página Web. Respecto a las extranjeras, la selección ha sido aleatoria. De este estudio se ha conseguido identificar cuáles son los conocimientos informáticos básicos que deben incluirse en la formación de un Ingeniero Industrial.

En primer lugar, se hará una clasificación de los contenidos impartidos en dichas asignaturas:

(1) Teoría de Informática y ordenadores personales (Software y Hardware)
(2) Sistemas Operativos (Windows 95/NT, UNIX, DOS, etc.)
(3) Programas básicos (procesadores de texto, hojas de cálculo, programas de presentaciones, programas de dibujo, navegadores, entornos de programación, etc.)
(4) Conceptos básicos de programación (paradigmas de programación, definición de algoritmo y programa, definición de traductores e intérpretes, etc.)
(5) Conceptos de Programación (tipos de datos simples, sentencias de control, estructuras de datos, subprogramas, etc.)
(6) Lenguaje de Programación (C, Pascal, etc.)

Las universidades consultadas han sido: Université de Moncton. Canadá; University of California, Berkeley. Estados Unidos; Carnegie Mellon University. Pittsburgh. Estados Unidos; Technische Universität Braunschweig. Braunschweig. Alemania; L'Ecole Nationale Supérieure de Génie Industriel. Francia; Ecole Nationale Supérieure d'Ingénieurs Electriciens de Grenoble. Francia; Université Paul Sabatier. Francia; University of Bristol. Gran Bretaña; University College London. Londres. Gran Bretaña; University of Southampton. Highfield, Southampton. Gran Bretaña; Università degli studi di Parma. Facoltà di Ingegneria. Italia; Università di Pisa. Facoltà d'Ingegneria. Italia y Lunds Tekniska Högskola. Lunds Universitet. Suecia.

Los datos extraídos de estas universidades y agrupados por contenidos, se muestran en la gráfica 1.



Gráfica 1. Conocimientos impartidos en las distintas universidades extranjeras consultadas

De estos datos se deduce claramente que en las asignaturas de informática, mayoritariamente, se imparten conocimientos de programación y la sintaxis de un lenguaje de programación. Sin embargo, estos conocimientos se imparten en las asignaturas de Fundamentos de Informática I (FI1) y Fundamentos de Informática II (FI2), por lo que los aspectos de esta clasificación que afectarían a la asignatura objeto de estudio, ICO, serían los grupos (1), (2) y (3). Si consideramos detalladamente la información resumida en la gráfica 1 y únicamente las asignaturas en las que se imparten conocimientos básicos de informática, se obtiene la tabla 1.

Universidad	**Asignatura**	**(1)**	**(2)**	**(3)**
Université de Moncton. Canadá	Principios de programación	Hw	Conceptos	
Carnegie Mellon University. Pittsburgh. Estados Unidos	Workshop sobre habilidades con ordenadores		UNIX	Access, Pag. Web Excel, PhotoShop PowerPoint
Technische Universität Braunschweig. Braunschweig. Alemania	Ciencia de los ordenadores en Ingeniería Mecánica (Mech)	Hw Sw	UNIX/ Windows	Procesadores Texto Bases de datos Librerías Software
L'Ecole Nationale Supérieure de Génie Industriel. Francia	Informática: Algorítmica y programación			Ofimática
	Lenguajes Orientados a Objetos		Conc./UNIX	
Université Paul Sabatier. Francia	Informática I		Conceptos	
University of Bristol. Gran Bretaña	Estudios profesionales, Electrónica e Informática I (Mech)			Excel
University College London. Londres. Gran Bretaña	Informática	Hw Sw		
Università degli studi di Parma. Facoltà di Ingegneria. Italia	Fundamentos de Informática	Hw Sw		Word Excel Access
Università di Pisa. Facolta` d'Ingeg. Italia	Fundamentos de Informática	Hw Sw	Conceptos Windows	

Tabla 1. Agrupación por contenidos del estudio en universidades extranjeras

Agrupando los datos de la tabla 1, se tiene la tabla 2.

	Hw	**Sw**	**Conc. S.O.**	**Uso S.O.**	**Ofimática**
N. asigna.	5	4	4	4	5

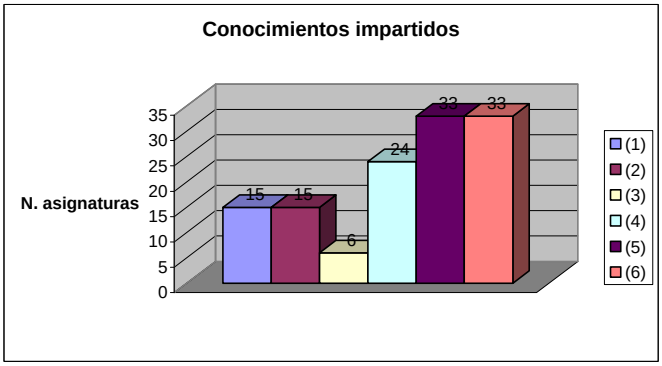
Tabla 2. Agrupación por contenidos del estudio de la tabla 1

De estos datos se deduce que fundamentalmente se imparten conceptos de Hardware, Software, de Sistemas Operativos, se enseña el uso de un Sistema Operativo y se enseña ofimática. También se observa que en la parte de ofimática, se imparte, principalmente, Excel.

Por su parte las Universidades Españolas consultadas son las siguientes: Universidad de Cantabria; Universidad Carlos III de Madrid; Universidad de Castilla-La Mancha; Universidad de Extremadura; Universidad de Girona; Universitat Jaime I; Universidad de la Rioja; Universidad de Las Palmas de Gran Canaria; Universidad de Málaga; U.N.E.D.; Universidad de Oviedo; Universidad del País Vasco; Universitat Politècnica de Catalunya.

Barcelona; Universitat Politècnica de Catalunya. Terrassa; Universidad Politécnica de Madrid; Universidad de Salamanca; Universidad de Valladolid; Universidad de Zaragoza; Universidad Alfonso X el Sabio y Universidad de Navarra.

En la gráfica 2 figuran los valores extraídos del estudio realizado.



Gráfica 2. Conocimientos impartidos en las distintas universidades españolas consultadas

Considerando únicamente los datos (1)-(3) de aquellas asignaturas relacionadas con ICO, como en las Universidades Extranjeras, los datos son los que se muestran en la tabla 3.

Universidad	**Asignatura**	**(1)**	**(2)**	**(3)**
Universidad de Cantabria	Fundamentos de Informática	Hw Sw	Conceptos	
	Sistemas Informáticos I		UNIX	Multimedia Bases de datos
Universidad Carlos III de Madrid	Programación	Hw Sw		
Universidad de Castilla-La Mancha	Fundamentos de Informática	Hw Sw	Conceptos/ MS-DOS	WordPerfect, Harvard Graphics, Matlab
Universidad de Extremadura	Fundamentos de Informática	Hw	Conceptos/ Windows 98	
Universidad de Girona	Introducción a los Ordenadores	Hw	Conceptos Windows/ UNIX	Internet
Universitat Jaime I	Fundamentos de Informática	Hw Sw	Conceptos	
Universidad de la Rioja	Fundamentos de Informática		Windows	
Universidad de Las Palmas de Gran Canaria	Informática Básica	Hw	MS-DOS/ Windows	Bases de datos Ofimática, Internet
Universidad de Málaga	Fundamentos Informáticos	Hw Sw	Windows	
Universidad de Oviedo	Informática	Hw	Conceptos OPEN-VMS	

Universidad	Asignatura	(1)	(2)	(3)
	Computadores I	Hw		
Universidad del País Vasco	Fundamentos de Informática		Conceptos	
Universitat Politècnica de Catalunya. Terrassa	Informática Básica	Hw Sw		
Universidad Poliécnica de Madrid	Tecn. de la Información			Internet Bases de datos
Universidad de Salamanca	Informática I (Electri)	Hw Sw	Conceptos MS-DOS/ Windows	
	Métodos informáticos(Electri)	Hw Sw	Conceptos MS-DOS/ Windows	
Universidad de Valladolid	Introducción a la Informática	Hw		
	Fundamentos de Informática		UNIX	
Universidad de Navarra	Informática 1	Hw Sw	Windows	Word Excel

Tabla 3. Estudio comparativo sobre las universidades españolas consultadas

Agrupando los datos de la tabla 3, se obtiene la tabla 4.

	Hw	Sw	Conc. S.O.	Uso S.O.	Ofimática
N. asigna.	15	9	9	12	6

Tabla 4. Agrupación por contenidos del estudio de la tabla 3

De estos datos se deduce que, fundamentalmente se imparten, por orden de importancia, conceptos de Hardware, se enseña el uso de un Sistema Operativo, se imparten conceptos de Software y de Sistemas Operativos y por último se enseña ofimática.

4. Propuesta de nuevo temario para ICO

Como consecuencia de la constante evolución de la informática, del estudio realizado y de la experiencia adquirida impartiendo el plan de estudios actual, se propone un nuevo temario para estos estudios, más acorde con los conocimientos de los alumnos a los que va dirigido y mejor adaptado a las necesidades que plantea en la actualidad la informática para usuarios.

4.1. Objetivos a cubrir

Los objetivos planteados en este nuevo temario serían similares a los del anterior pero teniendo en cuenta los factores mencionados.

Cuando el alumno acaba satisfactoriamente la instrucción será capaz de:

- Conocer los conceptos mínimos de informática que deben poseer los titulados en una Ingeniería no informática para poder desenvolverse en un ordenador personal bajo sistemas operativos de tipo Windows.
- Conocer de forma básica las herramientas informáticas de uso común que pueden ser de gran utilidad tanto a lo largo de la carrera como en su futuro profesional.
- Enlazar conocimientos de esta asignatura con otras asignaturas informáticas de la titulación, de forma que cumpla un papel introductorio en el uso del entorno de programación.
- Conocer el método de trabajo en las aulas informáticas.

La carga lectiva de la asignatura, así como el reparto de la misma entre la parte teórica y la de laboratorio, se mantendrá constante, por lo que habrá 2 créditos asignados a la parte de teoría y 1.75 créditos a la parte de laboratorio.

La parte práctica se destina a que el alumno plasme en el laboratorio todos aquellos conocimientos que se van a ir introduciendo en las clases teóricas.

En una asignatura como ésta son las sesiones de laboratorio las que marcan el ritmo de impartición de las clases teóricas, ya que serán éstas las que sirvan de punto de entrada a las diferentes prácticas en las que se trabajará con los conceptos y herramientas introducidos. Además, se debe empezar enseñando el manejo del entorno de programación que se utiliza en FI1, que es una asignatura, también, de primer curso, primer cuatrimestre.

4.2. Parte teórica

La propuesta del nuevo temario en lo concerniente a la parte de teoría quedaría de la siguiente forma:

Tema 1: **Introducción a un entorno de programación** [2]. En este tema se pretende introducir al alumno a los conceptos de programa y programación, así como las principales características de un entorno de programación. Estos conceptos se particularizarán en el entorno del Turbo C++ (utilizado en FI1 y FI2).

Este tema sería conveniente, en principio, impartirlo una vez se haya introducido con cierta profundidad al alumno en el mundo informático, pero es necesario que el alumno conozca estos contenidos al inicio del curso para poder hacer uso de los mismos en otras asignaturas que se imparten de forma simultánea, como se ha comentado.

Tema 2: **Componentes Hardware** [1]**.** Aquí se ofrecerá al alumno una visión de los diferentes componentes de un ordenador personal. Esta visión será lo más completa posible de cara a conseguir dos objetivos:

- Que el alumno sea capaz de distinguir los diferentes periféricos y componentes principales con los que se puede encontrar.
- Que quede clara la estructura de la Unidad Central y sus diferentes partes (memoria central, unidad central de proceso, ...), así como su modo de funcionamiento en general.

Dentro de este tema se complementará la formación del alumnado no sólo con la proyección de fotografías e imágenes de los diferentes elementos sino que también se le proporcionará muchos de estos componentes hardware (desde discos duros hasta procesadores) de cara a que se familiarice lo máximo posible con ellos.

Tema 3: **Sistemas Operativos, Windows y su explorador** [1]**.** En este tema se comentarán los conceptos teóricos sobre Sistemas Operativos, particularizando para Windows, y el explorador. El manejo práctico de los mismos se impartirá en sesiones de laboratorio.

Tema 4: **Utilidades** [1]**.** Una vez introducidos en Windows, se presentarán a los alumnos diferentes utilidades que les podrán ser de gran utilidad en su uso diario del ordenador. Las utilidades propuestas para ser mostradas serían las siguientes:

Utilidades de mantenimiento del sistema. Este punto se centrará en herramientas como el Scandisk y el Defragmentador, que permitirán a los alumnos la revisión y el mantenimiento de sus unidades de disco.

Utilidades de compresión. Este apartado servirá para explicar qué son y en qué consisten las utilidades de compresión de archivos. Se comentarán de forma teórica los aspectos más relevantes de WinZip, de nuevo el manejo de sus distintas opciones se verá con detalle en las prácticas.

Utilidades de detección de virus. Debido al gran incremento de virus que se produce a diario, así como a la facilidad de infectarse con los mismos debido a medios como internet, es de gran interés explicar a los alumnos en qué consisten y cómo funcionan los virus. Se les indicarán cuáles son las pautas a seguir para evitar infectarse en la medida de lo posible, así como cuáles son los principales programas existentes en la actualidad para salvaguardarse de los virus. Se completará este punto mostrando el manejo de una aplicación antivirus en particular, el VirusScan. También, se practicará y utilizarán sus distintas opciones en el laboratorio.

Tema 5: **Microsoft Word** [3]. En el trabajo diario con los ordenadores personales es de vital importancia el manejo de algún procesador de textos que permita al usuario la redacción y modificación de todo tipo de documentos. En la actualidad, el procesador de textos más extendido es el Microsoft Word, por lo que se ha

optado por este procesador como el candidato ideal a ser enseñado a los alumnos.

Se les introducirá por tanto en el manejo de esta aplicación partiendo de cero y hasta alcanzar un nivel medio que les permita desenvolverse sin problemas en la elaboración de documentos que incluyan desde texto –con cualquier formato- hasta tablas, gráficos e imágenes. De nuevo, se explicarán aspectos teóricos y se practicará con sus distintas opciones en el laboratorio.

Tema 6: **Microsoft Excel** [4]. Atendiendo a las necesidades que tendrá en un futuro el alumnado, no es posible descartar la enseñanza de alguna hoja de cálculo, puesto que no hay que olvidar que se trata de alumnos que estudian una Ingeniería, más concretamente Ingeniería Industrial.

Dentro de las diferentes hojas de cálculo entre las que es posible escoger no cabe duda que la elección de Microsoft Excel queda más que justificada por tratarse de la hoja de cálculo más extendida actualmente. Se mostrará al alumnado el funcionamiento de esta hoja de cálculo, si bien no se enseñará hasta un nivel excesivamente elevado, pues no hay que olvidar que se trata de alumnos no informáticos que no necesitan en principio profundizar excesivamente en determinados aspectos de esta herramienta. Al igual que en Word, se explican aspectos teóricos y se utiliza en las prácticas.

Tema 7: **Software** [1]. Dentro de este bloque se persigue el proporcionar a los alumnos una visión general de los diferentes tipos de aplicaciones que existen en la actualidad para dar solución a las distintas necesidades de los usuarios. Así pues, se mostrarán varios grupos de aplicaciones diferenciados por la temática que les concierne.

Es necesario destacar que será en este tema donde se enseñará a los alumnos las diferentes aplicaciones y usos del área de Internet. Esta área es de conocimiento indispensable para todos ellos en su futuro profesional, puesto que actualmente la práctica totalidad de las grandes empresas poseen acceso a Internet, siendo igualmente considerable el uso en las PYMES[1] españolas. De esta manera, se les hará especial incidencia en el tema de Internet, desde la navegación por la red hasta la utilización del correo electrónico.

4.3. Parte práctica

Una vez se ha proporcionado una visión de los diferentes temas que se tratarán en la parte teórica de la asignatura, se expondrán a continuación cuales serán las prácticas que se proponen con el fin de complementar, profundizar y asentar los conocimientos impartidos en las clases teóricas:

Práctica 1: **Entorno del Turbo C++** [2]. Esta práctica, que es la consecuencia natural del Tema 1 de teoría, tiene como objetivo que el alumno comience a manejar las principales opciones que proporciona el IDE del Turbo C++. Hay que señalar que el objetivo perseguido no es en ningún momento que el usuario aprenda a programar en forma alguna –esto es algo que escapa al temario de la presente asignatura- por lo que para poder probar determinadas opciones del entorno (como la compilación y el linkado) se proporcionará el código necesario.

Práctica 2: **Windows y su explorador de archivos** [1][2]. El tema 3 de teoría se verá complementado con esta práctica en la que se pretende familiarizar al alumnado con los sistemas operativos Windows. En primer lugar se explicará la parte práctica de Windows y del explorador. Al mismo tiempo y con posterioridad a la explicación, los alumnos practicarán con los nuevos conocimientos.

Una vez se haya terminado esta práctica los alumnos deberían ser capaces de desenvolverse sin problemas en Windows, así cómo poder realizar las operaciones comunes con archivos y carpetas (copiar, renombrar, crear, ...).

Práctica 3: **Manejo de Microsoft Word** [3]. Los conocimientos teóricos impartidos en el tema 5 serán ampliados de forma práctica en el laboratorio, con la explicación del profesor. A lo largo de la sesión se planteará a los alumnos la elaboración de diversos documentos en los que entren en contacto con las distintas opciones de Word.

[1] Según un estudio concluido en enero del 2002 por la consultora *DMR Consulting* en 3.500 compañías españolas con una plantilla máxima de 250 empleados el 62% de las mismas disponía de acceso a Internet

Así pues, una vez completada la práctica, los alumnos estarán en disposición de elaborar documentos con Microsoft Word, sin importar los formatos que necesiten, ni si hay que insertar elementos tales como tablas e imágenes.

Práctica 4: **Manejo de Microsoft Excel** [4]. Con esta práctica se buscan dos objetivos principales: por un lado que los alumnos aprendan a desenvolverse, al menos a un nivel básico, con el Microsoft Excel (visto en el tema 6 de teoría y ampliado de forma práctica en esta sesión), y por otro que sean capaces de trasvasar información entre Microsoft Excel y Microsoft Word.

De esta forma se plantearán, por un lado, una serie de ejercicios destinados específicamente a practicar con Excel, y por otro lado, otros ejercicios en los que el alumno deberá combinar la utilización de Excel y Word para poder alcanzar los objetivos pedidos.

Práctica 5: **Utilidades de uso general** [1][2]. Esta sesión esta pensada para que los alumnos puedan practicar el uso de las aplicaciones introducidas a lo largo del tema 4 de teoría y ampliados de forma práctica en el laboratorio. Uno de los objetivos principales que se perseguirá será que el alumno sea capaz de comprimir y descomprimir ficheros sin problemas, probando las diferentes opciones que WinZip ofrece. Como complemento se pretende también que los alumnos utilicen VirusScan para detectar posibles infecciones en grupos de ficheros entre los que se habrán colocado algunos que harán que el antivirus indique que están infectados.

Práctica 6: **Internet y correo electrónico** [1][2]. Dada la gran importancia actual de Internet, ya reseñada al exponer el tema 7 de teoría, es necesario plantear una práctica de cara a que los alumnos sean capaces de habituarse a la navegación por Internet y al manejo del correo electrónico. Entre los diferentes ejercicios que se les propondrán se incluirá el uso de buscadores y de diferentes portales para poder conseguir los objetivos planteados en los ejercicios.

Práctica 7: **PhotoShop** [5]. Si bien el tema central de esta práctica no está directamente relacionado con ningún tema de teoría en particular, el aprendizaje de una utilidad para la modificación de imágenes como es PhotoShop introducirá a los alumnos en las aplicaciones de retoque y manejo de imágenes en general, a la vez que servirá para darles mayor seguridad en el uso general del ordenador. Dentro de esta práctica se enlazará con otras aplicaciones, como puede ser Microsoft Word, de forma que se refuerce el conocimiento de las mismas en operaciones como la inserción de imágenes en documentos.

5. Conclusión

En este artículo se ha presentado la situación actual de ICO en la ETSII, se han estudiado distintos planes de estudio de universidades extranjeras y españolas y se ha realizado una propuesta de cambio. Se han descrito los objetivos que se consideran más adecuados y se han desarrollado los contenidos que mejor se ajustan a dichos objetivos.

Referencias

[1] García, J.R., Juan, M.C., *Informática y ordenadores personales para no especialistas. Servicio de Publicaciones de la UPV. Ref. 97-580, 1997*

[2] Juan, M.C., García, J.R., *Introducción a los Computadores. Material Complementario*, Servicio de Publicaciones de la UPV. Ref. 2000-248, 2000

[3] García, J., Rodríguez, J.I., Brazales, A., Funes, P., Fernández, J., Carrasco, E., Aprenda Microsoft Word 97 como si estuviera en primero, ESII de la Universidad de Navarra, http://www1.ceit.es/asignaturas/informat1/AyudaInf/AprendaInf/Word97/Word97.pdf

[4] García, J., Rodríguez, J.I., Brazales, A., Funes, P., Fernández, J., Aprenda Microsoft Excel 97 como si estuviera en primero, ESII de la Universidad de Navarra, http://www1.ceit.es/asignaturas/informat1/AyudaInf/AprendaInf/Excel97/Excel97.pdf

[5] Quirós, E., Quirós, S., *Photoshop 5.5 práctico: guía de aprendizaje*, Ed. McGraw-Hill/Interamericana deEspaña, D.L., 2000

Ingeniería del Software

Ingeniería del Software aplicada a un Laboratorio de Introducción a la Programación en Ada

María José García, Pedro Lara, Luis Fernández
Dept. de Programación e Ingeniería del Software
Universidad Europea CEES
28670 Villaviciosa de Odón
e-mail:{pepa, pedro,lufern}@dpris.esi.uem.es

Alberto Díaz
Ing. Técnica en Informática de Sistemas
Centro de Estrudios Superiores Felipe II- UCM
28300 Aranjuez
e-mail: adiaz@cesfelipesegundo.com

Resumen

La metodología aplicada en el primer Laboratorio de Programación al que se enfrentan los alumnos de una carrera de Ingeniería (o Ingeniería Técnica) en Informática es determinante para establecer en ellos unos buenos hábitos ante la programación. En esta ponencia se presenta la solución adoptada en la Escuela de Informática de la Universidad Europea CEES. El enfoque elegido pretende aunar la enseñanza de los fundamentos de la programación y la introducción de conceptos básicos de ingeniería del software para el desarrollo de aplicaciones. También se exponen los criterios de evaluación aplicados, así como la forma de coordinar y gestionar los grupos, a fin de utilizar en todos ellos unos criterios homogéneos.

1. Introducción

Laboratorio de Programación I es una asignatura obligatoria primer curso para las titulaciones de Ingeniería Informática, Ingeniería Técnica en Informática de Gestión e Ingeniería Técnica en Informática de Sistemas. Esta asignatura cuenta con una dedicación de 6 créditos prácticos durante el segundo cuatrimestre.

Contando desde la implantación del primer curso de los anteriores planes de estudios, el Laboratorio de Programación I lleva impartiéndose ya siete años en la Escuela Superior de Informática. Cada curso han ido surgiendo diversos problemas debidos a la propia idiosincrasia de la asignatura y a su carácter eminentemente práctico.

Por un lado, se ha decidido utilizar una metodología que está basada en los fundamentos de la Ingeniería del Software, de tal forma que se orienta al alumno hacia una forma completa de solucionar los problemas, en lugar de centrarse sólo en la tarea de la implementación en un determinado lenguaje de programación. Así se enlazaría con las posteriores asignatruas

Por otra parte, puesto que ésta es la primera asignatura en la que los alumnos deben construir una solución software completa a un problema dado, se considera fundamental el trabajo individual. Esto implica la necesidad de distribuir a los alumnos en grupos de manera que cada alumno pueda, durante las horas de clase, trabajar con un ordenador. Además los grupos reducidos permiten un mejor seguimiento de la evolución de los alumnos. En nuestro caso los laboratorios están dotados con 30 equipos, por lo que la medida que se suele adoptar es la distribución en grupos de entre 20 y 25 alumnos, cada uno de ellos con un horario y un profesor distinto.

El problema entonces es doble: hay que decidir cómo adoptar el enfoque de la ingeniería del software con alumnos que se enfrentan por primera vez con la implementacion de soluciones en el ordenador y además coordinar a todos los profesores de manera que se sigan unas pautas de actuación unificadas (a fin de ser lo más ecuánimes posibles y de evitar diferencias entre los grupos).

Este curso 2001-2002 el problema se ha incrementado puesto que son ya trece los grupos que se han tenido que formar.

Además es importante guardar una absoluta coherencia entre los contenidos y criterios de esta asignatura y la de Introducción a la Programación, tambien de primer curso y que tiene carácter troncal y anual, prevista como implantación en el plan de estudios como implantación de la troncalidad de Metodología y Tecnología de la

Programación. Su objetivo es proporcionar los contenidos más teóricos sobre programación que se aplicarán después en el Laboratorio de Programación I.

Otro factor que influye en la necesidad de establecer de una manera clara los parámetros de la asignatura es la heterogeneidad de los profesores. Muchos de ellos es la primera vez que imparten esta asignatura, por lo que es importante poder proporcionarles a principio de curso unas guías de actuación:

- Es necesario realizar una planificación previa de la asignatura, incluyendo enunciados, diseños, conjunto de pruebas y fechas de entrega de cada una de las prácticas.
- También es fundamental establecer unos criterios de evaluación claros y únicos.
- Para conseguir evitar incoherencias entre los diversos grupos y también para detectar las posibles copias de prácticas entre alumnos de distintos profesores deben existir mecanismos de coordinación y comunicación entre los profesores de la asignatura.
- Del mismo modo se tiene que facilitar el flujo de información alumno ↔ profesor.

El enfoque adoptado para la enseñanza de la programación asume la conveniencia de enseñar primero los principios básicos de la programación procedimental y estructurada (frente a quienes prefieren comenzar directamente con la orientación a objetos). La elección del lenguaje Ada no es casual ya que se pensó en un lenguaje que permita programación procedimental relativamente natural. Pero que, a la vez (frente a lenguajes como Pascal) permita una transición interesante hacia la enseñanza de tipos abstractos de datos (en la asignatura de segundo curso de Estructuras de Datos y de la Información de nuestros planes de estudios) o hacia la orientación a objetos (para asignaturas como Programación Orientada a Objetos).

Por otra parte, la introducción de los conceptos tradicionales de Ingeniería del Software en una asignatura de primer curso permite a los alumnos tener una base sobre la que cimentar posteriormente los conocimientos que adquirirán en las asignaturas plenamente dedicadas a esta materia que se imparten en cursos superiores (dos asignaturas de carácter troncal y una obligatoria que suponen un total de 24 créditos para Ingeniería del Software en la carrera de Ingeniería Informática, 12 créditos (6 troncales y 6 obligatorios) en Ingeniería Técnica en Informática de Gestión y otros 6 créditos obligatorios en Ingeniería Técnica en Informática de Sistemas).

2. Herramientas de trabajo

2.1. Herramientas para los alumnos

Como herramientas de trabajo los alumnos disponen en los laboratorios de máquinas PC pentium 300 MHz. Las prácticas se realizan en el lenguaje de programación ADA, por lo tanto en cada computadora está instalado un compilador (GNAT: GNU Ada Translator), un depurador (GDB: GNU Debugger), y los manuales de referencia del lenguaje (incluyendo el estándar ISO correspondiente [3] y libros típicos como [1] y [5]). Se trabaja también con el editor de desarrollo pcGRASP (Graphical Representations of Algorithms, Structures, and Proceses) y las aplicaciones ofimáticas WORD 98 y Powerpoint para la documentación (memoria) de las prácticas. También disponen de la herramienta Microsoft VISIO para dibujar los diagramas de estructuras.

Para la elección del compilador, depurador y editor que se utilizan en la asignatura se ha tenido en cuenta la necesidad de que los alumnos puedan seguir trabajando en las prácticas fuera del horario lectivo. Por lo tanto se han seleccionado herramientas potentes pero de libre distribución que se ponen a disposición de los alumnos a través de FTP anónimo. También a través del FTP anónimo y la página de la asignatura se entrega a los alumnos, en el momento oportuno, la documentación para la realización de cada práctica.

2.2. Herramientas para los profesores

2.2.1. Herramientas de ayuda a la corrección

A lo largo del proceso de evaluación de los programas entregados por los estudiantes han de realizarse toda una serie de actividades que en gran parte se convierten en una labor tediosa y repetitiva. Entre estas actividades destacan fundamentalmente las siguientes: validación del

ajuste del código entregado al diseño realizado en las fases previas, detección de errores graves en el código y búsqueda de plagios o copias más o menos evidentes.

En un intento de reducir el tiempo dedicado a estas labores surgió la necesidad de llevar a cabo un proyecto de desarrollo que facilitase, automatizase e incluso eliminase en algunos casos cualquier acción manual por parte del docente en relación con los procesos citados anteriormente, particularizado para algunas características de Ada.

Actualmente, dicho proyecto ha dado como fruto un primer entorno piloto que cubre parte de la funcionalidad prevista detectando errores en implementaciones no coincidentes con los diseños entregados (genera un esquema del diseño arquitectónico que se ha implementado incluyendo la modelización de los bucles y número de llamadas a los módulos), encontrando y mostrando errores graves de codificación (utilización inadecuada de las variables) y analizando los datos obtenidos de las comparaciones cruzadas de varios ficheros fuente en busca de indicadores de plagio ([2] y [4]). En particular muestra estadísticas sobre líneas iguales/modificadas/añadidas/borradas, y sobre identificadores iguales/modificados).

Además este entorno permite la compilación y ejecución de cada uno de los ficheros fuente utilizando la entrada/salida estándar o redirigiendo estas a ficheros para unificar las pruebas realizadas por el profesor y comprobar el ajuste con las especificaciones.

Como herramienta de apoyo en la corrección de las prácticas se utiliza además un analizador de código desarrollado como proyecto fin de carrera por un alumno de la Escuela. Este analizador permite realizar tanto análisis estático como dinámico de un determinado código, mostrando tanto grafos de llamadas como de flujo, así como el valor de una serie de métricas definidas por el profesor.

2.2.2. Herramientas de coordinación

Puesto que el número de profesores que imparten la asignatura es bastante elevado, se ha creado la figura de "coordinadores de laboratorio", encargados de tomar ciertas decisiones básicas como son la elección de enunciados de las prácticas de las fechas de entrega de las mismas, y el establecimiento de unas normas comunes y criterios de evaluación uniformes. No obstante, se utiliza una lista de distribución en la que están incluidos todos los profesores de laboratorio así como los coordinadores de la asignatura. De esta manera se facilita la comunicación entre todos, pudiéndose comentar las posibles modificaciones o aclaraciones que, sobre cada práctica, realice cada profesor con sus alumnos. Esto permite una máxima coherencia sin necesidad de costosas reuniones, poco factibles debido a la diversidad de horarios de los profesores.

2.3. Herramientas para la comunicación alumno ↔ profesor

2.3.1. Página Web y repositorio de información de la Asignatura

Como para la mayoría de las asignaturas impartidas en la Escuela de Informática de la Universidad Europea CEES, existe una página web en la que de manera unificada todos los alumnos pueden recoger información general. En este caso en ella recogemos los objetivos generales de la asignatura, el sistema de evaluación, información sobre grupos, profesores y aulas y normas generales para la realización de las prácticas.

Además se proporciona un enlace al repositorio de documentos de la asignatura en el que, además de una plantilla para la entrega de la memoria, se incluye a medida que avanza el curso información específica sobre cada práctica: objetivo, enunciado y fechas de entrega de cada fase. En el momento adecuado, es también allí donde se pone a disposición de los alumnos el diseño arquitectónico en el que tienen que basar su implementación.

2.3.2. Automatización de la entrega de prácticas

Para unificar el modo de entrega de las prácticas eliminando en lo posible el trasiego de papeles y disquetes se ha construido una aplicación a la que se accede a partir de la página web de la asignatura. Mediante un pequeño formulario que el alumno debe rellenar, y que le permite adjuntar los ficheros necesarios, las prácticas son enviadas

por correo electrónico a cada profesor, guardándose registro del momento exacto de entrega.

3. Metodología para la realización de las prácticas

Uno de los objetivos de la asignatura de laboratorio de programación I, aparte de la enseñanza de los fundamentos de la programación, consiste en inculcar en los alumnos el uso de un enfoque de ingeniería del software para el desarrollo de aplicaciones. Durante la asignatura de Introducción a la Programación ya han adquirido los conocimientos necesarios sobre éstas técnicas, en particular, para la fase de diseño se les ha enseñado la metodología de los refinamientos sucesivos, y también las diferentes estrategias algorítmicas básicas [6]. Es también en esa asignatura donde se les presenta la sintaxis del lenguaje ADA, que es en el que se implementarán las soluciones a los problemas propuestos en el laboratorio.

Durante la primera semana de laboratorio se pone en contacto al alumno con el entorno que va a encontrar en la asignatura:

En la primera clase se explica a los alumnos la dinámica del laboratorio, número y tipo de prácticas, metodología y criterios de evaluación.

En la segunda clase se realiza la primera toma de contacto con el laboratorio. En esta clase se le entrega a cada alumno un enunciado y un esbozo de la implementación de una práctica sencilla parecida a algún ejercicio resuelto previamente en la asignatura de Introducción a la Programación. Se trata de que el alumno se familiarice con el entorno de programación que va a utilizar a lo largo del curso, completando la codificación que se le entrega; adicionalmente debe depurar el programa ejemplo, detectando y eliminando diferentes errores introducidos en el código: errores de compilación, mal funcionamiento y utilización inadecuada de variables (errores de ámbito).

A partir de la segunda semana empieza la realización del resto de las prácticas de las que consta la asignatura. Cada una de ellas está centrada en la utilización de alguno de los elementos de programación introducidos en la asignatura de teoría. La primera se dedica a la abstracción procedimental, la segunda a la selección, la repetición y los archivos de texto, la tercera al uso de tipos definidos por el usuario sencillos (enumerados, subrangos y arrays simples), la cuarta a la utilización de arrays y registros, y la última, a archivos binarios y memoria dinámica.

La realización de cada una de las prácticas consistirá en desarrollar cada una de las fases del ciclo de vida clásico: análisis, diseño, implementación y prueba, siguiendo la metodología que se describe a continuación.

3.1. Análisis

La *fase de análisis* se realizará el primer día de clase de cada práctica (normalmente lunes), cada grupo con su profesor. El enunciado se pone a disposición de los alumnos el viernes anterior a través del ftp anónimo, de manera que han tenido tiempo de revisarlo previamente. Se repasa el enunciado, se discuten posibles ambigüedades y se llega a unas especificaciones claras y concisas sobre lo que hay que hacer.

3.2. Diseño

La *fase de diseño* consiste en la elaboración del diseño de datos, arquitectónico y procedimental y de la colección de pruebas a realizar. Los diseños se realizan siguiendo la filosofía de la programación estructurada y el método de los refinamientos sucesivos.

- Diseño de datos: deberá seguir la notación EBNF.
- Diseño arquitectónico: se presentará un diagrama de estructuras completo según la notación clásica [7].
- Diseño procedimental: tendrá que reflejar, al menos, el pseudocódigo del programa principal y de los procedimientos de cierta complejidad.

Una vez que los alumnos hayan entregado la documentación asociada a esta fase, el profesor entregará un modelo de diseño de datos y arquitectónico aproximado (sin especificar exactamente el flujo de datos entre los módulos) junto con el interfaz (formato de presentación de los datos de entrada y salida). Este modelo es el

que deberán seguir los alumnos en las siguientes fases. De nuevo, para evitar papeles innecesarios, el modelo de diseño se pone a disposición de los alumnos en el ftp anónimo de la asignatura.

3.3. Implementación

La *fase de codificación* será la traducción del modelo de la fase de diseño entregado por el profesor al lenguaje Ada, corrigiendo errores con el compilador, y ajustándose al interfaz indicado.

La razón para que codifiquen a partir del diseño del profesor y no del suyo propio es evitar que realicen la implementación y después ajusten el diseño a esa codificación (se ha observado que esto es lo que hacían algunos alumnos en años anteriores, de modo que se ponían a codificar sin detenerse a planificar primero un diseño adecuado)

Para facilitar a los profesores la labor de corrección, de modo que les sea más sencillo identificar los diferentes elementos del código, los alumnos seguirán la siguiente normativa de codificación:

- Al inicio del código, en las primeras líneas del fichero adb, se incluirán los datos del autor (Nombre, nº de expediente, grupo, fecha de inicio de la práctica)
- Todos los módulos deberán estar comentados (¿qué hacen? ¿cómo?), sobre todo aquellas cosas especialmente complicadas.
- Notación para los identificadores:
    1. Identificadores compuestos: separados por guión_bajo
    2. Constantes: todo el identificador en mayúsculas
    3. Variables: empiezan por minúsculas, resto en minúsculas
    4. Funciones o procedimientos: empiezan por mayúsculas, resto en minúsculas

3.4. Pruebas

La *fase de prueba* consistirá en primer lugar en el diseño de un conjunto unificado de pruebas, determinando para cada caso de entrada cuál debería ser el resultado a obtener. Posteriormente se deberá ejecutar el programa en todas aquellas situaciones detectadas como significativas en el diseño de las pruebas, asegurando, a través de ellas, que el programa funciona correctamente y se obtienen los resultados esperados.

Utilizando los medios que proporciona la herramienta desarrollada en esta universidad para capturar la entrada y la salida mediante ficheros de un programa implementado en ADA, será posible probar con una sola ejecución el funcionamiento de las prácticas.

El alumno entregará el resultado de las pruebas diseñadas, y de cualquier otra prueba que se haya realizado (capturadas en ficheros). Deben incluirse resultados finales tanto positivos como negativos. Para los casos en que los resultados no son los esperados, se indicará la posible causa y posibles soluciones (justificar fallos). Los alumnos serán penalizados en la evaluación si entregan un juego de pruebas incompleto, que no detecte errores del código.

3.5. Documentación

Cualquier aplicación software debe estar debidamente documentada. Es esto lo que se les pretende inculcar nuestros futuros programadores. Los alumnos de esta asignatura deben entregar una memoria final (siguiendo un formato específico) que incluya la documentación específica asociada a cada una de las fases anteriores (según el modelo que se les proporciona) y además un manual de usuario contemplando aspectos como a quién va destinado el programa, cómo se ejecuta, qué errores devuelve y cómo se interpretan, requisitos de ejecución, incompatibilidades etc.)

4. Evaluación de las prácticas

En cuanto a la metodología de evaluación, es necesario volver a hacer hincapié en la importancia de la homogeneidad entre los grupos. No hay que olvidar que hay alumnos que están distribuidos en diferentes grupos pero están matriculados en la misma asignatura, y comparten el resto de las clases.

Se ha intentado huir de una tabulación de distintos fallos con una serie de descuentos asociados porque es imposible corregir una práctica basándose en si un fallo descuenta 2 puntos y otro 1. En su lugar se presentan una serie de criterios para guiar en la puntuación de cada

práctica. Pero como siempre ocurre en programación es muy difícil encontrar una forma sistemática de puntuar programas: se necesita tener una visión global del programa y una serie de criterios básicos (y quizás algo de experiencia) para poder asignar una nota.

4.1. Evaluación continua

En esta Universidad se sigue el modelo de evaluación continua. Esto implica la valoración del trabajo del alumno a lo largo de toda la asignatura, pero también la obligación por parte del alumno de una cierta regularidad de resultados.

Las prácticas van incrementando su complejidad a lo largo de la asignatura, siendo la última práctica la más completa.

Al final de la asignatura se realizará un examen que no tendrá nota (si bien es una observación más a disposición del profesor en su evaluación) puesto que sólo es una comprobación de que las prácticas las ha realizado cada alumno.

La calificación de la asignatura vendrá dada por una media ponderada de las notas en cada una de las prácticas (a medida que se incrementa la complejidad de las prácticas, aumenta también su peso en la nota final).

Para aprobar la asignatura se pueden presentar las siguientes posibilidades:

- Alumnos que hayan ido aprobando todas las prácticas: sólo tendrán que presentarse al examen final para comprobar que son autores de la última práctica.
- Alumnos con alguna práctica suspensa antes de la última:
  1. Si no alcanzaron una nota mínima (que se ha establecido en 3 sobre un total de 10) en las prácticas más importantes, pierden la posibilidad de aprobar en junio.
  2. En otro caso: repetirán para junio las prácticas que no hayan llegado a la nota mínima o las no entregadas, presentándose al examen final para comprobar la autoría. En dicho examen se realizarán preguntas sobre ellas (razonamientos, cambios de codificación o de diseño respecto a la anterior versión...).
- En todo caso en la última práctica se debe obtener al menos un 5 para poder aprobar la asignatura.
- En la convocatoria de septiembre se deben entregar tres prácticas (dos de ellas son repetidas del curso) aunque sólo deben repetirse en caso de tenerlas suspensas o no presentadas). El examen de septiembre será personalizado, porque se trata de comprobar la autoría de unas prácticas que se han realizado fuera del periodo de clases y por tanto sin el seguimiento directo de los profesores.

4.2. Criterios de corrección

Como siempre, la consigna es conseguir la mayor homogeneidad en cuanto a los criterios que se aplican para la corrección de cada uno de los aspectos de cada práctica.

Dado que se realizan dos entregas por práctica, debemos valorar tanto la fase de diseño como la de codificación. La proporción será 1/3 la nota de diseño, 1/3 la nota de codificación, 1/3 la nota de pruebas y documentación.

4.2.1. Fase de diseño

- El diseño de datos deberá seguir la filosofía top-down, dejando perfectamente definidas las estructuras de datos adecuadas, incluso aunque aún no sean capaces de codificarlas.
- Reglas que se deben valorar en el diseño arquitectónico:
  1. Diseño estructurado
  2. Seguir la filosofía EPS (entrada-proceso-salida)
  3. Mínimo acoplamiento
  4. Máxima coexión
- El diseño procedimental se debe corresponder con el diseño de datos y arquitectónico entregados, tiene que especificar claramente el interfaz de cada módulo diseñado.
- El diseño de pruebas: Se comparará con el conjunto oficial de pruebas.

4.2.2. Fase de implementación

Existen errores que obviamente conducen al suspenso inmediato:

- Práctica que no compila.
- Implementación diferente del diseño (del profesor)
- No se ajusta a las especificaciones (incluso a las de interfaz)

Hay también errores que conducen al suspenso:

- Mala utilización del ámbito de las variables (utilización de variables globales sin pasarlas por parámetro).
- Mala utilización del ámbito de los módulos (invocación a módulos hermanos).
- Mala estructuración, utilización improcedente de instrucciones de salida (iteraciones de las que se puede salir mediante un exit, procedimientos de los que se sale con un return...)
- Resultados que produzcan error en tiempo de ejecución o que sean contrarios a las especificaciones
- Resultados erróneos (no exactos)

Hay también cuestiones de estilo que pueden ayudar a dilucidar la nota final obtenida en una práctica:

- Hay costumbres que se deben evitar
  - Inclusión innecesaria de sentencias NULL
  - Utilización de bucles definidos en lugar de bucles indefinidos, y viceversa
  - Mala utilización de instrucciones de selección, anidamientos inadecuados, utilización de varias condicionales simples en lugar de una condicional múltiple
  - Funciones que realizan operaciones de entrada/salida de datos
  - Procedimientos que, en realidad, implantan lo que debe ser una función
  - Varias sentencias de retorno dentro de una misma función
- Y algunas que es necesario inculcar
  - Inicialización de variables
  - Uso de identificadores significativos
  - Módulos con una única funcionalidad
  - Comentarios adecuados en el código

4.2.3. Fase de pruebas y documentación

Para facilitar una corrección más cómoda y uniforme, se diseñará un conjunto de pruebas concebido como un fichero de casos de valores de entradas. La idea es que este fichero, que se repartirá a todos los profesores, será la prueba "oficial" que se aplicara para decidir si el programa "funciona" o no.

ADA proporciona una manera sencilla de redireccionar la entrada/salida estándar a ficheros, permitiendo así que se generalicen la utilización de ficheros de prueba. Además, la herramienta de ayuda a la corrección que se ha desarrollado permite automatizar este proceso.

A efectos de evaluación, hay que observar si el programa funciona en todos los casos de prueba (se supone que esto es lo que deberían lograr siempre los alumnos), si falla en algún caso puntual o si ni siquiera permite ejecutar las pruebas (no se respeta el interfaz) o está plagado de fallos.

5. Conclusiones

Se ha presentado una metodología para la realización de prácticas en el laboratorio de programación de primer curso que trata de definir una dinámica que permita homogeneizar los distintos grupos de alumnos de dicha asignatura.

Esta metodología está basada en los conceptos tradicionales de ingeniería del software en el desarrollo de programas, concretamente en las fases del ciclo de vida clásico: análisis, diseño, implementación y prueba.

El hecho de obligar a realizar un diseño previo sin utilizar el ordenador (mejor dicho, utilizándolo como herramienta de apoyo para "mostrar el diseño" pero sin escribir ni una línea de código) potencia la elaboración de una planificación previa en lugar de ponerse a implementar directamente.

Por otro lado, que tengan que implementar a partir de un diseño de otra persona también les acostumbra a adaptarse a la participación de otras personas en el desarrollo de programas.

Se consigue facilitar la corrección de las prácticas a los profesores mediante un conjunto de pruebas oficial, ajustado al interfaz que se les entrega a los alumnos, y que mediante la

redirección de la entrada/salida a fichero permite probar la práctica con una sola ejecución.

Los criterios de evaluación se intentan definir de la manera más clara y concisa posible para evitar la falta de coherencia entre distintos profesores de la misma asignatura, sobre todo entre aquellos que imparten grupos que coinciden juntos en el resto de asignaturas.

Estos criterios están basados en conceptos introducidos en la asignatura de Introducción a la Programación y tratan de fomentar el desarrollo de programas estructurados y bien diseñados.

La metodología adoptada en la asignatura no depende del uso de Ada como lenguaje sino que es exportable a otras asignaturas similares que empleen un lenguaje procedimental.

Referencias

[1] J.G.P. BARNES. *Programming in Ada 95. 2nd Edition* Addison-Wesley, 1998.

[2] P. CLOUGH, *Plagiarism in natural and programming languages: an overview of current tools and technologies*, CS-00-05, Internal Report, Department of Computer Science, The University of Sheffield, junio, 2000.

[3] ISO, *ISO/IEC 8652:1995, Information technology : Programming languages. Ada*, ISO, 1995.

[4] S. A. MENGEL, J. V. ULANS, *A Case Study of the Analysis of Novice Student Programs*, Proceedings of the 12th Conference on Software Engineering Education and Training, 1998.

[5] J. SKANSHOLM. *Ada. From the Beginning. 2*, Addison-Wesley, 1994.

[6] A.B. TUCKER et al., *Fundamentos de informática: Lógica, resolución de problemas, programas y computadoras*, Madrid, McGraw-Hill / Interamericana, 1994.

[7] E. YOURDON, L. CONSTANTINE, *Structured Design*, Prentice-Hall, 1979.

Una herramienta para la enseñanza de patrones en Ingeniería del Software

Macario Polo, Juan Ángel Gómez, Mario Piattini y Francisco Ruiz
Escuela Superior de Informática – Universidad de Castilla-La Mancha
Paseo de la Universidad, 4
13071-Ciudad Real
macario.polo@uclm.es

Resumen.
Se presenta una herramienta que genera una aplicación totalmente ejecutable a partir de un diagrama de clases dibujado con Rational Rose. La herramienta considera el diagrama como la estructura de la capa de Dominio de un sistema de tres capas. A partir de esta idea, la herramienta genera las capas adyacentes (Presentación y Almacenamiento), dándole ciertas funcionalidades suministradas por un conjunto de patrones de diseño.

1. Introducción.

Los patrones de diseño son parte de los contenidos de la asignatura "Ingeniería del Software II", del 5º curso de la Ingeniería en Informática de la Universidad de Castilla-La Mancha. La asignatura consta de 9 créditos, 6 de los cuales corresponden a teoría y 3 a clases de laboratorio. Entre las herramientas que los alumnos utilizan en los laboratorios se encuentra Rational Rose, que es utilizada como base para la construcción de una aplicación ejecutable que, independientemente de que funcione correctamente o no, debe poseer un buen diseño, arquitectura robusta, etc., aspectos éstos que son los más valorados a la hora de determinar la calificación. La idea de este trabajo práctico es que los alumnos apliquen a un caso concreto los contenidos teóricos de la asignatura, de los que los patrones de diseño son una buena parte.

Durante las clases teóricas, se pone especial énfasis en la necesidad de dotar a las aplicaciones de una arquitectura robusta, y suelen utilizarse como ejemplos aplicaciones con tres capas (normalmente Presentación, Dominio y Almacenamiento). Una aplicación de tres capas bien elegida posee la versatilidad suficiente como para incluirle los elementos necesarios para presentar ejemplos de prácticamente todos los patrones de diseño que se presentan durante la asignatura. Así, por ejemplo:

1. La propia utilización de una arquitectura multicapa es ya un buen patrón, que dota a las aplicaciones de gran escalabilidad, que permite la reutilización, etc. Además, las clases y paquetes de las aplicaciones de tres capas bien construidas poseen valores adecuados de cohesión y acoplamiento, cuyos equilibrados valores son probablemente dos de los más básicos principios de un diseñador de software.
2. El hecho de que los objetos de la capa de Presentación necesiten "refrescarse" para mostrar los cambios de estado experimentados por los objetos de la capa de Dominio, permite la incorporación a la aplicación de clases que constituyen implementaciones del patrón Observer.
3. La utilización de una base de datos que almacena (en forma de registros) las instancias persistentes procedentes de clases de la capa de Dominio, permite:
   - 3.1 La utilización de alguno de los muchos patrones existentes para hacer corresponder clases y tablas.
   - 3.2 La utilización de patrones para decidir a qué clases deben asignarse las responsabilidades relacionadas con la persistencia de los objetos.
   - 3.3 La utilización de patrones para transformar el diagrama de clases de la capa de Dominio (o parte de él) al esquema físico de la base de datos.

3.4 La utilización de algún patrón que centralice el acceso de los objetos de la capa de Dominio a la base de datos (un Broker o Agente).

Desde luego, resulta imposible construir en el laboratorio todas las posibles variantes de la aplicación utilizando todos los posibles patrones de diseño, incluso aunque ésta sea sencilla. También es difícil conseguir que los estudiantes implementen, aunque sea fuera del laboratorio, todas las posibles variantes. Pero el caso es que, observando las reglas de transformación y diseño que, más o menos, rigen el mecanismo de utilización de los patrones comentados en los puntos anteriores, se observa que el proceso de aplicación de dichos patrones es –para una persona- un proceso muy automático que puede, además, ser automatizado por una herramienta.

En la Escuela Superior de Informática de Ciudad Real hemos construido una herramienta que, a partir de un diagrama de clases dibujado con Rational Rose, permite al usuario generar código directamente ejecutable. El código se genera de acuerdo a un conjunto de patrones, que el alumno ha podido elegir de un conjunto de patrones disponibles. La generación de código tarda sólo unos pocos segundos, lo que permite al alumno generar en muy poco tiempo lo que habitualmente tardaría horas o días, y sin ningún error. Esto permite aprovechar más aún las horas de laboratorio, ya que se libera tiempo que puede dedicarse a la aplicación práctica de otros contenidos presentados en las horas de teoría.

En este artículo presentamos algunos detalles del diseño e implementación de esta herramienta, así como su forma de uso y sus posibilidades.

2. Aspecto de la herramienta.

La Figura 1 muestra el aspecto de la ventana principal de nuestra herramienta. En ella, el alumno selecciona un fichero que contenga un modelo de Rational Rose. El conjunto de clases contenido en dicho modelo se muestra en la lista de la izquierda, debiendo el usuario seleccionar las clases para las que se desea generar código.

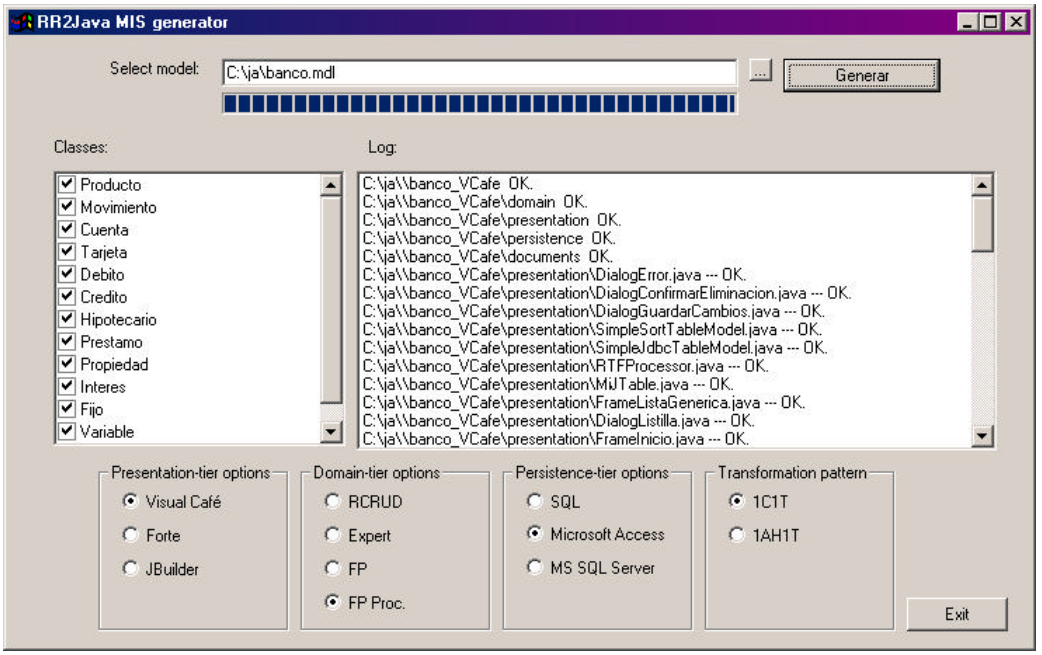


Figura 1. Pantalla principal de la herramienta.

El alumno debe saber que la herramienta ubicará las clases seleccionadas en la capa de Dominio de la aplicación generada, por lo que no es conveniente seleccionar clases que representen ventanas, etc. Para cada una de las tres capas para las que se va a generar código pueden elegirse diversas opciones (situadas en la parte inferior de la ventana):

1. Para la capa de Presentación puede elegirse el entorno de desarrollo para el que se generará código.
   - 1.1 Aunque se observan varios entornos de lenguaje Java, ocurre que los Frames construidos con, por ejemplo, Visual Café no son editables con Forte o JBuilder, razón por la que los posibles entornos se colocan como opciones excluyentes.
   - 1.2 Se genera una pantalla "tipo ficha" por cada clase seleccionada. Además, se hace que cada clase de la capa de Presentación conozca a su instancia correspondiente de la capa de Dominio (por ejemplo: una pantalla que muestra los datos de un empleado conoce a su objeto Empleado correspondiente).
   - 1.3 Se generan un conjunto de ventanas adicionales y "constantes", cuya implementación no depende o depende muy poco de la aplicación que se está generando. Ejemplos de estas ventanas son un diálogo para mostrar mensajes de error, una ventana que muestra listas de registros, una ventana que pide confirmación antes de borrar o modificar, etc.
   - 1.4 Todas las pantallas generadas poseen un conjunto de operaciones que permiten invocar operaciones de persistencia sobre los objetos de la capa de Dominio (las pantallas "tipo ficha", por ejemplo, tienen un botón "Guardar", que guarda el objeto actualmente mostrado en la base de datos, mediante una llamada al método *insert* o *update* –depende del caso- del correspondiente objeto de la capa de Dominio).
2. La capa de Dominio se genera mediante una "traducción directa" de las clases seleccionadas al lenguaje de programación considerado. Además, para cada una de estas clases se genera el siguiente conjunto de métodos:
   - 2.1 Un constructor sin parámetros, que da a los campos de la clase valores por defecto (por ejemplo, el valor cero a los *long*).
   - 2.2 Un constructor con parámetros que permite materializar objetos (esto es, construir instancias de clases a partir de los registros almacenados en la base de datos). Los valores de los parámetros pasados a este constructor se corresponden con los valores de las columnas que forman la clave principal del registro que queremos materializar.
   - 2.3 Métodos *insert*, *delete*, *update* y *delete*, que permitirán actualizar el objeto en la base de datos. Esto constituye una implementación del patrón CRUD (operaciones para hacer Create, Read, Update y Delete), expuesto por Yoder [4].
3. Para la capa de Persistencia se genera siempre un Agente [1] cuya implementación es siempre constante, y que ejecuta instrucciones SQL sobre la base de datos. Existen cuatro formas de asignar y generar las responsabilidades de persistencia:
   - 3.1 Mediante el patrón Experto, cada clase persistente define e implementa sus operaciones de persistencia. Es decir, en ellas reside completamente el código de las operaciones *insert, delete,* etc. Esta alternativa tiene la desventaja de que hace a las clases de la capa de Dominio completamente dependientes del gestor de base de datos utilizado.
   - 3.2 Para desacoplar a las clases de la capa de Dominio del sistema gestor de base de datos, las operaciones de persistencia pueden delegarse a "fabricaciones puras" [2]. De este modo, si se cambia de gestor de base de datos, sólo se necesitará modificar las clases asociadas (las fabricaciones puras, que residen en la capa de Almacenamiento), y no las de Dominio, que son mucho más complejas puesto que son las que verdaderamente implementan la solución del problema propuesto (que habitualmente será mucho más que gestionar persistencia). El sencillo ejemplo con que ilustramos en clase esta dependencia del gestor de base de datos es que, si se usa Access, deben usarse las palabras *true* o *false* para guardar valores en columnas boolea-

nas, mientras que se utilizan 1 y 0 con SQL Server. Existen, de todos modos, varias formas de implementar las operaciones de persistencia en las fabricaciones puras, como hacerlas estáticas o no (Figura 2). Nuestra herramienta siempre genera como estáticas estas operaciones. En cualquier caso, y aún habiendo decidido que las operaciones de persistencia en las fabricaciones puras sean estáticas, nuestra herramienta ofrece dos posibilidades: que la fabricación pura genere y directamente ejecute la instrucción SQL (opción "FP" en la ventana de la Figura 1), o mediante generación de sentencias preparadas (opción "FPProc" de la Figura 1).

3.3 Las operaciones de persistencia pueden también ser asignadas mediante el patrón RCRUD [3]. RCRUD (Reflective Create, Read, Update & Delete) es una clase, totalmente reutilizable, que genera mediante introspección (*Reflection)* las operaciones de persistencia de cualquier clase. El único requisito para utilizar RCRUD es que las clases persistentes sean especializaciones de RCRUD.

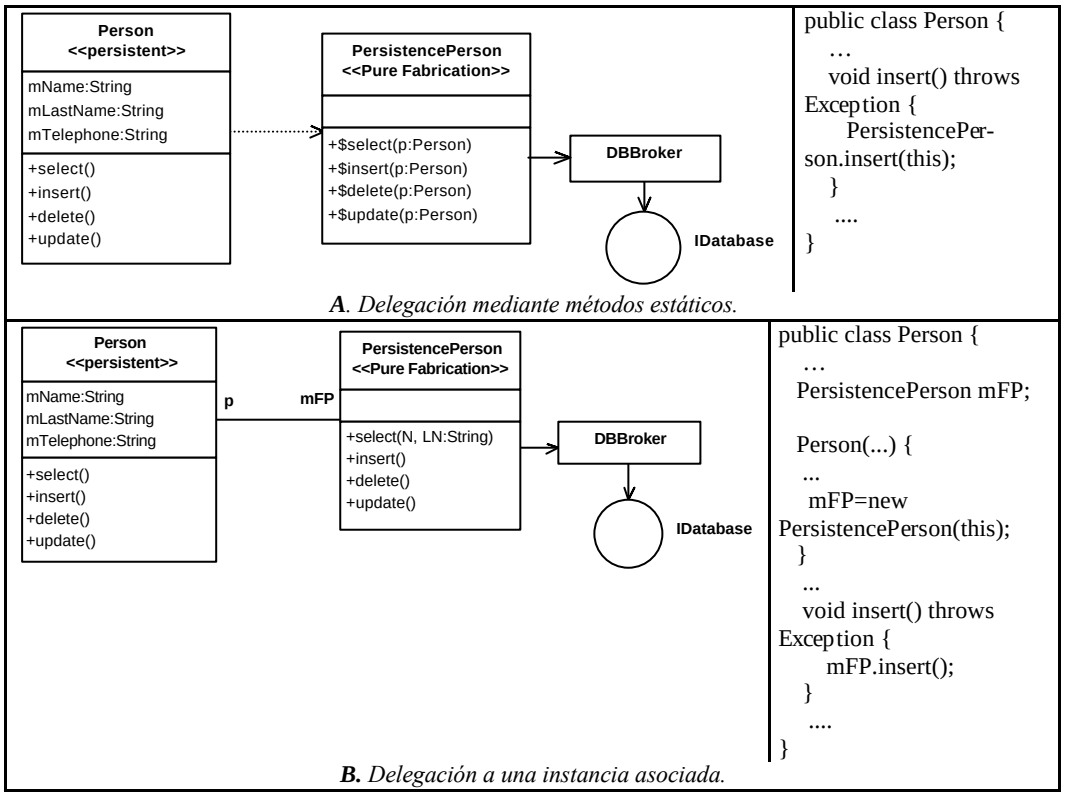


Figura 2. Posibles formas de delegar las operaciones de persistencia.

4. Se genera, además, la base de datos resultante de transformar el diagrama de clases mediante uno de los siguientes patrones:
   - 4.1 Una clase, una tabla
   - 4.2 Un árbol de herencia, una tabla
   - 4.3 Tenemos pendiente de implementación la generación de la base de datos mediante el patrón "Un camino de herencia, una tabla.

     La siguiente figura muestra tres esquemas relacionales (B, C y D) obtenidos de la aplicación de estos tres

patrones de transformación al diagrama de clases (A).

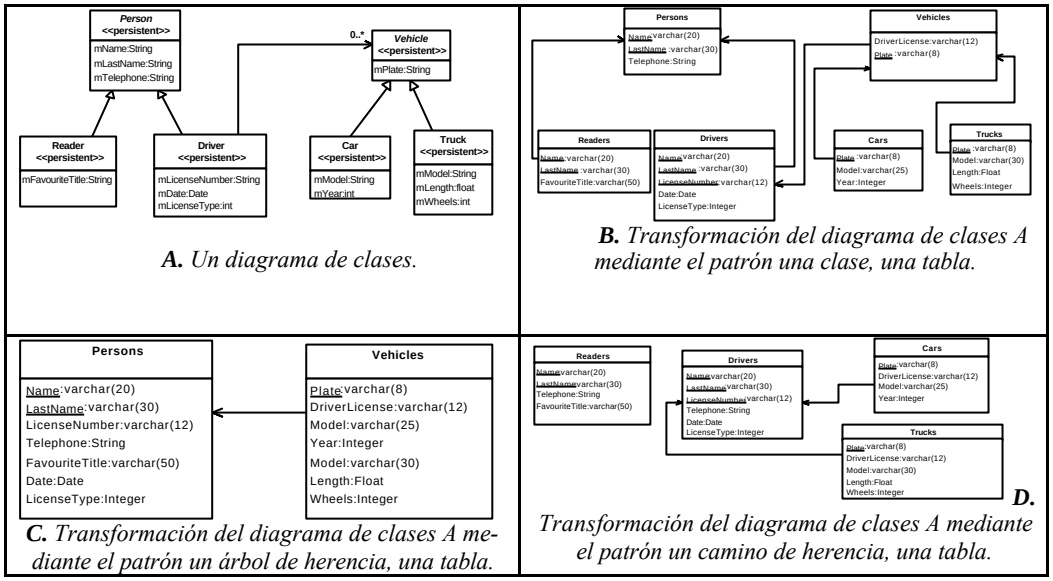


Figura 3. Tres posibles transformaciones a relacional de un mismo modelo de clases.

La base de datos puede generarse directamente en un fichero Access, en una base de datos de un servidor SQL Server, o bien como un programa de definición de datos en SQL 92.

Con estas consideraciones, los objetos que intervienen y se generan para gestionar en cierto escenario los datos de una instancia de clase Persona son los mostrados en la Figura 4 (suponiendo que se han seleccionado fabricaciones puras para delegar las operaciones de persistencia).

3. Utilización de la herramienta en los laboratorios.

La herramienta es capaz de generar en segundos varias versiones diferentes de una aplicación que gestiona una base de datos. Además de ser ejecutable, la aplicación generada puede ser importada desde Rational Rose o alguna otra herramienta (como Poseidon, de Gentleware), de manera que puede cargarse su diagrama de clases para realizar comparaciones de los diferentes diseños, etc.

La Figura 5 muestra un fragmento de la estructura de la aplicación obtenida al generar código para las clases que hemos recuadrado, y que fueron originalmente dibujadas con Rational Rose:

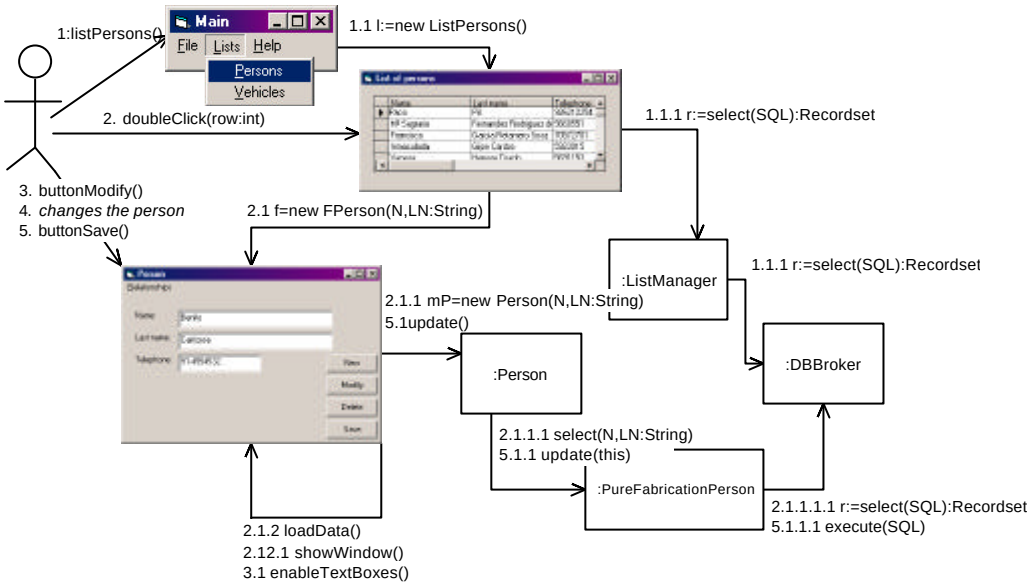


Figura 4. Objetos que intervienen en una aplicación generada por la herramienta, en este ejemplo para gestionar cierto escenario de una persona.

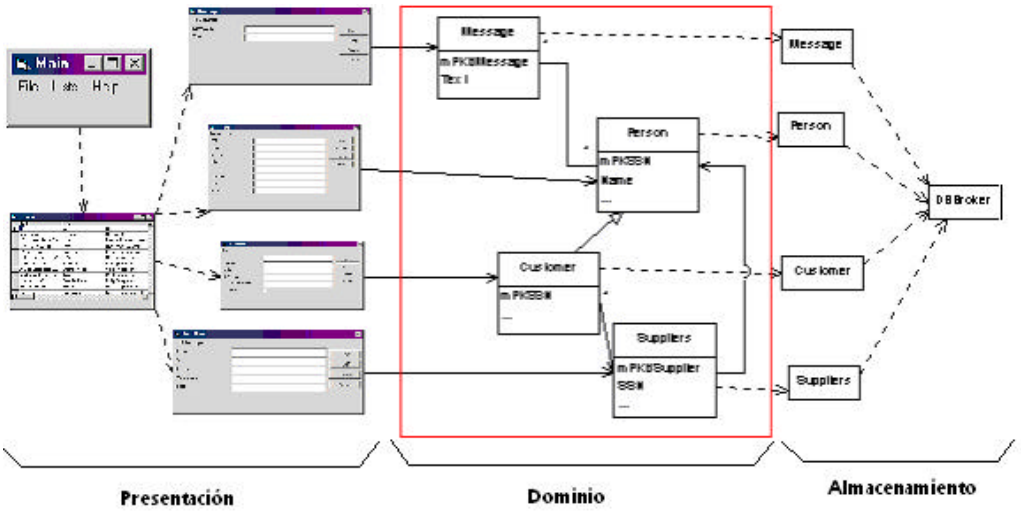


Figura 5. Fragmento de la estructura de una de las aplicaciones generadas.

Cuando el usuario ejecuta la aplicación generada, se encuentra con una pantalla principal que le da a acceso a los listados de todas las tablas existentes en la base de datos. Haciendo doble clic en una de las filas de la lista, se abre la correspondiente pantalla de tipo ficha, que permite la ges-

tión del registro asociado a través de la capa de Dominio.

4. Arquitectura de la herramienta generadora.

La Figura 6 muestra un fragmento del diseño de la herramienta generadora: incorpora la definición de un metamodelo que permite representar aplicaciones de tres capas. La clase "Three-tier application" contiene tres referencias a objetos que se encargan de la generación del código para la capa de Presentación, Dominio y Almacenamiento mediante la implementación de los tres interfaces representados en la figura. De este modo, la adición a la herramienta de un generador de código para otro lenguaje, base de datos u otro tipo de entorno precisa únicamente la construcción de una clase que implemente el interfaz deseado. En estos momentos estamos implementando dos nuevos generadores que generarán Servlets y JSPs en la capa de Presentación.

5. Conclusiones y trabajos futuros.

En este artículo se ha presentado una herramienta que resulta de gran ayuda para la enseñanza de patrones en asignaturas de Ingeniería del Software. La herramienta toma un diagrama de clases dibujado con Rational Rose y genera código multicapa de alta calidad. La gran ventaja aportada por la herramienta es que genera en muy poco tiempo diferentes aplicaciones, cada una con un diseño diferente (dependiendo de los patrones elegidos), lo que libera a los estudiantes de tediosas tareas de implementación de ejemplos, dejando más tiempo para la aplicación práctica de otros contenidos teóricos de la asignatura.

Referencias.

[1] Buschman F., Meunier R., Rohnert H., Sommerlad P. and Stal M. (1996). A System of Patterns: Pattern-Oriented Software Architecture. Addison Wesley.

[2] Larman C. (1998). Applying UML and Patterns. Upper Saddle River, NJ: Prentice-Hall.

[3] Polo M, Piattini M and Ruiz F. (2001). RCRUD: Reflective Create, Read, Update and Delete. Proc. of the Sixth European Conference on Pattern Languages of Programs (EuroPlop'2001). Irsee, Alemania. También disponible en (9 de mayo de 2002): http://www.inf-cr.uclm.es/www/mpolo

[4] Yoder, JW. (2001). *Patterns for making business objects persistent in a relational database.* Tutorial at the Conference on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA), Tampa Bay, Florida, USA.

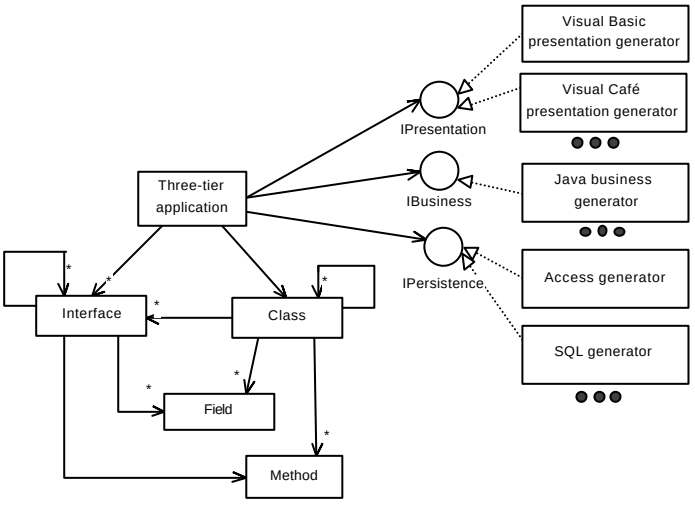


Figura 6. Arquitectura de la herramienta generadora.

Ingeniería de Agentes Software

Òscar Coltell,
Dept. de Lenguajes y Sistemas Informáticos
Universitat Jaume I
12080-Castellón
e-mail: coltell@lsi.uji.es

Ricardo Chalmeta
Dept. de Lenguajes y Sistemas Informáticos
Universitat Jaume I
12080-Castellón
e-mail: rchalmet@lsi.uji.es

Resumen

Este trabajo presenta una propuesta de una asignatura para el segundo ciclo de las titulaciones de Ingeniería Informática, denominada *Ingeniería de Agentes Software*, para ofrecer a los futuros ingenieros una formación más específica en el desarrollo de proyectos de Inteligencia Artificial donde se apliquen los conceptos de agente y sistemas multiagente. En particular, se describen los conceptos, objetivos y estructura de esta asignatura, en forma de proyecto docente.

1. Introducción

Tradicionalmente, se han empleado muchos esfuerzos en la sistematización y formalización de la gestión y el desarrollo de proyectos en el ámbito de la Ingeniería del Software (ISW), desde los primeros trabajos de Bohem [3] hasta la actualidad [14]. Sin embargo, estos aspectos se han descuidado en lo que respecta al desarrollo y gestión de proyectos en ámbitos que se acercan más a la investigación que a la aplicación industrial y profesional [5] como, por ejemplo en la Inteligencia Artificial (AI), que será la disciplina objeto de estudio en este trabajo.

Se ha dado el caso de que en la investigación en IA no han sido absolutamente primordiales los plazos, la eficiencia y los resultados, a diferencia de los proyectos de ingeniería. Entonces, las investigaciones fundamentales se han enfocado hacia el incremento de la potencia expresiva de las teorías específicas, el refinamiento de las técnicas y la reducción de complejidad de los sistemas asociados. Pero se ha perdido de vista que, al fin y al cabo, cualquier proyecto de IA no hace más que emplear recursos de varios tipos, empezando por los humanos, siguiendo con los financieros, y terminando en los técnicos (hardware) y lógicos (software). Así pues, se puede deducir que son igualmente recomendables, en este contexto, la metodología, la rigurosidad y la sistematización bajo un enfoque de ingeniería [17]. Esta recomendación se ha convertido en necesidad imperiosa, cuando la IA ha trascendido los laboratorios de investigación en universidades e institutos, para ser aplicada en el mundo empresarial. Por ejemplo, se están invirtiendo muchos esfuerzos en proyectos que implican, entre otros, la Ingeniería y la Gestión del Conocimiento, la Robótica real y virtual, y los Sistemas Multiagente (MAS: *Multi-Agent Systems*), como uno de los aspectos de la Inteligencia Artificial Distribuida (DAI: *Distributed Artificial Intelligence*) [20], aplicados a distintos campos comerciales e industriales.

La DAI es una subdisciplina de la Inteligencia Artificial que trata, entre otros campos, el enfoque de agente y los MAS. El concepto de *agente* en general se aplica para representar una unidad artificial plenamente operativa que actúa en delegación de un humano o de otro sistema artificial [20]. La preocupación por aplicar sistemas DAI en la resolución de problemas existe desde hace algunos años, sobre todo en el contexto del soporte a la decisión [10] y la gestión de la producción [9]. También se ha utilizado en aplicaciones más específicas, como el control urbano, entrenamiento en tiempo real, control y gestión de instrumentos de medida en aceleradores de partículas, gestión de recursos en factorías distribuidas, sistemas de telecomunicaciones [4].

Más recientemente, con la introducción de los MAS se ha extendido el espectro de aplicación de la DAI con bastante éxito [16] [22] [8]. Pero

también todo esto comportó que se tomara en serio la gestión y desarrollo de proyectos de DAI y de MAS [12].

Además, otro problema derivado de esta eclosión tecnológica ha sido la falta de profesionales con una formación suficiente que combine el conocimiento y la capacidad, tanto en DAI y MAS, como en Ingeniería del Software y Proyectos Informáticos [21]. Esta preocupación también está recogida en la última propuesta del *Computing Curricula 2001 Computer Science* [19].

La situación es parecida en nuestro país empezando por las titulaciones relacionadas en la universidad y su planteamiento académico. Ocurre que tradicionalmente suele existir cierto distanciamiento metodológico y científico entre los que investigan y enseñan en IA respecto de los que trabajan en ISW o en Ingeniería del Hardware. Por tanto, es preciso tomar conciencia de que el trabajo en proyectos que se basan en DAI y/o en MAS deben ser multidisciplinares y empezar a formar a los futuros ingenieros informáticos en ese sentido [5] [11].

Por tanto, el objetivo de este trabajo es el planteamiento de una asignatura que pretende aglutinar los enfoques científicos, metodológicos y técnicos, de la IA, la DAI, los MAS, la Ingeniería de Proyectos y la Ingeniería del Software. Esta asignatura se denomina *Ingeniería de Agentes Software* y se describen sus objetivos y estructura, así como el contexto institucional y el marco académico en que se puede encuadrar con el propósito de ofrecer a los futuros ingenieros informáticos una formación más específica en el desarrollo de proyectos basado en MAS.

En la sección siguiente se describirá la asignatura propuesta intentando cubrir los distintos aspectos docentes y académicos. A continuación, se estudiará la estructura de contenido teórico y práctico. Después se describirán los resultados esperados. Y en la sección siguiente se expresarán las conclusiones obtenidas

2. Descripción general de la propuesta

Para proponer una nueva asignatura dentro de un Plan de Estudios de una titulación universitaria es preciso justificar bajo qué contexto y para qué se quiere desarrollar la misma. Entonces, hay que explicar los distintos factores asociados a cualquier asignatura: contexto, objetivos, metodología docente y estructura de contenidos y prácticas. Y esto es lo que se va a hacer a continuación.

2.1. Contexto de la Asignatura

En el contexto universitario local de los autores, actualmente la asignatura propuesta no existe en los planes de estudios de Ingeniería Informática de la Universitat Jaume I, ni en el de 1991 [1], ni en los que se acaban de estrenar en esta curso académico [2]. Y por la prospección curricular hecha por los autores en diversas universidades, se puede afirmar que tampoco existe en otros planes de estudios (salvo error o excepción). Sin embargo, esta materia debería ser tenida en cuenta para futuras revisiones de planes de estudios a medida que la IA, la DAI y los sistemas orientados a agentes vayan tomando mayor protagonismo en las aplicaciones industriales, comerciales, y científicas en general [18] [19]. Por lo tanto, el contexto académico de esta asignatura es el segundo ciclo de Ingeniería Informática en la intensificación de Sistemas de Información. Sería recomendable que se ofertara en el último curso, posteriormente a *Ingeniería del Software* y otras materias relacionadas con la Inteligencia Artificial.

La experiencia acumulada por los autores en asignaturas relacionadas con la *Ingeniería del Software*, la Ingeniería de Integración de Sistemas y la Ingeniería de Sistemas de Información para la Producción, tanto de segundo ciclo, como en tercer ciclo, ha sido valiosa a la hora de diseñar el contenido de la asignatura que se presenta. De hecho, una de las asignaturas de tercer ciclo que se imparte actualmente, denominada *Desarrollo de Software Orientado a Objetos Avanzado*, es un pequeño prototipo de lo que los autores conciben como una asignatura de más envergadura.

2.2. Objetivos de la asignatura

Se pretende que los alumnos adquieran conocimientos teóricos y prácticos específicos para el desarrollo de agentes y sistemas multiagente en un contexto amplio de la DAI y de las aplicaciones software (MAS, sofbots) locales o

en Internet (web). Se quiere introducir a los alumnos en el manejo de herramientas de desarrollo de agentes comerciales cuyas características son más estables que los prototipos universitarios. También se tiene la intención de fomentar las habilidades del trabajo en equipo multidisciplinar para el desarrollo de proyectos DAI y MAS en particular. En resumen, se pretende formar el embrión del futuro Ingeniero de Sistemas Multiagente.

2.3. Metodología Docente

Dado el cariz científico y tecnológico de la asignatura, se ha adoptado un enfoque docente basado en un modelo instruccional teórico denominado MISE (Modelo Instruccional de la Situación Educativa) [6], donde se plantea fundamentalmente la interacción entre profesor, asignatura y estudiante, y da especial importancia al proceso de enseñanza/aprendizaje con distintos niveles de objetivos: metas de la titulación, generales de la asignatura y operativos de cada unidad instruccional. Con respecto a los objetivos generales, se establecen según tres dominios distintos: cognitivo, socio-afectivo y psicomotor.

Con respecto al *dominio cognitivo*, se ha establecido una segmentación del contenido de la asignatura. En primer lugar, se ha establecido un núcleo de contenido que sea expuesto por el profesor en forma de clase magistral, pero que sirva de introducción y de enganche al resto de la asignatura. En segundo lugar, se ha diseñado un espectro de temas fuera del núcleo que sean menos teóricos y que puedan ser desarrollados por los alumnos. En tercer lugar, el profesor debe proporcionar las prácticas introductorias relacionadas con la parte que él mismo ha expuesto. Y en cuarto lugar, los alumnos deberían abordar la realización de un proyecto de mayor envergadura, que aglutinara los conocimientos teóricos y prácticos nucleares, pero que se base en los temas periféricos.

Con respecto al *dominio socio-afectivo*, se pretende facilitar que los estudiantes desarrollen su habilidad de ser responsables en su trabajo, para trabajar y tomar decisiones en equipo, que sean constructivos y buenos comunicadores, etc. Para ello, por una parte se ha pensado que los alumnos establezcan un "contrato" con el profesor en el que se comprometan a desarrollar un tema elegido del conjunto de temas periféricos. También se pueden añadir otras cláusulas que estipulen una valoración adicional si el tema se expone al resto de compañeros, sea en forma de seminario, sea en forma de grupos de discusión. Por otra parte, se ha previsto dar la suficiente flexibilidad y responsabilidad a cada alumno para que desarrolle la asignatura o los trabajos y prácticas, de forma individual o colectiva formando un equipo de trabajo.

Con respecto al *dominio psicomotor*, se pretende que los alumnos desarrollen sus habilidades en el desarrollo de las prácticas, trabajando con las herramientas de diseño y desarrollo, y adquiriendo experiencia en la aplicación de estas herramientas a casos concretos.

2.4. Evaluación de la asignatura

Dada la flexibilidad y responsabilidad que se puede conceder a cada alumno para que desarrolle la asignatura o individualmente o formando equipo de trabajo, la evaluación debe estar acorde con las propuestas metodológicas mencionadas y así, debe estar compartimentada y aplicarse según la modalidad de trabajo elegida por los alumnos.

Se valorará en primer lugar el contenido teórico teniendo en cuenta la diversidad de recorridos. Si el alumno ha elegido desarrollar solamente el temario básico, se le valorará hasta un nivel máximo previamente estipulado y publicitado, que siempre será menor que la nota máxima oficial. En cambio, si el alumno ha elegido desarrollar además alguno de los temas periféricos mediante su propio contrato, la valoración podrá ser superior. Y finalmente, si el alumno presenta en público el trabajo realizado, puede alcanzar la máxima nota establecida para esta franja.

Por otra parte, se valorará el trabajo continuado del alumno en las sesiones de prácticas, tanto en el aspecto técnico del uso de las herramientas, como del aspecto socio-afectivo en el trabajo en equipo y participación activa. Para que los alumnos puedan revisar el esfuerzo realizado, se debe pedir una memoria de prácticas que incluya un resumen de los resultados obtenidos en cada una de las prácticas o sesiones.

Finalmente, se valorará la capacidad de trabajo en equipo y alcance de los objetivos en el

dominio cognitivo según la Taxonomía de Bloom [6], comprensión, aplicación y resolución de problemas, mediante el desarrollo de un pequeño proyecto que integre las disciplinas mencionadas. También se valorará la responsabilidad en el trabajo y la capacidad de superación mediante la programación de revisiones periódicas en las que el profesor pueda supervisar los avances técnicos y las actitudes respecto de las dificultades surgidas.

Aunque no es realmente imprescindible en el diseño de la asignatura, se puede dar una estimación inicial cuantitativa del peso de cada aspecto valorado: valoración teórica hasta el 60%, valoración práctica hasta el 15%, y valoración aplicativa hasta el 25%, sumando así el 100% de la calificación total. Dentro de estos pesos, hay que considerar, por ejemplo, las rutas de temario planteadas, siendo del 40% la ruta básica, y dejando el margen del 20% restante para evaluar los contratos.
aplicación a casos concretos.

2.5. Caracterización académica

Dados los objetivos y el planteamiento general de esta asignatura, se estima que su carga docente debe ser considerable y no debe encuadrarse en un solo semestre, sino distribuirse a lo largo de todo el curso. Según las estipulaciones oficiales sobre planes de estudios, a las asignaturas anuales se recomienda que se les asigne entre 9 y 15 créditos. En el caso concreto de este trabajo, se ha elegido la máxima carga de 15 créditos. La distribución de los mismos se detalla en la sección siguiente.

Por otra parte, no se considera necesario estudiar aquí el carácter de troncalidad, obligatoriedad u optatividad de la asignatura porque eso está de momento regulado por las leyes correspondientes.

Finalmente, se recomienda que esta asignatura se asigne al último curso del segundo ciclo de Ingeniería Informática, sea en la intensificación de Sistemas de Información, sea en otra más próxima a la IA.

3. Estructura de la asignatura

La estructura de la asignatura se basa lógicamente en el contenido teórico y práctico que aporten conocimientos de las disciplinas mencionadas más arriba. Así, principalmente se incluirán la Inteligencia Artificial e la Ingeniería del Software, pero restringidas a meras introducciones para no solapar con otras asignaturas. Y después se incluirán los conocimientos integrados, como los Agentes y los MAS. Es importante incorporar también una introducción a la Ingeniería de Requisitos, de Componentes y de Integración porque los sistemas de IA suelen tener planteamientos complejos. Y se cierra la asignatura con una exposición de las distintas tecnologías implicadas en el desarrollo de sistemas DAI y MAS.

3.1. Estructura del contenido teórico

El contenido teórico se ha dividido en seis unidades instruccionales: Introducción a la Ingeniería de Agentes, Revisión de Disciplinas Asociadas, Enfoques, Estrategias y Metodologías de Desarrollo, Entornos y Herramientas de Diseño y Desarrollo, Calidad y Análisis y Gestión del Riesgo, y Caracterización de Ámbitos de Aplicación de MAS. A continuación se describen con más detalle estas unidades.

Unidad Teórica I: Introducción a la Ingeniería de Agentes. Esta unidad se compone de los temas siguientes: (1) Objetivos, contexto y aplicación de la Ingeniería de Agentes Software; (2) Perfil profesional del Ingeniero de Agentes. (3) Introducción a la DAI; (4) Introducción a los Agentes en general; y (5) Introducción a los Sistemas Multiagente. La carga docente de esta unidad será de 1,5 créditos (15 horas).

Unidad Teórica II: Revisión de Disciplinas Asociadas, Enfoques. Esta unidad se compone de los temas siguientes: (1) Revisión de la Ingeniería del Software; (2) Revisión de la Ingeniería de Requisitos; (3) Revisión de la Ingeniería de Componentes; (4) Revisión de la Ingeniería de Integración de Sistemas; (5) Revisión de la Inteligencia Artificial; (6) Tecnologías para el desarrollo de sistemas software; (7) Tecnologías para el desarrollo de sistemas DAI; (8) Tecnologías para el desarrollo de sistemas MAS. La carga docente de esta unidad será de 1,5 créditos (15 horas).

Unidad Teórica III: Estrategias y Metodologías de Desarrollo. Esta unidad se compone de los temas siguientes: (1) Estrategias

de diseño y gestión de proyectos multidisciplinares; (2) Técnicas estándar para el modelado de sistemas: UML [15]; (3) Metodologías generales de desarrollo de objetos y agentes software; (4) Metodologías específicas de desarrollo de sistemas multiagente. La carga docente de esta unidad será de 2 créditos (20 horas).

Unidad Teórica IV: Entornos y Herramientas de Diseño y Desarrollo. Esta unidad se compone de los temas siguientes: (1) Herramientas de diseño y modelado de objetos y agentes; (2) Entornos de desarrollo de agentes; (3) Lenguajes de modelado, comunicación y generación de agentes y arquitecturas asociadas. La carga docente de esta unidad será de 2 créditos (20 horas).

Unidad Teórica V: Calidad y Análisis y Gestión del Riesgo. Esta unidad se compone de los temas siguientes: (1) Métricas de agentes y arquitecturas de agentes; (2) Calidad de los proyectos y sistemas MAS; (3) Análisis del riesgo; (4) Gestión del riesgo. La carga docente de esta unidad será de 0,7 créditos (7 horas).

Unidad Teórica VI: Caracterización de Ámbitos de Aplicación de MAS. Esta unidad se compone de los temas siguientes: (1) Aplicaciones web, (2) Sistemas heterogéneos integrados; (3) Sistemas de control complejo (tráfico urbano, etc.) Esta parte debe servir para ofrecer temas concretos de trabajo para los alumnos. La carga docente de esta unidad será de 1,3 crédito (13 horas).

Esta parte suma un total de 9 créditos (90 horas).

3.2. Estructura del contenido práctico

El contenido práctico se ha dividido en seis unidades instruccionales: Planteamiento y Diseño de Proyectos de DAI, Gestión de Recursos y Proyectos, Herramientas de Modelado; Entornos de Desarrollo, Análisis de Calidad y del Riesgo, y Estudio de Viabilidad de Proyectos MAS. En enfoque metodológico general será el trabajo sobre casos prácticos en la medida de lo posible.

Unidad Práctica I: Planteamiento y Diseño de Proyectos de DAI. Esta unidad se centra en el trabajo sobre las actividades y estrategias necesarias para abordar los proyectos en DAI. También se trata de introducir alguna metodología técnica, como por ejemplo, el Método Unificado de Desarrollo de Software [7]. La carga docente de esta unidad será de 1 crédito (10 horas).

Unidad Práctica II: Gestión de Recursos y Proyectos. Esta unidad se centra en el trabajo sobre las actividades y estrategias necesarias para adquirir, gestionar y supervisar los recursos necesarios para desarrollar los proyectos en DAI. Estudia especialmente la gestión de recursos humanos y la dirección y coordinación de equipos de proyecto. La carga docente de esta unidad será de 0,5 créditos (5 horas).

Unidad Práctica III: Herramientas de Modelado. Esta unidad se centra en el trabajo con una o dos herramientas de modelado de sistemas en general y de sistemas de agentes. En el primer caso, se apuesta por alguna herramienta que aplique UML, ya que es un estándar OMG [15] (versiones 1.3, 1.4 y 1.5), y está previsto que en breve alcance la estandarización ISO con la versión 2.0 [13]. Por ejemplo, *Rational Rose 2001 Enterprise Edition*, de *Rational Corp*. La carga docente de esta unidad será de 1,5 créditos (15 horas).

Unidad Práctica IV: Entornos de Desarrollo. Esta unidad se centra en el trabajo con los entornos de desarrollo de agentes y quizá se contemple algún entorno de desarrollo de software orientado a objetos. Uno de los posibles productos puede ser *Agent-builder* de *Reticular Systems*. La carga docente de esta unidad será de 1,5 créditos (15 horas).

Unidad Práctica V: Análisis de Calidad y del Riesgo. Esta unidad se centra en la aplicación de metodologías y técnicas para el Análisis de Calidad según los contenidos teóricos dados. También se trabaja el Análisis de Riesgos aplicado al desarrollo de los proyectos MAS. La carga docente de esta unidad será de 0,5 crédito (5 horas).

Unidad Práctica VI: Estudio de Viabilidad de Proyectos MAS. Esta unidad se centra en el diseño y estudio preliminar de proyectos concretos pertenecientes a los distintos ámbitos de aplicación de los MAS que se han caracterizado en la parte teórica. En principio, los ejemplos son proporcionados por el profesor. Pero también se puede considerar que esta unidad sirva para el planteamiento inicial del proyecto a desarrollar en la otra franja de evaluación. La carga docente de esta unidad será de 1 crédito (10 horas).

Esta parte suma un total de 6 créditos (60 horas).

4. Resultados

El resultado estimado es un proyecto docente sobre una propuesta de nueva asignatura. Pero éste se muestra parcialmente desarrollado ya que no se han descrito los detalles de objetivos y estructura para cada uno de los temas teóricos y unidades prácticas, así como las actividades complementarias y ejercicios a realizar. Tampoco se incluye la bibliografía básica y recomendada. Esto último también estaría en función de la bibliografía disponible una vez se empezara a construir la asignatura.

Por otra parte, el estudiante que haya desarrollado esta asignatura debe poseer, no solamente un nivel aceptable de conocimientos teóricos y prácticos de dos disciplinas como IA e Ingeniería del Software, sino también debe haber adquirido un conjunto de habilidades y responsabilidades que le confieran una perspectiva profesional más amplia. Con ello podrá abordar los problemas reales del campo de los sistemas DAI y MAS con criterios más consolidados.

3. Conclusiones

Se presentan los objetivos, contenido y metodología docente de una asignatura denominada *Ingeniería de Agentes Software* que se puede encuadrada en el segundo ciclo de Ingeniería Informática.

Dado que esta asignatura se ha diseñado con una fuerte carga de créditos, quizá un planteamiento alternativo podría ser la división en asignaturas menores que, con algunas materias complementarias, pudieran formar una intensificación propia.

La *Ingeniería de Agentes Software* pretende formar a los futuros ingenieros informáticos en el desarrollo de sistemas en el ámbito de la Inteligencia Artificial Distribuida y, más concretamente, de Sistemas Multiagente. Así, habrá profesionales preparados para abordar los grandes proyectos que actualmente plantea la sociedad respecto de las nuevas tecnologías, las comunicaciones, la Medicina y la Bioinformática.

Referencias

[1] 22164-RESOLUCIÓN de 20 de Julio de 1991, de la Universidad Jaume I de Castellón, por la que se hace público el Plan de Estudios de Ingeniero en Informática de esta Universidad. Boletín Oficial del Estado, 29-8-1991; 207: 28671-28682.

[2] 22893-RESOLUCIÓN de 20 de noviembre de 2001, de la Universidad “Jaume I” de Castellón, por la que se hace público el plan de estudios de la titulación de Ingeniero en Informática de esta Universidad. Boletín Oficial del Estado, 5-12-2001; 291: 45065-45078.

[3] Boehm B. Software Engineering Economics. Prentice-Hall, 1981.

[4] Chaib-draa B. "Industrial applications of distributed AI". Communications of the ACM; 38(11);1995: 49-53.

[5] Denning PJ. “The profession of IT: who are we?”. Communications of the ACM, 44 (2); 2001: 15-19.

[6] Doménech, F.; *Proceso de enseñanza/ aprendizaje universitario. Aspectos teóricos y prácticos*. Publicacions de la Universitat Jaume I, Castellón ,1999.

[7] Jacobson I., Booch G., Rumbaugh J. *El Proceso Unificado de Desarrollo de Software.* Addison-Wesley, Madrid, 2000.

[8] K. S. Barber, C. E. Martin. "Dynamic reorganization of decision-making groups". Proceedings of the fifth international conference on Autonomous agents; 2001: 513-520.

[9] Laurent JP., Lanusse A., Panet BP. "Distributed artificial intelligence: a necessary paradigm for supervising production management activities". Proceedings of the second international conference on Industrial and engineering applications of artificial intelligence and expert systems; 1989: 326-335.

[10] Lu SC-Y., Thompson JB. "A distributed artificial intelligence approach to integrated engineering design". Proceedings of the first international conference on Industrial & engineering applications of artificial intelligence & expert systems; 1988: 438-446.

[11] M.J.Wooldridge, N.R.Jennings. "Software Engineering with Agents: Pitfalls and Pratfalls". IEEE INTERNET COMPUTING, 3(3); 1999: 20-27.

[12] Michael Schillo, Hans-Jürgen Bürckert, Klaus Fischer, Matthias Klusch. "Towards a definition of robustness for market-style open multi-agent systems". Proceedings of the fifth international conference on Autonomous agents; 2001: 75-76.

[13] Object Management Group. "UML 2.0 RFP. Request For Proposal UML 2.0 Diagram Interchange RFP". OMG Document: ad/2001-02-39, October 22, 2001. Obtained from http://www.celigent.com/omg/uml2wg.htm, at 03/02/2002.

[14] Pressman RS. *Ingeniería del SW. Un enfoque práctico* (5ª ed.)Mc Graw-Hill; 2001.

[15] Rumbaugh J., Jacobson I., Booch G. *El Lenguaje Unificado de Modelado. Manual de Referencia*. Addison-Wesley, Madrid, 2000.

[16] Samir Aknine, Hamid Aknine. "Contribution of a multi-agent cooperation model in a hospital". Proceedings of the third annual conference on Autonomous Agents; 1999: 406-407.

[17] Sánchez MA, Gómez-Senent E. "La profesión de ingeniero". Gómez-Senent E. (ed.) La ingeniería desde una perspectiva global. Servicio de Publicaciones de la Universidad Politécnica de Valencia, Valencia, 2000.

[18] Spooner DL. "A Bachelor of Science in information technology: an interdisciplinary approach". Proceedings of the thirty-first SIGCSE technical symposium on Computer science education; 2000: 285-289.

[19] The Joint Task Force on Computing Curricula. "Computing Curricula 2001 Computer Science. Final Report, December 15, 2001". ACM Journal of Educational Resources in Computing, 1(3); 2001: 240.

[20] Weiss G (ed.). *Multiagent Systems. Modern Approach to Distributed Artificial Intelligence*. MIT Press, Cambridge, MA (USA), 1999.

[21] Yi Shang, Hongchi Shi, Su-Shing Chen. "Agent technology in computer science and engineering curriculum". Annual Joint Conference Integrating Technology into Computer Science Education, 5th annual SIGCSE/SIGCUE conference on Innovation and technology in computer science education; 2000: 123-126.

[22] Yi Shang, Hongchi Shi, Su-Shing Chen. "An intelligent distributed environment for active learning". Journal of Educational Resources in Computing, 1(2); 2001.

Reparto de la carga de trabajo en la realización de prácticas en grupo mediante una herramienta de estimación

Macario Polo, Mario Piattini y Francisco Ruiz
Escuela Superior de Informática – Universidad de Castilla-La Mancha
Paseo de la Universidad, 4
13071-Ciudad Real
macario.polo@uclm.es

Resumen.
En la calificación final de muchas asignaturas se pondera, junto a la nota del examen final o de los posibles parciales, la realización de un trabajo práctico, que a veces puede realizarse en grupo. Es deseable que, en estos casos, los diferentes alumnos que realizan el trabajo dispongan de algún método que les permita realizar, a priori, un reparto lo más equitativo posible, de manera que cada uno realizara un esfuerzo aproximadamente similar al del resto de compañeros.
En este artículo se presenta una herramienta que, con el fin mencionado, hemos empezado a utilizar este año en Ingeniería del Software II, de quinto curso. La herramienta realiza una estimación del tamaño en puntos-función de un sistema orientado a objetos utilizando el método de Antoniol et al. [2]. Un beneficio añadido del uso de la herramienta es la percepción, por parte de los alumnos, de la utilidad de las métricas de software, que habitualmente son simplemente mencionadas en las clases de teoría.

1. Introducción.

En la evaluación de muchas asignaturas interviene, además de la calificación obtenida en los exámenes, la nota obtenida en algún trabajo práctico, que puede ser a veces realizado en grupo. El reparto de las cargas de trabajo entre los diferentes alumnos debe ser lo más equitativo posible, de manera que todos dediquen aproximadamente el mismo esfuerzo al desarrollo del trabajo.

Dependiendo de la asignatura, es posible que los alumnos conozcan ya el concepto de métrica y que, incluso, conozcan detalles sobre algunas de las métricas más conocidas. En el caso de Ingeniería del Software II, del 5º curso de Ingeniería en Informática de la Escuela Superior de Informática de la Universidad de Castilla-La Mancha, el estudio en detalle de algunas métricas forma parte de los contenidos del primer parcial. Además, algunos alumnos conocen métricas adicionales utilizadas en gestión de proyectos, en el caso de que hayan cursado en 4º Planificación y Gestión de Sistemas de Información. El segundo parcial de Ingeniería del Software II se dedica a desarrollo orientado a objetos, haciendo un énfasis especial en la aplicación de patrones de diseño. Como práctica del segundo parcial, los alumnos deben desarrollar una aplicación completa por parejas, debiendo entregar al profesor toda la documentación generada, lo que incluye diagramas de casos de uso, algunos diagramas de interacción, algunos diagramas de estado y uno o más diagramas de clases.

Con el objetivo de que los dos compañeros dediquen el mismo esfuerzo al desarrollo de la práctica, y teniendo en cuenta que en las clases de laboratorio se utiliza la herramienta Rational Rose (aunque puede utilizarse cualquier otra, como Poseidon, que es además gratuita), hemos desarrollado una herramienta que es capaz de realizar una estimación del tamaño en puntos-función del sistema que se debe desarrollar, a partir del diagrama de clases dibujado con Rational Rose. El método utilizado para calcular el número de puntos-función es el descrito por Antoniol et al. [2].

Éste es el primer año en que utilizamos esta herramienta, por lo que no disponemos de datos propios sobre la bondad de las estimaciones realizadas. Los autores del método, sin embargo, dedican gran parte de su artículo a la validación empírica de las estimaciones, ofreciendo excelentes resultados. No obstante, entendemos que hay suficientes diferencias entre los proyectos evaluados (proyectos industriales frente a prácticas de una asignatura) como para revisar,

ajustar y adaptar el método a nuestro entorno docente.

A pesar de esta carencia de resultados propios sobre las estimaciones realizadas, los alumnos perciben claramente los beneficios del uso de métricas en un proyecto real.

El resto de este artículo se organiza de la siguiente manera: en la Sección 2 presentamos resumidamente el método utilizado por la herramienta para realizar la estimación del número de puntos-función. Las características de la herramienta y su utilización en prácticas se presentan en la Sección 3; por último, presentamos en la Sección 4 nuestras conclusiones y trabajos futuros.

2. Breve descripción del método implementado para calcular puntos-función.

Antoniol et al. [2] proponen la utilización del diagrama de clases de un sistema orientado a objetos para realizar una estimación de su futura funcionalidad mediante el uso de puntos-función. Proponen cuatro variantes para realizar la estimación, si bien obtienen resultados muy parecidos con todas ellas. Es por esto por lo que, para desarrollar la herramienta, hemos implementado la variante más sencilla.

En esta variante, cada clase se cuenta como un Internal Logical File (ILF), y cada actor como un External Input File (EIF). Cuando estos conjuntos han sido cargados con los elementos adecuados, se recorren de la siguiente manera:

- Inicialmente, a cada ILF y EIF se le asigna el valor RET=1. Por cada atributo simple del elemento se cuenta un DET, y se le suma un RET por cada asociación o agregación de cardinalidad mayor que uno.
- Si el número de DET es menor que 51, se le asigna complejidad Low (L), y Average (A) en caso contrario.

A continuación se consideran los métodos de cada ILF y EIF. Los autores agrupan las External Inputs, External Outputs y External Inquiries bajo la única denominación de "Service Requests" (SR). Cada método concreto (no abstracto) del ILF o EIF considerado se considera una SR, asignándole un DET por cada argumento simple, y un RET por cada argumento complejo.

A cada SR se le asigna complejidad Low, Average o High de acuerdo con la Tabla 1:

	FTR<=1	FTR<=2	FTR>2
DET<=4	Low	Low	Average
DET<=15	Low	Average	High
DET>15	Average	High	High

Tabla 1. Asignación de complejidad a las Service Requests.

Una vez calculada la complejidad de todos los elementos (ILFs, EIFs y SRs), se calcula el número de Puntos-función sin ajustar (Unadjusted function-points) del sistema según el número de elementos de cada tipo (ILF, EIF o SR) y su complejidad. Para hacer este cálculo se rellenan las casillas sombreadas de la Tabla 2:

	Low				Average				High				Total
Tipo de elemento	Cantidad		Peso		Cantidad		Peso		Cantidad		Peso		
SR		X	3	+		X	5	+		X	7	=	
ILF		X	7	+		X	10	+		X	15	=	
EIF		X	5	+		X	7	+		X	10	=	
							Nº de puntos -función sin ajustar:						

Tabla 2. Pesos para calcular el número de puntos-función sin ajustar.

Una vez calculado el número de Puntos-función sin ajustar, el usuario puntúa, de acuerdo ya a las normas de Albretch [1] y del IFPUG (International Function-Points User Group) un conjunto de 14 parámetros correspondientes a 14 características generales del sistema (véase Tabla 3) con un valor entre 0 y 5, según el grado de influencia de la característica en el sistema (Tabla 4).

1. Comunicación de datos
2. Procesamiento distribuido
3. Perfomance (desempeño)
4. Configuración del equipamiento
5. Volumen de transacciones
6. Entrada de datos *on-line*
7. Interfase con el usuario
8. Actualización *on-line*
9. Procesamiento complejo
10. Reusabilidad
11. Facilidad de implementación
12. Facilidad de operación
13. Múltiples locales
14. Facilidad de cambios

Tabla 3. Características generales del sistema.

Descripción	Nivel de influencia
No está presente o no influye	0
Influencia mínima	1
Influencia moderada	2
Influencia media	3
Influencia significativa	4
Influencia fuerte	5

Tabla 4. Niveles de influencia de las "características generales del sistema".

La suma de los catorce niveles de influencia se utiliza en la siguiente ecuación para calcular el parámetro "Factor de ajuste":

Factor de ajuste = (Nivel de influencia *0,01) + 0,65

Ecuación 1. Ecuación para el cálculo del factor de ajuste.

Calculado el factor de ajuste, se obtiene el resultado esperado, que es el Número de puntos-función ajustados:

Puntos-función ajustados = Puntos-función sin ajustar X Factor de ajuste

Ecuación 2. Calculo del número de puntos-función ajustados.

3. Descripción de la herramienta y su uso para la realización de prácticas.

Como se observa, el proceso del cálculo de puntos-función es complicado. Afortunadamente, nuestra herramienta automatiza la mayoría de los pasos.

La Figura 1 muestra la ventana principal de la aplicación. Permite la apertura de cualquier modelo Rational Rose (ficheros con extensión .mdl). Mediante el botón "Load", la herramienta carga la lista de paquetes existentes en el modelo. Es posible que algunos de los paquetes no deban ser evaluados, por estar siendo completamente reutilizados. En el ejemplo mostrado en la figura se han descartado los paquetes estándar de java (aparecen en la lista "Discarded packages"), de manera que la herramienta evaluará únicamente las clases que deban ser desarrolladas por los alumnos.

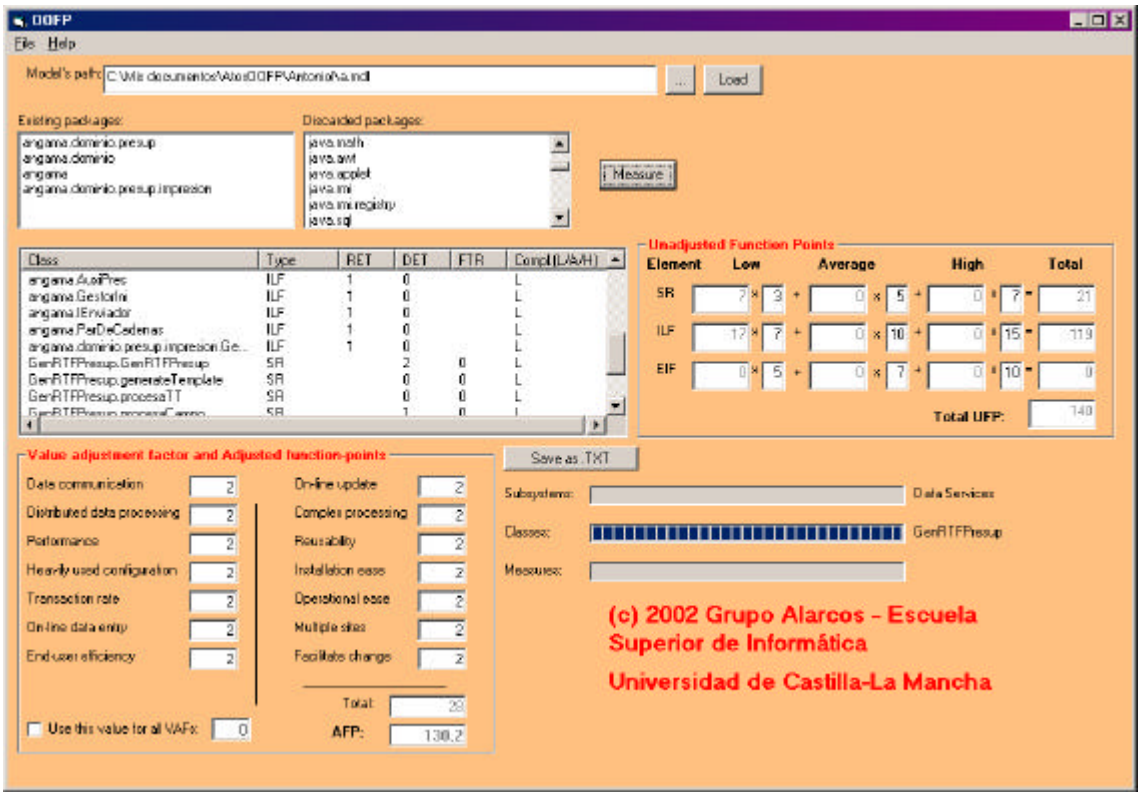


Figura 1. Ventana principal de la aplicación.

Una vez seleccionados los paquetes que se desean evaluar, se arranca el proceso de medida mediante el botón "Measure", que lanza las operaciones necesarias para realizar la medición según el método descrito en la sección anterior.

Los elementos evaluados, su clasificación como ILF, EIF o SR y su complejidad (Low, Average o High) aparecen en la lista que ocupa el lateral izquierdo de la pantalla. Los puntos-función sin ajustar son calculados automáticamente y mostrados en la tabla etiquetada "Unadjusted function-points". El usuario debe dar un valor entre 0 y 5 a las catorce características generales del sistema, con lo que el número de puntos-función ajustados aparece automáticamente en la caja de texto etiquetada "AFP" (130,2 en el ejemplo).

3.1 Uso de la herramienta por los grupos de alumnos.

Los alumnos deben disponer, evidentemente, del diagrama de clases que representa la estructura de su práctica con un cierto nivel de detalle (clases, métodos y parámetros de los métodos). Deben conseguir estructurar el trabajo en paquetes, de tal manera que el número de paquetes a desarrollar por cada alumno tenga un tamaño en puntos-función aproximadamente igual.

La experiencia de utilización en clase comienza con una breve descripción del método de los puntos-función, explicando un poco más en detalle el método de cálculo implementado en la herramienta. Realmente, se percibe en los alumnos mucha curiosidad por utilizar una herramienta de estimación "de verdad", y se observa que salen del aula de prácticas muy

satisfechos, probablemente al comprobar que las métricas tienen aplicaciones bien concretas y útiles, y que no son sólo meras fórmulas de dudosa aplicación.

4. Conclusiones y trabajos futuros.

La herramienta presentada permite, por un lado, repartir equitativamente el esfuerzo que deben dedicar los alumnos a desarrollar las prácticas de Ingeniería del software; y, por otro, les hace ver que las métricas de software son herramientas que tienen verdaderamente mucha utilidad.

La métrica elegida para realizar la estimación (funcionalidad de la aplicación en puntos-función) ha sido criticada por varios autores (por ejemplo, [3]); sin embargo, su utilización en las empresas está muy extendida y en la actualidad se han realizado múltiples adaptaciones para su utilización en el paradigma orientado a objetos ([2][4][5]).

Para finales de este curso dispondremos de datos que nos permitan evaluar la validez de las predicciones de este método en el entorno tan particular en que nos encontramos. Muy probablemente, deberemos realizar ajustes al método para aproximar las predicciones a los valores reales de funcionalidad. Podremos, posiblemente, encontrar una equivalencia válida entre puntos-función y líneas de código, que permita a los alumnos realizar estimaciones de tamaño seguras.

Esta herramienta se encuentra a libre disposición de toda la comunidad universitaria. Esperamos que otros compañeros la descarguen y utilicen, que nos envíen resultados cuantitativos de su aplicación, de manera que podamos mejorar los ajustes y estimaciones para ofrecerlos en una nueva edición de Jenui.

Referencias.

[1] Albretch, AJ. (1979). *Measuring application development productivity.* Proc. of the IBM Application Development Symposium (pp. 83-92). Monterrey, Canadá.

[2] Antoniol, G., Lokan, C., Caldiera, G., y Fiutem, R. (1999). A Function Point-Like Measure for Object-Oriented Software. Empirical Software Engineering, 4(3), 263-287.

[3] Dolado, J. y Fernández, L. (1999). ¿Merece la pena usar los puntos de función? Novática, nº 140, 57-62.

[4] Sneed, H. (1999). Project control for software quality. En Proc. of the EXCMO./SCOPE Conference. Shaker Publishing.

[5] Uemura, T., Kusumoto, S. y Katsuro, I. (2001). Function-point analysus using design specifications based on the Unified Modelling Language. Journal of Software Maitnenance and Evolution: Research and Practice, 13(4), 223-243.

Una Experiencia en Evaluación Continua Multicriterio Aplicada en un Laboratorio de Desarrollo de Software

Patricio Letelier
Departamento de Sistemas Informáticos y Computación
Universidad Politécnica de Valencia
e-mail: letelier@dsic.upv.es

Resumen

Este trabajo describe la experiencia de mejora docente llevada a cabo en una asignatura de quinto curso en la Titulación de Ingeniero en Informática en la Facultad de Informática de la Universidad Politécnica de Valencia. El cambio aplicado a la asignatura radica principalmente en pasar de un sistema de evaluación con pocos controles durante el cuatrimestre y con mucha ponderación para el examen final, hacia un planteamiento de evaluación continua con muchos controles durante el cuatrimestre y reduciendo la ponderación del examen final. Se explica es esquema de trabajo utilizado, la planificación de actividades y evaluaciones, y finalmente se comentan los resultados obtenidos.

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 2001-2002 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 progamas de ayuda a la mejora de la enseñanza del Proyecto EUROPA[1] [1]. La asignatura objeto de esta experiencia se denomina Laboratorio de Sistemas de Información[2] (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 el dotar al alumno de una visión integrada de la ingeniería del software. En LSI se aborda una revisión práctica del triángulo del éxito en desarrollo de software: notación-herramienta-proceso [2]. Se ejercitan las principales notaciones del enfoque estructurado (DFDs, DEs, E-R, etc.) y del enfoque orientado a objetos (UML). Se realizan casos de estudio utilizando las herramientas CASE System Architect y Racional Rose. Uno de los casos de estudio se aborda como un proyecto de desarrollo de software siguiendo la metodología Rational Unified Process (RUP).

LSI lleva implantada 4 años y ha tenido una matrícula de entre 40 y 60 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 implantación del proyecto de mejora realizado en LSI y los resultados obtenidos pueden ser útiles para experiencias similares en otras asignaturas.

El resto de este artículo está organizado como se describe a continuación. En la sección 2 se presenta el análisis de la situación actual (previa a la implantación del proyecto docente). La sección 3 define los objetivos del trabajo de mejora en LSI. En la sección 4 se describe el plan de actividades y el método docente. En la sección 5 se detalla el método de evaluación. Finalmente en la sección 6 se establecen las conclusiones del trabajo, comentando los resultados de la experiencia.

[1] Específicamente se trata de los programas AME2: "Nuevos Métodos de Enseñanza-Aprendizaje" y AME3: "Mejora de los Sistemas de Evaluación".

[2] http://www.dsic.upv.es/asignaturas/facultad/lsi/

2. Análisis de la Situación Previa de LSI

LSI se basa en la resolución de 3 casos de estudio de manera asistida por el profesor. La evaluación se establece mediante la entrega del resultado de cada caso de estudio más un examen final (que aporta el 50% de la nota final).

En LSI el trabajo desarrollado por el alumno esencialmente corresponde a tareas de modelado de sistemas de información. Esto hace que las soluciones no sean únicas y dependiendo de diversas consideraciones se intenta establecer cuáles alternativas de modelado son mejores, de acuerdo con criterios de calidad. En este contexto lo que parece más aconsejable es generar una solución consensuada e iterativa pasando por distintos niveles; reflexión y propuesta individual, discusión en grupos y debate (toda la clase).

En los últimos años, se ha introducido cierta diversificación en las actividades, añadiendo lecciones magistrales para presentar aspectos teóricos relevantes y para la introducción a las herramientas CASE utilizadas. Por otra parte se han implantado espacios de discusión en grupos de dos alumnos y generales (toda la clase) para comentar alternativas de modelado. Estos cambios repercutieron favorablemente en la opinión de los alumnos respecto de la asignatura, lo cual se reflejó en las encuestas docentes. Sin embargo, aún se detectaban una serie de inconvenientes, los cuales se indican a continuación:

- El trabajo de laboratorio se realiza en grupos de 2 alumnos. Aunque esto permite cierto intercambio de ideas y debate en la resolución de casos, está lejos de poder recrear el trabajo de equipo en un proyecto de desarrollo de software real.
- Considerando una matrícula de 50 alumnos, se obtienen 25 grupos de 2 alumnos. La gran cantidad de material resultante de los casos de estudio hace muy difícil su evaluación en detalle y normalmente los resultados se entregan al alumno muy desfasados respecto del momento en el cual se ha realizado el trabajo.
- Existen una serie de conceptos fundamentales que deben ser conocidos por los alumnos previamente, para un mejor aprovechamiento del caso de estudio. Hasta ahora esto se ha intentado cubrir mediante clases magistrales revisando dichos aspectos, pero esto ha restado demasiadas horas a la resolución de los casos de estudio.
- Por el contenido teórico que aborda LSI (ingeniería del software), el examen final, aunque enfocado a aspectos tratados durante el curso, suele ser demasiado amplio en contenidos. Los resultados en el examen final son muy bajos respecto a los resultados de laboratorio, lo cual causa desilusión en los alumnos. Hasta ahora se ha utilizado un examen de selección múltiple.

3. Objetivos del Proyecto de Mejora Docente

De acuerdo con los problemas identificados durante el análisis de la situación previa, el proyecto de mejora para LSI tuvo como objetivo el impulsar los siguientes cambios:

- Se incorporará un sistema de evaluación continua multicriterio. En lugar de evaluar al final de cada caso de estudio, se establecerán actividades cortas y se evaluarán cada una de ellas. De esta forma no se abandona el esquema de trabajo en casos de estudio, pero se mejora el seguimiento y apoyo del avance del alumno. La evaluación del trabajo asociado a dichas actividades constituirá el 70% de la nota final. Para poder establecer con mayor precisión las diferencias individuales, se mantiene el examen final, pero ahora con una ponderación de sólo 30% y en lugar de ser un examen de selección múltiple será un examen de desarrollo.
- Se promoverá el establecimiento de las soluciones en varios niveles; trabajo individual, discusión en grupos de dos y discusión en equipos, fomentando al final de cada actividad un debate. En particular para el caso de estudio más extenso (que consiste en realizar un proyecto de desarrollo de software, y que ocupa un 50% de las clases de la asignatura) se definirán equipos de desarrollo de 6 a 8 alumnos, estableciendo roles para cada uno de ellos (manager, analista, desarrollador, encargado de pruebas).
- Previendo el esfuerzo adicional asociado a la evaluación periódica de los alumnos, y para aprovechar al máximo la posibilidad de

seguimiento y retroalimentación, se introducirá un esquema de auto-evaluación para algunas actividades. Siguiendo unas pautas de evaluación establecidas por el profesor la nota podrá ponerla, el propio alumno, su compañero de grupo (cuando se trabaje en parejas) u otros integrantes del equipo (cuando se realiza el proyecto de desarrollo de software). De esta forma el profesor en este tipo de actividades podrá concentrarse en el seguimiento y corrección junto a los alumnos, discutiendo suposiciones y alternativas, y no en la calificación del alumno. Además, se promoverá la utilización de horas de tutorías para realizar parte del seguimiento de las actividades de los alumnos.

- Para la introducción de conceptos fundamentales previos a la realización de los casos de estudio se utilizarán tres medios: (a) breves presentaciones (de no más de media hora) mediante clase magistral utilizando trasparencias que quedarán disponibles para los alumnos, (b) realización de problemas pequeños guiados por el profesor y (c) controles escritos asociados a la lectura de artículos.
- Se fomentará el aprendizaje autónomo ofreciendo a los alumnos desarrollar un trabajo voluntario en temas de ingeniería del software. Con los trabajos presentados se realizará a fin de curso un seminario de ingeniería del software. Estos trabajos serán guiados por el profesor y tendrán una compensación en la nota final del alumno (que puede alcanzar a un incremento en un 10% de su nota final).

En resumen, las líneas directrices que pretende implantar o consolidar este proyecto de mejora docente son:

a. Diversificación de métodos de enseñanza y promoción del aprendizaje activo
b. Evaluación y seguimiento continuo del alumno
c. Trabajo en equipo enfrentando situaciones cercanas a las del mundo laboral
d. Incorporación al alumno al proceso de evaluación, autoevaluándose y evaluando a compañeros según pautas establecidas por el profesor
e. Mejor aprovechamiento de las horas de tutorías estableciendo en ellas revisiones de seguimiento de las actividades de los alumnos

4. Descripción del Plan de Actividades y del Método Docente

En el Anexo se muestra el detalle de las actividades que se realizaron, el método docente para cada una de ellas, el material de apoyo proporcionado al alumno y el tipo de evaluación.
Deben tenerse presente las siguientes consideraciones:

- Cada sesión corresponde a 2 horas de clases. Cada alumno tiene dos sesiones semanales. Se asume que la dedicación fuera de clases debería ser de alrededor de 3 horas semanales. Para los trabajos voluntarios del Seminario de Ingeniería del Software se estima una dedicación adicional de unas 10 horas.
- Tanto el planteamiento de cada actividad como el material de apoyo correspondiente son publicados en el microweb UPV[3] de la asignatura.
- La planificación mostrada en las tablas del Anexo corresponde a la planificación estimada, corregida con algunas pequeñas variaciones que se produjeron en la implantación.

5. Método de Evaluación

Para aprobar la asignatura se exige asistir al menos a un 80% de las sesiones.
El trabajo realizado durante el curso tiene una ponderación de 70% sobre la nota final y se establece mediante evaluaciones de actividades (todas de igual peso) tal como muestra la planificación de la tablas del Anexo. De las 21 evaluaciones planificadas, para efectos de cálculo de la nota media, se descartan las 3 peores evaluaciones del alumno. Tal como se indica en la tabla del Anexo, 8 evaluaciones corresponden a auto-evaluaciones de los alumnos.

[3] http://www.upv.es/pls/oalu/est_asi.Info_asig

El examen final tiene una ponderación de 30%. Además, la evaluación del trabajo voluntario (para los alumnos que lo realicen) podrá incrementar hasta un 10% la nota final.

6. Conclusiones

En términos generales el resultado ha sido muy satisfactorio. Desde la perspectiva del docente, se ha constatado un incremento en la motivación de los alumnos, los cuales se han involucrado mucho más en 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 se exigía al menos un 80% de asistencia), el incremento en asistencia a tutorías, la participación en clases y los mejores resultados en el examen final. 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 no oficial en la cual a cada alumno se le pidió que escribiera en un papel lo que más le gustó de la asignatura, lo que cambiaría o mejoraría, y que hiciera una estimación de tiempo de dedicación fuera del aula. Mayoritariamente valoraron el hecho de trabajar en equipos y el esquema de actividades cortas. Muchos sugirieron la eliminación del examen final por considerar que era suficiente con las evaluaciones previas. Finalmente, en promedio estimaron haber dedicado unas cuatro horas semanales fuera del aula.

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 resultó la introducción de auto-evaluaciones tanto por su aspecto formador para los alumnos como para aminorar el trabajo de evaluación por parte del profesor. Por otra parte, el esquema de trabajo y sobre todo el material 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.

Una deficiencia que se detectó y que era de esperar, fue el estrecho margen de flexibilidad respecto del tiempo disponible para las actividades planificadas. Aunque la estimación de tiempo para cada actividad resultó ser bastante apropiada, cualquier imprevisto podía impactar el programa, lo cual exigía llevar un control demasiado riguroso. Particularmente lo normal es que al arrancar el caso de estudio en equipos al principio sea necesario una integración de los miembros, que cada uno asuma su rol y se organicen. Sin embargo, esto está en conflicto con la dinámica inicial del trabajo en el caso de estudio. El resultado de esto fue que la mayoría de los grupos comenzaron retrazados en cuanto a los objetivos planteados e incluso alguno llevó esta situación hasta el fin del caso de estudio. Para evitar este problema podría darse por iniciado el caso de estudio con anticipación al momento en el cual las clases se dedican exclusivamente a él, lo cual permitiría tener ese tiempo de reacción inicial en cada grupo.

El seminario de Ingeniería del Software resultó satisfactorio, se presentaron 8 trabajos. Sin embargo, sería recomendable no dejarlo para fin de curso puesto que como es normal los alumnos están más preocupados por los exámenes de sus asignaturas.

El trabajo en parejas se reservó para el desarrollo de actividades de una sesión. Las parejas no eran establecidas para todo el curso. Esta fórmula de trabajo resulta normalmente efectiva, pero además en nuestra experiencia esto se vio favorecido por la motivación que significaba la evaluación inmediata de la actividad una vez finalizada la sesión.

La experiencia de auto-evaluación resultó interesante. Aunque, como se podía suponer, los alumnos en general se asignaron calificaciones buenas. Sin embargo, en el caso de evaluaciones cruzadas en parejas y sobre todo evaluaciones de equipo al parecer se crearon lazos de colaboración más estrechos, mejorando el nivel de compromiso y de acuerdos entre ellos.

Referencias

[1] Proyecto EUROPA: Una Esnseñanza Orientada al Aprendizaje. Vicerrectorado de Coordinación Académica y Alumnado. 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.

Anexo

Las siguientes tablas detallan las actividades, método docente, material de apoyo para el alumno y tipo de evaluación para cada una de las sesiones de la asignatura. Se dispone de 14 semanas lectivas, lo cual permite tener 28 sesiones de 2 horas cada una.

Nº Sesión	1
Tema	Introducción
Actividad	a) Presentación de la Asignatura b) Organización de parejas y equipos Presentación "Modelado de SI y Herramientas CASE"
Método Docente	Lección Magistral
Material de Apoyo	a) microweb UPV de LSI y web de la asignatura b) Trasparencias de la presentación, c) Documento: Introducción a Herramientas CASE y System Architect
Tipo de Evaluación	

Nº Sesión	2
Tema	Enfoque Estructurado: Modelado de Procesos
Actividad	a) Problema: Alternativas de Comunicación en un DE b) Problema: Ingeniería Inversa desde C a un DE
Método Docente	Resolución de los problema trabajando en parejas. Debate
Material de Apoyo	Documento: Análisis y Diseño Estructurado en System Architect
Tipo de Evaluación	Autoevaluación para cada problema

Nº Sesión	3
Tema	Enfoque Estructurado: Modelado de Procesos
Actividad	Caso de Estudio: Posicionamiento 2D. Evaluación de solución
Método Docente	Trabajando en parejas los alumnos deben evaluar el análisis y diseño propuestos. Debate
Material de Apoyo	Documento: Análisis y Diseño Estructurado en System Architect
Tipo de Evaluación	Autoevaluación

Nº Sesión	4
Tema	Enfoque Estructurado: Modelado de Procesos
Actividad	Caso de Estudio: Posicionamiento 2D. Evaluación de solución
Método Docente	Trabajando en parejas los alumnos deben evaluar el análisis y diseño propuestos. Debate
Material de Apoyo	Solución propuesta del caso de estudio como ficheros de System Architect
Tipo de Evaluación	Evaluación cruzada de la pareja

Nº Sesión	5
Tema	Enfoque Estructurado: Modelado de Procesos
Actividad	Caso de Estudio: Posicionamiento 2D. Determinar correcciones a defectos
Método Docente	Trabajando en parejas los alumnos deben establecer correcciones. Debate
Material de Apoyo	Solución propuesta del caso de estudio como ficheros de System Architect
Tipo de Evaluación	Evaluación cruzada de la pareja

Nº Sesión	6
Tema	Enfoque Estructurado: Modelado de Procesos
Actividad	Caso de Estudio: Posicionamiento 2D. Establecer mejoras y ampliaciones
Método Docente	Trabajando en parejas los alumnos deben establecer alternativas de mejora. Debate.
Material de Apoyo	Solución propuesta del caso de estudio como ficheros de System Architect
Tipo de Evaluación	Evaluación cruzada de la pareja

Nº Sesión	7
Tema	Enfoque Estructurado: Modelado de Procesos
Actividad	Caso de Estudio: Posicionamiento 2D. Pruebas
Método Docente	Trabajando en parejas los alumnos deben establecer pruebas de caja blanca para un módulo
Material de Apoyo	Solución propuesta del caso de estudio como ficheros de System Architect
Tipo Ev.	Evaluación por el profesor

Nº Sesión	8
Tema	Enfoque Estructurado Modelado de Datos
Actividad	Presentación: Modelado de Datos en System Architect
Método Docente	Lección Magistral y Problema guiado
Material de Apoyo	Trasparencias de la Presentación
Tipo de Evaluación	

Nº Sesión	9
Tema	Enfoque Estructurado: Modelado de Datos
Actividad	Problema: Modelado de Datos para Orden de Pedido
Método Docente	Trabajando en parejas los alumnos deben construir un Modelo ER usando el CASE SA. Debate.
Material de Apoyo	
Tipo de Evaluación	

Nº Sesión	10
Tema	Enfoque Estructurado: Modelado de Datos
Actividad	Caso de Estudio: Modelado ER de empresa Flexicash. Identificación de Entidades y Relaciones
Método Docente	Trabajando en parejas los alumnos deben construir un Modelo ER en System Architect Debate.
Material de Apoyo	
Tipo Ev.	Evaluación cruzada de la pareja

Nº Sesión	11
Tema	Enfoque Estructurado: Modelado de Datos
Actividad	Caso de Estudio: Modelado ER de empresa Flexicash. Validación del modelo y evaluación de la solución.
Método Docente	Trabajando en parejas los alumnos deben evaluar el modelo y validarlo respecto a los requisitos. Debate.
Material de Apoyo	
Tipo de Evaluación	Evaluación cruzada de la pareja

Nº Sesión	12
Tema	Enfoque Orientado a Objetos
Actividad	Introducción a Rational Rose
Método Docente	Trabajando en parejas los alumnos deben desarrollar una guía de laboratorio usando Racional Rose
Material de Apoyo	Guía de Laboratorio
Tipo Ev.	Evaluación por el profesor

Nº Sesión	13
Tema	Enfoque Orientado a Objetos
Actividad	a) Control escrito: Desarrollo de a) Software Orientados a Objeto b) Caso de Estudio: Desarrollo SW Empresa de Distribución. Fase Inicio
Método Docente	a) Lectura individual de un artículo técnico b) Presentación: Introducción a Rational Unified Process
Material de Apoyo	a) Artículo "Real-Life Object-Oriented Systems" b) Trasparencias de la Presentación
Tipo de Evaluación	Evaluación del control escrito por el profesor

Nº Sesión	14
Tema	Enfoque Orientado a Objetos
Actividad	Caso de Estudio: Desarrollo SW Empresa de Distribución. Fase Inicio. Modelo del Dominio, Prototipos de Interfaces y Características del SW

Método Docente	Trabajo en equipos, comienzo del caso de estudio. Construcción del Modelo del Dominio, Prototipos de Interfaces y Características del SW
Material de Apoyo	Documentación de Rational Unified Process
Tipo de Evaluación	

Nº Sesión	15
Tema	Enfoque Orientado a Objetos
Actividad	Caso de Estudio: Desarrollo SW Empresa de Distribución. Fase Inicio, Revisión Artefactos de Inicio
Método Docente	Trabajo en equipos. Validar con el profesor el Modelo de Dominio, Prototipos de Interfaces y las características del SW
Material de Apoyo	Documentación de Rational Unified Process
Tipo de Evaluación	Evaluación por el profesor

Nº Sesión	16
Tema	Enfoque Orientado a Objetos
Actividad	a) Presentación: RUP, Proceso dirigido por los Casos de Uso b) Caso de Estudio: Desarrollo SW Empresa de Distribución. Fase Elaboración, Identificación de Casos de Uso
Método Docente	Trabajo en equipos. Construcción de Diagrmas de Casos de Uso
Material de Apoyo	a) Documentación de Rational Unified Process b) Transparencias de la Presentación
Tipo de Evaluación	

Nº Sesión	17
Tema	Enfoque Orientado a Objetos
Actividad	Caso de Estudio: Desarrollo SW Empresa de Distribución. Fase Elaboración, Revisión de Identificación de casos de uso
Método Docente	Trabajo en equipos. Revisión con el profesor de los casos de uso
Material de Apoyo	
Tipo Ev.	Evaluación por el profesor

Nº Sesión	18
Tema	Enfoque Orientado a Objetos
Actividad	Caso de Estudio: Desarrollo SW Empresa de Distribución. Fase Elaboración, Especificación de Casos de Uso
Método Docente	Trabajo en equipos. Especificar los casos de uso
Material de Apoyo	Documentación de Rational Unified Process
Tipo de Evaluación	

Nº Sesión	19
Tema	Enfoque Orientado a Objetos
Actividad	Caso de Estudio: Desarrollo SW Empresa de Distribución. Fase Elaboración, Revisión de Especificación de Casos de Uso
Método Docente	Trabajo en equipos. Revisión especificación de casos de uso con el profesor
Material de Apoyo	
Tipo de Evaluación	Evaluación por el profesor

Nº Sesión	20
Tema	Enfoque Orientado a Objetos
Actividad	Caso de Estudio: Desarrollo SW Empresa de Distribución. Fase Construcción, Realización de Casos de Uso
Método Docente	Trabajo en equipos. Realización del los casos de uso utilizando Diagramas de Colaboración, Secuencia y de Clases
Material de Apoyo	Documentación de Rational Unified Process
Tipo de Evaluación	

Nº Sesión	21
Tema	Enfoque Orientado a Objetos
Actividad	a) Control Escrito: Estándares para Procesos de Desarrollo de Software b) Caso de Estudio: Desarrollo SW Empresa de Distribución. Fase Construcción, Revisión de Realización de Casos de Uso

Método Docente	Trabajo en equipos. Revisión con el profesor de las realizaciones de los casos de uso
Material de Apoyo	Referencias bibliográficas y web respecto de estándares para procesos de desarrollo de software: CMM e ISO 9000-3
Tipo de Evaluación	Evaluación por el profesor

Nº Sesión	22
Tema	Enfoque Orientado a Objetos
Actividad	Caso de Estudio: Desarrollo SW Empresa de Distribución. Fase Construcción, Diseño e implementación de la BD
Método Docente	Trabajo en equipos. Diseñar e implementar la BD
Material de Apoyo	
Tipo de Evaluación	Evaluación por el profesor

Nº Sesión	23
Tema	Enfoque Orientado a Objetos
Actividad	Caso de Estudio: Desarrollo SW Empresa de Distribución. Fase Construcción, Revisión del Prototipo y Pruebas
Método Docente	Trabajo en equipos. Revisar con el profesor el prototipo de la primera iteración y las pruebas asociadas
Material de Apoyo	
Tipo de Evaluación	Evaluación por el profesor

Nº Sesión	24
Tema	Enfoque Orientado a Objetos
Actividad	Caso de Estudio: Desarrollo SW Empresa de Distribución. Fase Construcción, Revisión del Prototipo y Pruebas
Método Docente	Trabajo en equipos. Revisar con el profesor el prototipo de la primera iteración y las pruebas asociadas
Material de Apoyo	
Tipo de Evaluación	Evaluación por el profesor

Nº Sesión	25
Tema	Enfoque Orientado a Objetos
Actividad	Caso de Estudio: Desarrollo SW Empresa de Distribución. Fase Construcción, Revisión de los artefactos hasta la Release I
Método Docente	Trabajo en equipos. Revisar con el profesor los artefactos de la primera release
Material de Apoyo	
Tipo de Evaluación	a) Evaluación por el profesor b)Evaluación del Manager del Grupo respecto de los integrantes c)Evaluación de los integrantes respecto del Manager

Nº Sesión	26
Tema	Seminario Ingeniería del Software
Actividad	Seminario con los trabajos voluntarios. Parte I
Método Docente	Seminario
Material de Apoyo	
Tipo de Evaluación	Evaluación por el profesor a los alumnos expositores

Nº Sesión	27
Tema	Seminario Ingeniería del Software
Actividad	Seminario con los trabajos voluntarios. Parte II
Método Docente	Seminario
Material de Apoyo	
Tipo de Evaluación	Evaluación por el profesor a los alumnos expositores

Nº Sesión	28
Tema	Conclusiones
Actividad	Presentación Conclusiones de la Asignatura
Método Docente	Lección magistral
Material de Apoyo	Trasparencias de la presentación
Tipo de Evaluación	

Integración de teoría y práctica / gestión y desarrollo en la docencia de la ingeniería del software

Pablo Gervás
Dept. de Sistemas Informáticos y Programación
Universidad Complutense de Madrid
28040 Madrid
e-mail: pgervas@sip.ucm.es

Resumen

Se presenta una propuesta de integración y coordinación entre las partes teórica y práctica de la ingeniería del software destinada a: facilitar la disponibilidad de especificaciones de partida, paliar las restricciones temporales para el desarrollo de proyectos a gran escala, proporcionar a los alumnos una experiencia real de trabajo en equipo y de uso de la documentación para la comunicación dentro de un equipo.

1. Motivación

Muchas de las técnicas que se utilizan hoy en día en ingeniería del software están dirigidas a resolver el problema básico de cómo convertir la tarea de desarrollo de software en un conjunto de subtareas disjuntas que puedan asignarse a partes distintas de un equipo de desarrollo sin dar lugar a una explosión de las necesidades de comunicación interna de todo el equipo [1]. Tareas como documentar los requisitos una vez aclarados con el cliente, describir explícitamente la planificación de las tareas en forma de diagramas de PERT o de Gantt, o generar modelos detallados del diseño [8,9,12] tienen como uno de sus objetivos principales el plasmar el resultado de una de estas subtareas disjuntas en que se ha repartido el proceso de desarrollo, de modo que ese resultado pueda ser utilizado por otra parte del equipo de desarrollo sin necesidad de consultas frecuentes o prolongadas con los miembros del equipo que llevaron a cabo la subtarea original. Esta virtud también puede entenderse de forma diacrónica, de modo que el abandono de miembros del equipo que habían realizado subtareas en el pasado no suponga que el resultado de esas tareas deje de estar disponible para el resto del equipo.

Estas verdades genéricas son a menudo demasiado abstractas para ser captadas en toda su magnitud por los alumnos, que están demasiado entusiasmados con la posibilidad de poner a prueba su recién adquirida capacidad para diseñar e implementar lo que sea menester. Para el tamaño de aplicación que resulta viable desarrollar en el marco de una única asignatura, el alumno medio tiene en la mayoría de los casos capacidad suficiente para retener en la cabeza tanto una visión general de la especificación, como una idea de la planificación suficiente para organizarse, como un esbozo del diseño que le basta para irlo programando. De hecho, esta manera de "documentar" sus propios procesos de trabajo tiene la ventaja importante de llevar incorporados sus propios métodos innatos de mantener un seguimiento continuo durante el proceso de desarrollo, revisar periódicamente esa "documentación", corregir lo que sea necesario, comunicar las modificaciones realizadas a quien puedan afectar (normalmente un proceso paralelo pendiente del propio alumno), y de forma general llevar a la práctica muchas de las recomendaciones básicas de los textos de ingeniería del software, todo ello con un esfuerzo mínimo de tiempo y un gasto de papel y tecnología casi nulo. Esta es la razón de que, en determinadas circunstancias, se pueda desarrollar software con éxito sin aplicar explícitamente técnicas de ingeniería del software.

La labor de enseñar ingeniería del software es una tarea ingrata porque requiere a menudo ponerle al trabajo de los alumnos trabas especiales que se lo hagan más difícil, con el bien intencionado objeto de que aprendan maneras de

hacerle frente a dificultades de ese tipo [3]. Una versión de esta labor radica en imponer la necesidad de trabajar en equipo, lo cual obliga a hacer explícitos parte de esos procesos mentales de "documentación" que mencionaba antes. Aún así, para equipos pequeños, es factible el llegar a sincronizar "documentaciones" mentales del proceso de los participantes mediante una comunicación verbal periódica, sin necesidad de documentos formales o modelos explícitos.

Si imponemos la necesidad de documentar explícitamente como un requisito *sine qua non* del trabajo de la asignatura, independientemente de la cantidad de explicaciones teóricas que aportemos, es probable que los alumnos (siempre sometidos a presiones temporales varias) adopten la solución de llevar a cabo el trabajo de desarrollo de la manera en que les resulta más cómodo (sin ingeniería del software) y generen *a posteriori* la documentación que se les pide (ahorrándose de ese modo las distintas iteraciones sucesivas, el seguimiento, la revisión, la modificación...). Si se atienen al orden explicado en teoría generarán una serie de documentos a cada paso que reflejan esas "documentaciones" mentales que ellos se van construyendo en paralelo, pero rara vez volverán a consultarlos más adelante, resultándoles mucho más cómodo referirse a la versión que recuerdan de cuando redactaron el documento. En el mejor de los casos lo harán convencidos de que la teoría la tienen clara y, si llega el momento, sabrán aplicarla. Quizá uno de los desafíos más importantes en la docencia de la ingeniería del software hoy en día sea el conseguir que ese momento llegue, y que llegue en el marco del trabajo práctico asociado a la asignatura.

2. Problemas a resolver

Es importante combatir la visión, frecuente entre los alumnos y en parte inducida por la posición de la asignatura dentro del plan de estudios, de que la ingeniería del software es simplemente una manera rígida, dogmática y prescindible de encorsetar la programación. Se presentan por orden de importancia relativa los problemas que se pretende resolver con la propuesta.

2.1 Redundancia del trabajo de documentación

Un problema fundamental es el ya mencionado de proporcionar un marco en el que la labor de documentar explícitamente los procesos de desarrollo no sea un trabajo vano, sino que cumpla un cometido concreto con objetivos fácilmente perceptibles y evaluables por el alumno.

2.2 Desarrollo a pequeña o gran escala

Se pretende atacar de maneras diferenciadas dos problemas distintos. Por un lado los alumnos necesitan tener la oportunidad de enfrentarse a título individual a cada una de las cuestiones relevantes que se tratan en la asignatura. Esto requiere que se trabaje en equipos pequeños para que todos los integrantes de un equipo tengan ocasión de participar en los procesos que se lleven a cabo, y adquieran experiencia en el uso de las técnicas y en la resolución de los problemas. Por otro lado, las técnicas de ingeniería del software están pensadas para hacer frente a circunstancias de trabajo que son específicas de (o al menos mucho más frecuentes en) situaciones en que se trabaja con proyectos grandes, con participación de muchos trabajadores, y con una expectativa significativa de continuidad y consistencia del proceso de desarrollo más allá de un proyecto o una plantilla de personal concretos. Esto hace muy deseable que los alumnos experimenten, en la medida de lo posible, circunstancias de trabajo de estas características, para que perciban los problemas que acarrean, la dificultad de solucionarlos, y el papel de las tecnologías de ingeniería del software a la hora de resolverlos.

Las consideraciones curriculares de instituciones importantes son unánimes a la hora de recomendar la inclusión de un proyecto de desarrollo a gran escala en la formación de todo informático [10]. En [5] se presentaban distintos métodos empleados en universidades de EEUU y el Reino Unido para solventar esta cuestión. La mayor parte de ellos no son viables en el contexto que nos ocupa. La única alternativa viable sería la denominada "curso con proyecto encapsulado", consistente en que los estudiantes realicen un proyecto de desarrollo de software después de terminar la mayoría del curso. La presente propuesta pretende adelantar el momento de comienzo del trabajo práctico, mediante una reorganización adecuada de los contenidos teóricos por un lado, y la inclusión de un proyecto a gran escala para el segundo cuatrimestre que entronca directamente con prácticas a pequeña escala realizadas durante el primer cuatrimestre. De este modo se dispone de más tiempo total para

el desarrollo y, para cada concepto, se acercan más en el tiempo el momento de su exposición teórica y el de su posterior puesta en práctica.

2.3 Restricciones temporales

Es difícil desarrollar un proyecto a gran escala dentro del marco temporal que impone una asignatura concreta dentro de un plan de estudios, y que cada alumno llegue a ver finalizado el proyecto en el que ha trabajado, al menos hasta un punto suficiente como para permitirle hacer una valoración personal del rendimiento conseguido con su esfuerzo, y de la medida en que las técnicas utilizadas han servido para hacer frente a los problemas encontrados. Esta dificultad aumenta si además se pretende conjugar el desarrollo de un proyecto a gran escala con la realización de prácticas a pequeña escala.

2.4 Disponibilidad de especificaciones

A la hora de plantear prácticas y proyectos para la docencia de la ingeniería del software es difícil encontrar material de partida sobre el que los alumnos puedan desarrollar una especificación sobre la que basar ejercicios de diseño.

La solución ideal sería conseguir clientes reales que estuviesen dispuestos a ejercer como tales para algún proyecto de formación. Esta solución suele ser inviable por dificultades de disponibilidad de clientes, compatibilidad de horarios de posibles clientes y estudiantes, voluntad de los clientes de aceptar las restricciones asociadas (temporales, en la tecnología o el tamaño de las aplicaciones a realizar), y de imposibilidad de garantizar un servicio de mantenimiento o de seguimiento más allá del final de la asignatura.

Otra solución es dejar el papel del cliente en manos del profesor. Esta solución plantea varias dificultades. Por un lado, el profesor de ingeniería del software es demasiado consciente tanto de cómo debe ser la especificación ideal como de cuáles son los errores más habituales de los clientes o de los ingenieros del software, y corre el riesgo de proporcionar una entrada como cliente que no sea significativa. Por otro lado, si el número de alumnos es apreciable, el profesor deberá o dedicar un momento distinto a atender a cada grupo (con un proyecto distinto, lo que puede dificultar mucho su labor) o decidir que atiende a todos los grupos a la vez (con lo cual se desvirtúa el propósito del ejercicio). La alternativa de proporcionar la especificación por escrito reduce, a mi entender, el valor del ejercicio, puesto que restringe las posibilidades de poner en práctica el proceso de captura de requisitos tal como se describe en los textos básicos de la materia (en términos de entrevistas con el cliente, iteraciones sucesivas, negociación a partir de prototipos...[9,12]).

3. Formato de la propuesta

La presente propuesta pretende atacar estos problemas y se ha implementado este año en la asignatura de Ingeniería del Software, impartida en la titulación superior de Ingeniería Informática de la Facultad de Informática de la Universidad Complutense de Madrid. Esta asignatura forma parte del segundo ciclo, y concentra todos los contenidos del plan de estudios sobre el tema de ingeniería del software. Se trata de una asignatura de 18 créditos (12 teóricos y 6 prácticos). Los alumnos que la cursan ya han estudiado estructuras de datos, metodología y tecnología de la programación, y programación orientada a objetos.

La innovación de la propuesta radica en una manera específica de coordinar la teoría y la práctica, y en un planteamiento de la parte práctica de la asignatura orientado a dar respuesta a los problemas que se han reseñado. El orden en que se va cubriendo la parte teórica se ha definido específicamente para: garantizar que los alumnos pueden empezar a trabajar en la parte práctica en los primeros momentos del curso, y proporcionar a cada paso del desarrollo práctico aquel material teórico que es de aplicación directa en el punto alcanzado. Este paralelismo se mantiene durante los primeros meses, cuando los alumnos están absorbiendo los conceptos teóricos que necesitan para trabajar en la práctica, y se relaja ligeramente hacia el final del curso, cuando los alumnos ya están desarrollando, tienen experiencia en el proceso de desarrollo, y toleran mejor el planteamiento de alternativas, generalizaciones, o discusiones más teóricas. En este punto la teoría se desvincula de la práctica para cubrir el resto del temario contenido en el plan de estudios. La distribución temporal propuesta aparece en la figura 1. A continuación detallamos aquellas características de la propuesta que pretenden resolver los problemas descritos.

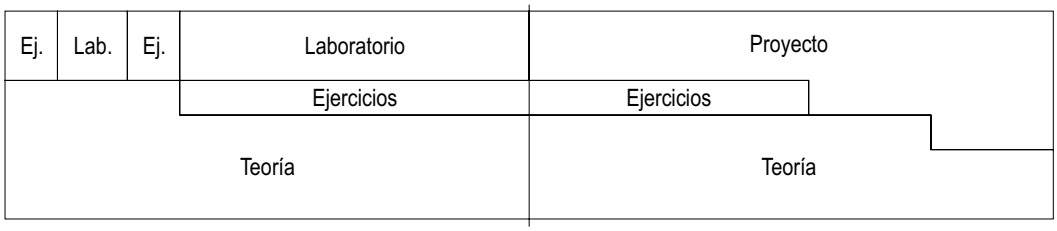


Figura 1. Distribución temporal

3.1 Parte teórica

Para hacer posible una solución del tipo que se propone, es necesario proporcionar a los alumnos directrices muy claras sobre la manera de llevar a cabo este tipo de procesos. Se ha optado por recurrir al Proceso Unificado de Desarrollo [8] de Rational como marco teórico en el que encuadrar la propuesta. Esto supone que el proceso de desarrollo a realizar en la parte práctica tendrá lugar conforme a un *plan de fase* en el que se recorran tres fases distintas: de *inicio*, en la que se reúnen los requisitos y se planifica el proceso a grandes rasgos, de *elaboración,* en la que se establece la arquitectura y se diseña el sistema, y de *construcción*, en la que se da forma concreta al sistema[1]. La cuarta fase (de *transición*) descrita en la teoría, más relacionada con la instalación de la aplicación, su depuración final y mantenimiento se obviará para hacer frente a las restricciones temporales. Cada fase tendrá una serie de *iteraciones*, cada una de las cuales dará lugar a una *construcción* del sistema. En cada iteración, según el punto del plan de fase en que se encuentre, se han de incluir proporciones variables de los siguientes flujos de trabajo: captura de requisitos, análisis, diseño, implementación y prueba (más de los primeros en las iteraciones correspondientes a la fase de elaboración, más de los últimos en las correspondientes a la fase de construcción). Además se han de aplicar las siguientes tareas de gestión de proyectos: estimación, planificación, gestión de configuración, y gestión de calidad.

El orden de exposición de los distintos contenidos teóricos mencionados se defiende más adelante en relación con la necesidad de coordinarlo con el trabajo práctico a desarrollar.

[1] Estas descripciones de cada una de las fases son a grandes rasgos y no pretenden ser exhaustivas. Para más datos ver [7,8].

Es irreal suponer que desde el primer momento los alumnos van a ser capaces de aplicar con éxito en la práctica un modelo de proceso tan detallado, casi al mismo tiempo que están descubriendo ese modelo en las exposiciones teóricas. Sin embargo se considera interesante que los alumnos tengan la experiencia de gestionar su propio trabajo. Para resolver este dilema se ha optado por establecer una distinción entre los mecanismos de gestión aplicados en las distintas partes del curso. Durante el primer cuatrimestre los alumnos trabajan en grupos pequeños gestionados por el profesor, que impone fechas de entregas, orden de trabajo, métodos a seguir...; durante el segundo cuatrimestre trabajan en grupos grandes que se autogestionan.

3.2 Parte práctica

La parte práctica que se propone consta de dos partes diferenciadas: una componente de desarrollo a pequeña escala (prácticas en equipos de 2 a 3 personas) y una componente desarrollo a gran escala (proyecto en equipos de 20 personas). La parte de desarrollo a pequeña escala se lleva a cabo durante el primer cuatrimestre (gestionada por el profesor en lo relativo a tareas a realizar, plazos de entrega...), y la parte de proyecto se lleva a cabo durante el segundo cuatrimestre (gestionada ya por los propios alumnos). Con esta división se consigue dar cabida a los beneficios discutidos en el apartado 2.2 para cada una de las alternativas (pequeña escala y gran escala). Las dos se desarrollan siguiendo el Proceso Unificado de Desarrollo.

Para evitar los problemas de restricciones temporales descritos en el apartado 2.3, se sugiere el uso de un modelo de proceso iterativo e incremental, que permita detener el proceso en el momento en que termine el primer cuatrimestre, quedando la aplicación a desarrollar en el punto en que estuviera al terminar la iteración correspondiente.

Se impone la restricción de que cada proyecto durante el segundo cuatrimestre debe desarrollarse a partir de una de las especificaciones que hayan resultado de las prácticas del primer cuatrimestre.

Esta restricción ya hace cobrar importancia a la documentación generada durante las prácticas del primer cuatrimestre, puesto que ha de servir más adelante para que un grupo mayor de alumnos base en ella su proyecto del segundo cuatrimestre. A pesar de eso se ha preferido insistir en potenciar la importancia de la documentación explícita, por considerar que la posibilidad de que una práctica concreta no fuese elegida para ser continuada podría dar excusas para no tomarse en serio su documentación adecuada. Para garantizar que todas las prácticas reciben la atención adecuada, se impone un mecanismo de rotación periódica de aplicaciones que se describe más adelante.

3.3 Prácticas a pequeña escala con intercambio

El orden que se propone inicialmente para estas etapas es: desarrollo de una propuesta, gestión de configuración, estimación de esfuerzo para la propuesta desarrollada, captura de requisitos (diagrama de casos de uso y requisitos no funcionales), planificación, y análisis (diagramas de clases y de secuencia).

Al final de cada etapa, cada equipo propone el material que ha desarrollado (junto con el que se recibiera al principio de la etapa como resultado de etapas anteriores) a otro equipo para que lo desarrolle en la etapa siguiente. Este intercambio se desarrolla por orden de listado de los equipos para evitar problemas. De este modo, cada equipo participa en la elaboración de material propio correspondiente a todas las etapas, y a la vez adquiere la experiencia de trabajar sobre material ajeno. A cada paso deberán tener en cuenta toda la documentación generada por los distintos grupos que hayan trabajado sobre el problema. Este proceso expone a los alumnos a la utilidad práctica real de los documentos que se les van exigiendo en cada etapa.

Como mecanismo fundamental de intercambio se propone el uso de una página web donde se encuentra disponible el material correspondiente a cada proyecto. Cada grupo actualizará al final de cada etapa el material disponible para el proyecto sobre el que ha estado trabajando. Este mecanismo de intercambio sirve adicionalmente como primera experiencia de los alumnos en temas de gestión de configuración.

A continuación se presentan detalles breves del tipo de trabajo involucrado en cada etapa y las ventajas que supone en cada una la obligación de intercambiar material:

Redacción de una propuesta de proyecto: En grupos de 3 personas, los alumnos deben establecer el contexto de gestión de un proyecto imaginario, y redactar la propuesta de proyecto. Cada grupo establece un documento preliminar sobre el proyecto a acometer, incluyendo básicamente: objetivos, usuarios, funcionalidad, requisitos de hardware, posible división en módulos. Aunque cada equipo trabaja con la participación exclusiva de los desarrolladores, mediante este ejercicio se consiguen simular muchas de las condiciones reales que se dan en una primera reunión con un cliente: tiempo limitado, imprecisión inicial muy grande respecto a los objetivos, diferencias de opinión entre los participantes, presión por conseguir un documento (por básico que sea) en el que se detalle un primer esbozo, desconocimiento del alcance real del proyecto por no haberse recopilado todavía formalmente los requisitos...

Gestión de configuración: Desde el momento en que un grupo necesita proporcionar material a otro aparecen ya problemas de formato de materiales, de protocolos de elección de nombres... Esto proporciona una ocasión para explicar la teoría relativa a esta parte de la gestión de configuración. Asimismo se introduce ya el concepto de modificación de un documento recibido como definitivo (raro es el grupo al que no le gustaría hacer modificaciones a la especificación que recibe, sea para hacerla más atractiva o más fácil de obedecer), y del proceso a seguir para anotarlo y documentarlo.

Estimación: Se pretende obtener una idea de cuánto dinero y cuánto esfuerzo (cuánta gente durante cuánto tiempo) pensamos que va a requerir el proyecto que se propone. Aunque no se disponga de datos a estas alturas para llevar a cabo una estimación válida, se trabaja en condiciones similares a las que se darían en una situación real.

Requisitos: Deben realizarse la identificación de actores y casos de uso, la representación de los mismos en UML, y la redacción de descripciones de flujos de eventos de un caso de uso. Se utiliza

una herramienta software para facilitar la representación.

Planificación: En el momento en que hay una serie de casos de uso y una estimación de esfuerzo disponibles puede empezarse a planificar. El intercambio de material supone una dificultad añadida tan solo en el sentido de que cada equipo planifica, no su propio trabajo futuro, sino el del equipo que va a heredar el material que está preparando. Las restricciones de personal correspondientes deben ser tenidas en cuenta. Se utiliza una herramienta sofware para la planificación de proyectos.

Análisis: Se elaboran diagramas de clases y de interacción a partir de los diagramas de casos de uso dados (también con ayuda de herramienta software). Cada grupo procesa diagramas de casos de uso sobre proyectos que no ha visto hasta ese momento, para enfatizar que toda la información necesaria para el desarrollo debe haberse detallado durante la especificación (sea en los diagramas o en las descripciones textuales en forma de requisitos adicionales).

Aunque sería posible (e incluso deseable) continuar con este patrón de trabajo hasta completar un recorrido completo de los flujos de trabajo de desarrollo (diseñando, implementando y probando los proyectos), nuestra experiencia hasta la fecha es que el proceso de compaginar la exposición teórica y la puesta en práctica reduce el ritmo al que se puede cubrir la materia, y, teniendo en cuenta que se ha de partir de un cierto marco teórico imprescindible antes de poder empezar el trabajo práctico, que el mecanismo de intercambio requiere un proceso de "digestión" de la documentación recibida antes de que el grupo pueda empezar a producir, y que a cada paso los alumnos deben aprender a dominar las técnicas nuevas que se van introduciendo, no es realista aspirar a cubrir más etapas prácticas en un solo cuatrimestre. Por otro lado esto tiene la ventaja de poder dedicar las últimas semanas de exposición teórica del cuatrimestre a proporcionar a los alumnos los conceptos básicos sobre diseño (y en menor medida gestión de calidad) que van a necesitar para poner en marcha con buen pie el proyecto a gran escala en el segundo cuatrimestre.

3.4. Proyecto a gran escala

Al término del primer cuatrimestre, las prácticas realizadas han dejado como resultado una página web con una serie de proyectos con propuestas, estimación, especificación, planificación, y modelo de análisis. De todo este material, el correspondiente a las primeras etapas de las prácticas ha sido revisado por varios equipos, y cada uno ha generado no sólo versiones mejoradas sino documentos adicionales de justificación de las modificaciones. Todo este material se aprovecha como punto de partida para los proyectos a gran escala a desarrollar en el segundo cuatrimestre, paliando así parcialmente tanto las restricciones temporales que supondría el empezar un proceso de desarrollo a gran escala cuando ya solo quedan tres meses de tiempo útil, como la disponibilidad de especificaciones.

Todos los alumnos han tenido ocasión de familiarizarse con la elaboración de los distintos materiales. Aunque cada grupo ha realizado un ejemplo de planificación, el trabajo de todos los grupos hasta ese punto ha sido planificado por el profesor, en el sentido de que ha impuesto las tareas a realizar (elaboración de los distintos materiales) y los hitos (fechas de entrega de cada etapa). Asimismo, los alumnos han operado bajo un mecanismo básico de gestión de configuración (propuesto por el profesor) en forma de página web.

A partir de este punto se establecen grupos mucho más grandes (en torno a 20 personas por grupo) para continuar el trabajo práctico durante el resto del curso. Cada uno de estos grupos tiene la responsabilidad de gestionar el desarrollo de su proyecto. Esto supone tanto la planificación del trabajo, como la gestión de configuración.

El único requisito impuesto por el profesor es que se debe planificar el trabajo en tres iteraciones (una por cada mes restante de curso), y que al terminar cada iteración deben revisarse no solo los productos concretos obtenidos sino los procedimientos empleados para conseguirlos.

La exposición teórica se desvincula del desarrollo concreto de cada proyecto, pero sigue estando en las primeras etapas orientada a proporcionar conceptos necesarios.

Una de las dificultades principales que encuentran los alumnos en este punto es la falta de criterios para elegir entre los distintos proyectos propuestos. Puesto que tienen que elegir sobre todo en función de la documentación que hay generada, esto supone un trabajo serio de revisar la documentación disponible, y esto se aprovecha

para impartir la teoría relativa a la calidad del software y los proceso de gestión de calidad en el desarrollo. Estos conceptos se presentan ligados a la planificación en tres iteraciones y a la necesidad de revisar los procedimientos de trabajo en cada iteración, con objeto de mejorar la calidad.

Para hacer esto posible se presenta el concepto de *revisión formal técnica* (*Formal Technical Review* o FTR) como el principal método de validación de la calidad de un proceso o de un producto. Un grupo debe examinar parte o todos los resultados de su trabajo (planes, especificaciones, modelos, diseños, código, casos de pruebas, documentación,...) para buscar problemas potenciales. Se insiste en que este proceso tiene ventajas adicionales a la hora de garantizar que todos los miembros del grupo, por haber participado en revisiones del trabajo de sus compañeros, estarán familiarizados con él y podrán hacerse cargo de él en caso de ausencia, abandono, o enfermedad del compañero.

Así mismo, se explica el concepto de *métrica* como medida específica que se toma de un proceso para poder comprobar objetivamente si está funcionando como es debido. Se exige a los alumnos que en cada iteración diseñen las métricas que consideren necesarias para evaluar sus procedimientos de trabajo y recopilen los datos pertinentes. Estos datos se utilizarán en las revisiones técnicas formales de los procedimientos establecidos que se han de realizar al principio de cada nueva iteración.

Finalmente se introduce el concepto de sesión de evaluación de un proyecto, en la que cada grupo debe hacer un resumen de la situación del proyecto en el que trabaja, incluyendo los procedimientos que están empleando, los problemas que han tenido, el ajuste a la planificación inicial, y datos objetivos que tengan disponible sobre su rendimiento hasta la fecha. Este tipo de sesiones cumplen un objetivo múltiple. Por un lado, permiten a cada grupo enterarse de los problemas y soluciones que está experimentando los demás, maximizando el número de experiencias concretas a las que se ve expuesto cada alumno, aunque sea indirectamente. Por otro lado, obligan a los alumnos tanto a realizar un seguimiento del desarrollo del proyecto como a plantearse la manera en que el desarrollo observado puede presentarse a terceras personas de manera coherente e informativa. Por último, permite al profesor mantener un seguimiento y evaluación de la labor de los alumnos. Esto garantiza que se pueda detectar y remediar a tiempo posibles errores de concepto (que constituyen un riesgo importante cuando se está dejando la labor de gestión a los alumnos), o problemas serios en la interacción personal dentro de los grupos. Al mismo tiempo, simulan en cierta medida la necesidad real en el mundo de la empresa de rendir cuentas sobre el desarrollo de proyectos a la superioridad.

5. Valoración y conclusiones

Los problemas que se han planteado como objetivos son conocidos en la literatura sobre enseñanza de ingeniería del software [10,5,4]. Las distintas soluciones que se han propuesto históricamente han ido dando forma a los planes de estudios de ingeniería del software durante los últimos años, incluyendo cada vez más habitualmente una componente de desarrollo de proyectos. Aún así, es bastante habitual el fragmentar la docencia de la materia en dos partes disjuntas: una que concentra la materia relacionada con la gestión de proyectos, y otra centrada en la especificación y diseño de software. La presente propuesta está diseñada considerando que el proceso de desarrollo es una tarea compleja cuyas máximas dificultades residen no tanto en las subtareas concretas en que se puede subdividir sino precisamente en el proceso de división del trabajo original y de posterior integración de los resultados obtenidos. En este sentido puede entenderse la labor de especificación y diseño como una división de la funcionalidad (caso de uso) y la arquitectura (módulos) de una aplicación, y la labor de gestión como una división del trabajo en tareas que se asigna a distintos momentos y distintos miembros del equipo. En ambos casos el desafío mayor no es el establecer una división inicial, sino el integrar los resultados. Los problemas de integración se multiplican cuando además de integrar a ese nivel hay que integrar la división de gestión con la división de especificación y diseño. En esta propuesta se pretende integrar desde el primer momento todo lo posible estas subtareas, y facilitar al alumno un ejemplo de cómo todas las tareas que se le proponen interactúan en un proceso de desarrollo real.

Existen propuestas recientes [3,13] que pretenden dar solución a los mismos problemas. En [3] se propone asignar a cada equipo (de 3 personas) la gestión de proyectos y un presupuesto de puntos de evaluación que han de utilizar para contratar programadores de entre los miembros de otros grupos. En [13] se proponen una serie de 'trucos sucios' que se emplean para dificultar la tarea de los alumnos con objeto de que el proceso de desarrollo sea más afín a la experiencia real que han de sufrir luego. La presente propuesta proporciona ventajas similares sin necesidad de imponer criterios mercantilistas ni de que el profesor se presente como antagonista del alumno.

Esta propuesta es un refinamiento de otra anterior [6] en la que se proponía utilizar el modelo de madurez de la capacidad de software (SW CMM) [2] como referencia para que los alumnos intentaran a lo largo del curso constituir una 'empresa' de desarrollo con sus propios procedimientos de trabajo documentados. Se proponía intentar alcanzar el nivel 3 (cada grupo debía en el proyecto definir su propio proceso). La experiencia resultó positiva por cuanto que los alumnos desarrollaron por escrito versiones muy razonadas de los procedimientos de trabajo, pero no quedó claro hasta qué punto habían conseguido aprender de verdad a implantarlos en la práctica. En la propuesta de este año se ha introducido la idea de que la primera parte del proceso es gestionado por el profesor, lo que libera a los alumnos durante ese periodo para que puedan centrarse en dominar las técnicas. En términos prácticos se está tomando como referencia el nivel 2 (la asignatura, entendida como una colección de proyectos bajo la misma gestión general - la del profesor - sigue un proceso más o menos establecido - el que se ha discutido en teoría - pero que no está definido específicamente en ningún sitio - no hay manual de gestión de proyectos para la asignatura). Hasta la fecha (mediados de mayo) se vienen cumpliendo las previsiones. Está por ver si los resultados académicos ratifican la validez pedagógica del método.

Referencias

[1] Frederick P. Brooks, Jr. *The Mythical Man-Month.* Addison-Wesley, 1995 (1ªed. 1975).

[2] Carnegie Mellon University, SEI. *The Capability Maturity Model: Guidelines for Improving the Software Process*. Addison-Wesley Publishing Company, Reading, MA, 1995.

[3] Ray Dawson, Twelve Dirty Tricks to Train Software Engineers, ICSE 200, ACM.

[4] Joint IEEE/ACM Task Force. *Computing Curricula 2001. A Vision of the Overview Volume*, 2001, Steelman Draft

[5] Ford, G. 1991 SEI Report in Graduate Software Engineering Education, SEI, Carnegie Mellon University, Pittsburgh, Tech. Rep. CMU/SEI-91-TR-2

[6] P. Gervás, M.A. Gómez, A. Sarasa. "Ingeniería del software: ¿basta con desarrollar proyectos o haría falta probar a implantar procesos de desarrollo a largo plazo?", JENUI 2001.

[7] P.Kruchten. *The Rational Unified Process. An Introduction. Second Edition*. Addison - Wesley, 2000.

[8] I. Jacobson, G. Booch, J. Rumbaugh. *UML. El proceso unificado de desarrollo de software*. Ed. Addison Wesley Iberoamericana, 2000.

[9] R.S.Pressman. *Software Engineering: A Practitioner's Approach*. European adaptation, 5th edition. McGraw-Hill, 2000.

[10] Software Engineering Coordinating Committee (SWECC). *Guide to the Software Engineering Body of Knowledge*. http://www.swebok.org/

[12] I.Sommerville. Software Engineering. 6th edition. Addison-Wesley, 2001.

[13] K. Todd Stevens, Experiences Teaching Software Engineering for the First Time, ITiCSE 2001, ACM.

Inteligencia Artificial

SMIT: diseño e implementación de un agente sintético de presentación para las Unidades de Soporte a la Docencia del PLAN-G

Maria Aguilar, Clara Inés Peña, Ramón Fabregat
Institut d'Informàtica i Aplicacions (IIiA)
Universitat de Girona (UdG)
maguilar@eia.udg.es,clarenes@eia.udg.es, ramon.fabregat@udg.es

Resumen

Los agentes de presentación, o agentes sintéticos, son una herramienta muy potente para guiar y ayudar a los usuarios en su interacción con plataformas informáticas de gran tamaño o complicadas. En este artículo se presenta el agente SMIT (Synthetic Multimedia Interactive Tutor), un agente sintético que representa los mensajes dirigidos al estudiante en un sistema de tutoría inteligente, adoptando diferentes estilos antropomórficos. Este agente forma parte de los agentes de interfaz del sistema multiagente MAS-PLANG (MultiAgent System PLANG) diseñado para ofrecer características de adaptatividad a la plataforma USD (Unitats de Suport a la Docència) utilizada para dar soporte a la enseñanza-aprendizaje a través del web.

1. Introducción

Los agentes sintéticos se han desarrollado para animar interfaces de usuario utilizando personajes reales o de aspecto real. En entornos educativos virtuales, la concepción de estos agentes con apariencia agradable y cautivadora, provoca en el estudiante la ilusión de estar siempre asistido durante su proceso de aprendizaje, lo cual según los expertos, favorece ampliamente la adquisición del conocimiento.

Actualmente con la proliferación de las plataformas para generar agentes, se han desarrollado o utilizado diversos tipos de agentes sintéticos en entornos para el comercio electrónico y para la educación a distancia a través del web. Por ejemplo, podriamos citar los personajes introducidos por [3] para permitir diálogos animados de ventas de productos o el PPP persona propuesto por [4] para mostrar, explicar o comentar presentaciones gráficas de materiales educativos.

Para transformar el entorno educativo de las USD [13] desarrollado bajo el proyecto PLANG[1], en un sistema adaptativo teniendo en cuenta estilos de aprendizaje, se está construyendo el sistema multiagente MAS-PLANG [2], de cuya arquitectura forma parte el agente SMIT [9], un agente sintético de presentación que asume diferentes estilos antropomórficos para representar los mensajes dirigidos al estudiante. El objetivo de este documento es presentar en detalle los aspectos básicos de su diseño e implementación.

En la sección 2 se describe en forma global el diseño pedagógico de los tutores inteligentes como marco para la introducción en la sección 3, del sistema multiagente MAS-PLANG. A continuación las secciones 4 y 5 describen el diseño y la implementación del agente SMIT y la sección 6 finalmente concluye el documento con un resumen y aspectos del trabajo futuro.

[1] PLAN-G: PLAtaforma telemática de Nueva Generación para la enseñanza abierta y a distancia.

Este trabajo ha sido parcialmente financiado por la CICYT TEL-99-0976

2. Sobre el diseño pedagógico de tutores inteligentes

La incorporación de la informática en la tecnología educativa ha despertado desde el principio importantes expectativas sobre las mejoras que podría aportar al aprendizaje. Desde entonces ha existido una larga historia de éxitos y también de fracasos.

Arruarte [1] en sus apreciaciones teóricas incluidas en su tesis doctoral sobre generación de sistemas de educación inteligentes, ha hecho una importante descripción sobre el diseño pedagógico de los sistemas de tutoría inteligente (ITSs), la cual hemos aprovechado como base para el diseño del sistema multiagente MAS-PLANG, marco de funcionamiento del agente SMIT.

En particular, los sistemas ITS tratan de aplicar las técnicas de la Inteligencia Artificial al desarrollo de sistemas de aprendizaje asistido por computador con el propósito de construir sistemas de enseñanza inteligentes; donde la palabra *inteligente* se asocie a la capacidad de adaptación dinámica al desarrollo del aprendizaje.

Un ITS enfoca la educación como un proceso de cooperación entre el tutor y el alumno. En general este proceso esta guiado por el tutor, el cual ha de analizar el comportamiento del alumno tanto para saber cual es su nivel de conocimiento como para satisfacer sus requerimientos y así poder determinar y aplicar en todo momento las estrategias educativas que se consideren mas adecuadas. También permite guiar al alumno en su aprendizaje por unidades de información controladas evitando así que se pierda cuando navega a través de su contenido.

Las siguientes cuatro componentes caracterízan un sistema de tutoría inteligente:

- El *Módulo del dominio*. Que constituye el conocimiento de la materia a enseñar. Este se utiliza para determinar lo que se debe presentar al estudiante y la forma de evaluarlo.
- El *Módulo pedagógico*. En el que se representan las diferentes estrategias de enseñanza y se implementan los métodos de control de la sesión mediante la elección y sequencialización adecuada de dichas estrategias. Debe ser capaz de decidir, utilizando la información tanto del *Módulo del dominio* como del *modelo del estudiante*, que conceptos presentar en cada momento, cómo presentarlos y cuando y como interrumpir al estudiante.
- El *Modelo del estudiante*: Que representa la imagen que el sistema tiene del conocimiento que el estudiante haya adquirido durante el proceso de instrucción. También incorpora datos sobre su comportamiento que puedan tener repercusión en su forma de aprender.
- El *Módulo de diálogo*: Que define la interfaz de comunicación del sistema con el usuario. Esta componente se encarga de traducir las intervenciones del sistema de forma inteligible para el alumno y transformar las entradas de este a una representación interna que el sistema pueda procesar.

El agente SMIT en nuestro caso, se ubica en el *módulo de diálogo* del entorno virtual de aprendizaje de las USD para presentar al estudiante, por medio de metáforas antropomórficas agradables, los mensajes generados por los otros agentes del sistema (actualmente sólo representa los mensajes provenientes del agente programable SONIA [14] de la arquitectura del MAS-PLANG propuesta en la Figura 1) durante una sesión de aprendizaje.

El poder guiar al estudiante durante su proceso de aprendizaje es una característica muy importante a tener en cuenta en el diseño de tutores inteligentes y debe estar orientada no solo a permitir ver o trabajar con los contenidos didácticos considerando el nivel de conocimiento del estudiante o su estilo de aprendizaje como en nuestro caso, sino también a ofrecer un entorno de trabajo agradable y adaptado a las preferencias del estudiante. En este sentido, Opperman [12] realiza una muy buena clasificación de hacia donde dirigir las tendencias cuando se diseñan este tipo de sistemas:

- Los sistemas de tutoría inteligente pueden ser *adaptables* o configurables por el estudiante. En este caso, es el estudiante quien suministra explícitamente las características particulares que desea tener en su entorno de trabajo. Dependiendo por supuesto de la flexibilidad de cada entorno, esas preferencias podrían tender por ejemplo, a poder configurar los tipos de iconos

disponibles, la posición y el tamaño de las ventanas, las herramientas de navegación, etc.

- Los sistemas de tutoría inteligente pueden ser *adaptativos*. En este caso, es el sistema que mediante procesos especiales (técnicas de inteligencia artificial), percibe las características de aprendizaje del estudiante y le ofrece dinámicamente un entorno educativo personalizado.

El sistema multiagente MAS-PLANG sobre el que hablaremos en la siguiente sección, transforma el sistema hipermedia educativo configurable USD en un sistema hipermedia educativo adaptativo, teniendo en cuenta el estilo de aprendizaje del estudiante que lo utiliza.

3.- La arquitectura del MAS-PLANG

El sistema MAS-PLANG forma parte de las investigaciones desarrolladas por el Grupo de Comunicaciones y Sistemas Distribuidos del Instituto de Informática y Aplicaciones de la Universitat de Girona y se planteó para ofrecer características de adaptatividad a las Unidades de Soporte a la Docencia USD a través de la implementación de procesos para:

- Dirigir, controlar y coordinar la interactividad del estudiante con el sistema y los contenidos didácticos.
- Crear y mantener el *modelo de estudiante*.
- Filtrar la información proveniente de las bases de datos de acuerdo con algunos patrones preestablecidos.
- Evaluar el conocimiento del estudiante con base en su perfil de aprendizaje.

Para cumplir con estos objetivos, se diseñó un sistema multiagente utilizando agentes inteligentes con propiedades de sociabilidad para interactuar y colaborar con otros agentes y propiedades de adaptabilidad para según el caso permitir ser programados por el estudiante cuando ciertas tareas sean importantes para complementar algunas de las actividades de aprendizaje propuestas.

El sistema está compuesto por dos niveles de agentes: los agentes del nivel superior o *asistentes personales* denominados PDAs y los agentes del nivel inferior o *agentes de información* denominados IAs. En la siguiente figura se puede observar dicha arquitectura.

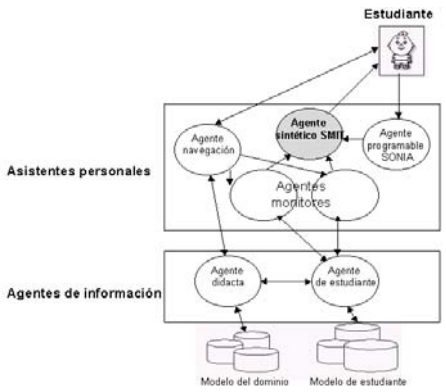


Figura 1.Arquitectura del MAS-PLANG

Los asistentes personales se ejecutan de forma concurrente con las aplicaciones del cliente que permiten el acceso al sistema, capturan las acciones del estudiante durante las actividades de una sesión de aprendizaje y le asisten con presentaciones amenas en situaciones particulares. Según su objetivo existen los siguientes:

- *El agente programable SONIA*: que permite al estudiante automatizar ciertas tareas de aprendizaje mediante la programación de determinados eventos.
- *Los agentes monitores*: que monitorizan las actividades del estudiante para generar retroalimentación a su *módulo de comportamiento.*
- *El agente Sintético SMIT*: que se introduce de forma animada en la interfaz para presentar al estudiante los mensajes provenientes de otros agentes.
- *El agente de navegación*: que organiza los caminos de navegación mediante la interacción directa con la base de datos y los agentes didacta y de usuario.

Los agentes de información actúan como intermediarios entre los agentes del nivel superior (PDAs) y el *módulo del dominio*, el *módulo de*

aprendizaje, y el *modelo del estudiante*. Estos se clasifican en:

- *El agente modelo del estudiante*: que se encarga de generar y mantener el modelo del estudiante (recibe información de los agentes monitores, aplica patrones de clasificación y categoriza los estudiantes, actualiza las bases de datos, etc.).
- *El agente didacta*: que selecciona las estrategias pedagógicas idóneas para la organización de la sesión de aprendizaje, basándose en el *modelo del estudiante*.

4. Diseño del Agente SMIT

Como su misión es la de mostrar de manera agradable al estudiante los mensajes provenientes de los otros agentes del sistema, SMIT adopta un estilo antropomórfico apropiado cada vez que percibe los mensajes dirigidos al estudiante para presentar su contenido de forma rápida y clara.

Para el diseño de la estructura de comunicación entre los agentes de interfaz de la arquitectura MAS-PLANG y el agente SMIT, hemos adoptado el método del tablón de anuncios o de la pizarra propuesto por [7], que consiste en reservar un espacio de memoria conocido por todos los agentes del sistema, para que puedan dejar allí los mensajes dirigidos al estudiante. SMIT entonces estará siempre atento a leer el contenido de la pizarra para realizar las acciones correspondientes con los mensajes encontrados. La Figura 2 esquematiza este proceso.

A continuación se puede observar la estructura de los mensajes dejados en la pizarra por los agentes de interfaz del sistema:

- *Tipo*: indica el tipo de mensaje. Por medio de este campo se decide el tipo de presentación que ha de efectuar el compositor.
- *Prioridad*: indica el grado de urgencia. Este campo permite ordenar los mensajes de la pizarra.
- *Remite*: identifica el agente que envía el mensaje. Esta información es necesaria para identificar al cerrar la sesión, el agente receptor de los registros de eventos sucedidos con respecto al tratamiento dado al mensaje puesto en la pizarra.
- *Contenido*: el texto del mensaje

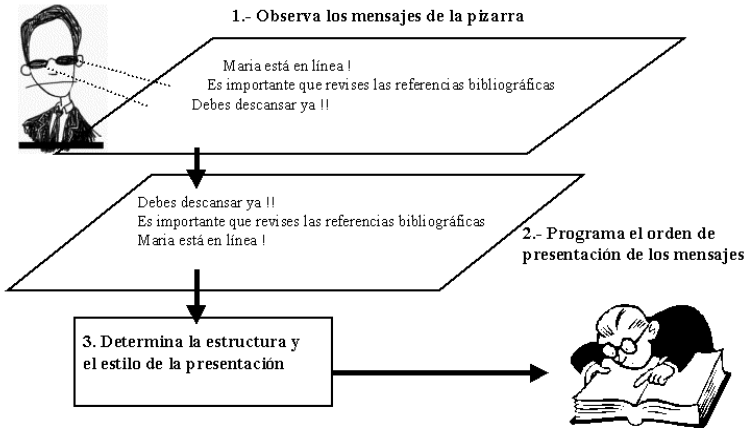


Figura 2. Estructura de las acciones realizadas por el agente SMIT

La presentación de un mensaje se construye con la siguiente secuencia:

- En un principio aparece el agente SMIT hablando (gesticulando, con movimientos sincronizados de la boca), mostrando el mensaje correspondiente a través de una interfaz como la que se muestra en la Figura 3. En este punto se puede interactuar con el agente mediante tres botones: se puede activar o desactivar el sonido de la presentación (que por defecto aparece con sonido para llamar la atención del alumno), se puede ver el historial de mensajes recibidos, o se puede aceptar el mensaje y cerrar la ventana.
- Si el alumno no acepta el mensaje en un tiempo determinado, SMIT se duerme, como se muestra en la Figura 4. Cuando está dormido el mensaje no se puede ver, pero se puede pedir su reproducción haciendo un clic en el botón de repetir mensaje. Cuando SMIT duerme también ronca. El hecho de que se duerma sirve tanto para indicar al usuario que el mensaje ya es viejo, como para llamar su atención al hacer ruido.
- Según el tipo de mensaje que presente, el agente SMIT puede aparecer con diferentes estados de ánimo, o moviendo de forma diferente la boca o los brazos. Si el mensaje mostrado además tiene una característica especial, puede aparecer un icono al lado de su contenido para resaltar dicha condición.
- También hay que tener en cuenta lo que sucede cuando llega un mensaje nuevo y aún no se ha aceptado el anterior. En ese caso se trabaja con una cola de mensajes en la misma ventana del agente para evitar equívocos y aparecen unas flechas que permiten el movimiento entre los mensajes existentes, para poder visualizar sus contenidos. Los mensajes van desapareciendo de la cola a medida que el usuario los va aceptando, pero el alumno siempre podrá consultar los mensajes que han ido llegando haciendo un clic en el botón del historial.

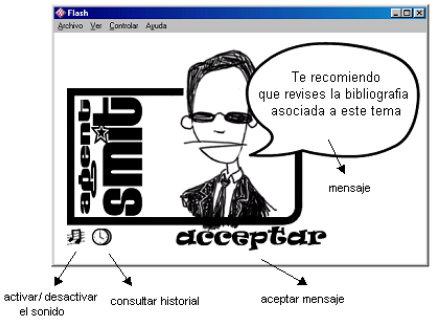


Figura 3. Interfaz del agente SMIT

Figura 4.- El agente SMIT dormido

La base de conocimiento del agente consiste en una tabla que guarda las características de los mensajes que pueden llegar a la pizarra y las instrucciones sobre como mostrarlos. Por ejemplo, el campo *tipo de mensaje* permite decidir la estructura que va a tener la representación del mensaje al estudiante. Es decir, es un campo clave que asigna el "estado de ánimo del agente" (alegre, serio, sorprendido) cuando muestra un mensaje y ese estado de ánimo repercute en un campo que indica el tipo de película en FLASH que se utilizará para la animación del personaje. Dependiendo del tipo de mensaje, también se manejan de una u otra forma sus contenidos textuales y en el idioma seleccionado por el estudiante en el entorno de aprendizaje

5. Implementación del agente SMIT

5.1. Lenguajes de programación utilizados

El sistema MAS-PLANG se ha desarrollado sobre la plataforma FIPA-OS [11] compatible con los estándares propuestos por FIPA [5] para la generación de agentes.

Como el lenguaje de programación básico de FIPA-OS es el Java, hemos utilizado para SMIT la arquitectura cliente-servidor, de tal manera que un *servlet* (SMIT servidor) realice los procesos de registro e integración del agente con los agentes de FIPA-OS y del MAS-PLANG (utilizando ACL [6] para el envio y recepción de mensajes) del lado del servidor y libere (a través de un *socket)* un *applet* (SMIT cliente) con la información necesaria (base de conocimiento) para el funcionamiento del agente del lado del cliente (interacción con el usuario).

En la máquina cliente, el *applet* para cumplir con su objetivo de representar los mensajes provenientes de los otros agentes del sistema y dirigidos al estudiante, realiza procesos de interacción directa con el motor de las presentaciones mediante sentencias escritas en JavaScript, que es el lenguaje básico de programación del entorno de aprendizaje de las USD. Para la comunicación entre los lenguajes Java y JavaScript hemos utilizado las clases LiveConnect desarrolladas por Netscape [10].

El motor de las presentaciones se desarrolló a través del programa Macromedia Flash [8] por su flexibilidad para la generación de presentaciones multimedia en animaciones para el web y porque es un programa que se basa en la reproducción correlativa de un conjunto de imágenes (*frames)* a un ritmo de 12 por segundo (o a un ritmo configurable) y permite mediante la programación en su lenguaje propio ActionScript, incluir interactividad a las animaciones y comunicarse con la programación en JavaScript del entorno de aprendizaje.

En la Figura 5 se puede observar la distribución de los diferentes módulos que conforman la programación del agente SMIT y sus interacciones con el módulo generador de las presentaciones, el cual recibe programación en JavaScript proveniente de la pizarra y mediante dos módulos escritos en HTML (con sentencias en JavaScript y Flash) decide la forma de representar los mensajes al estudiante.

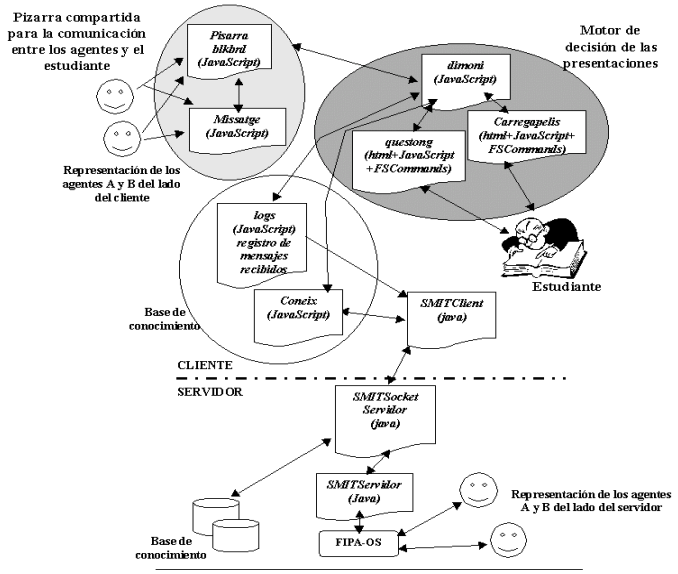


Figura 5. Esquema de la programación del agente SMIT

5.2. Funcionamiento general

El funcionamiento general de SMIT integrando cliente y servidor con la plataforma FIPA-OS se puede apreciar en la Figura 6. Cuando un alumno se conecta a la plataforma se realizan los siguientes procesos:

- Se genera el SMIT Servidor registrándose en FIPA-OS y recibiendo el conocimiento que le indica la forma de generar las presentaciones (qué estilo antropomórfico adoptar y qué movimientos realizar) (pasos 1 y 2).
- Se libera el *applet* SMIT cliente con la información de la base de conocimiento transferida por el SMIT servidor (paso 3).
- SMIT supervisa el estado de la pizarra atento a los mensajes dirigidos al estudiante que hayan dejado los demás agentes del sistema (paso 4).
- El motor generador de presentaciones decide de acuerdo al mensaje percibido y a la evaluación de la base de conocimiento del agente SMIT, la forma de representar el mensaje al estudiante (seleccionando el tipo de película FLASH de acuerdo al tipo de mensaje y combinándolo con el contenido del mismo) y a su vez mantiene un registro de los eventos relacionados con el tratamiento dado a dicho mensaje (agente emisor, hora de llegada del mensaje a la pizarra, hora de salida del mensaje al estudiante, hora de aceptación del mensaje por parte del estudiante, etc) (pasos 5 a 8).
- Si el estudiante decide terminar la sesión, el SMIT cliente envía el registro de eventos de mensajes al SMIT servidor, para que éste se encargue de distribuirlo a los correspondientes agentes emisores, actualice la base de conocimiento si fuera el caso y finalmente se "des-registre" de FIPA-OS (pasos 9 y 10).

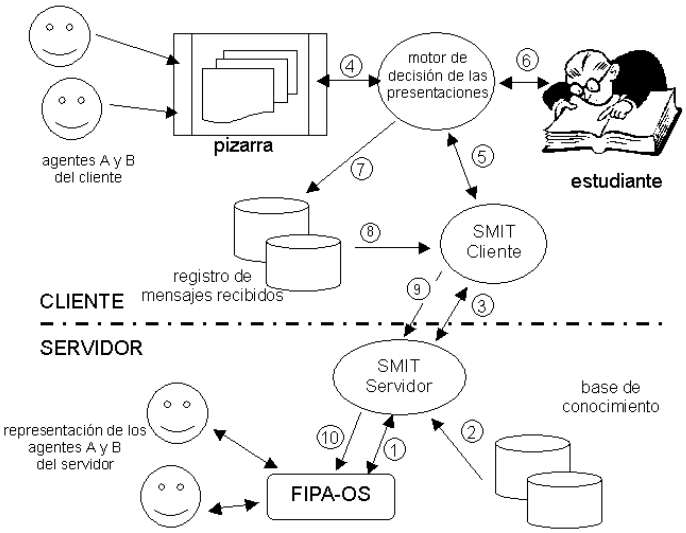


Figura 6. Funcionamiento general del agente SMIT

6. Conclusiones

Considerando que la asistencia al estudiante durante su proceso de aprendizaje es un factor primordial para facilitarle la adquisición del conocimiento, hemos generado el agente SMIT, un agente sintético que mediante estilos antropomórficos agradables, representa al estudiante los mensajes provenientes de los otros agentes del sistema integrados al entorno educativo de las Unidades de Soporte a la Docencia de la UdG.

Desde el punto de vista técnico, hemos logrado el desarrollo de un agente rápido en sus reacciones, sencillo en su uso y transparente a la aparición de otros agentes del sistema. Actualmente, probamos su funcionamiento representando los mensajes del agente programable SONIA de la arquitectura del MAS-PLANG pero pensamos poder contar en un futuro cercano, con la integración de los mensajes provenientes de los agentes monitores y del agente de navegación de esta misma arquitectura.

Referencias

[1] Ana Arruarte. *Fundamentos y diseño de IRIS: un entorno para la generación de sistemas de enseñanza inteligentes*. Tesis doctoral, Facultad de Informática, Universidad del País Vasco, San Sebastián, España, 1998.

[2] C. I. Peña, J. L. Marzo. *Adaptive Intelligent Agent Approach to Guide the Web Navigation on the PLAN-G Distance Learning Platform*, IEE Colloquium "Lost in the Web - Navigation on the Internet", London, November 1999.

[3] Elisabeth André, Thomas Rist. Presenting through Performing: on the use of multiple animated characters in knowledge-based presentation systems. German Research Center for Artificial Intelligence GmbH, Saarbrucken, Germany, 2000.

[4] Elisabeth André, Jochen Müller, Thomas Rist. *The PPP Persona: a Multipurpose Animated Presentacion Agent.* German Research Center for Artificial Intelligence GmbH, Saarbrucken, Germany, 1996.

[5] FIPA- Foundation for Intelligent Physical Agents. http://www.cselt.stet.it/fipa

[6] FIPA- Foundation for Intelligent Physical Agents. Agent Comunication Language. FIPA97 Specification, version 2.0, part 1. http://www.fipa.org

[7] Franco Zambonelli. *Coordination, Communication and Collaboration. Coordination Models and Technologies for Internet Agent*s. EASSS 2000, 2n European Agents Systems Summer School, Saarbrücken, Germany, 2000.

[8] Macromedia. http://www.macromedia.com

[9] M. Aguilar. *SMIT: Disseny i Implementació d'un Agent Sintètic de Presentació per les Unitats de Suport a la Docència del PLAN-G*, Proyecto final de carrera, Escola Politècnica Superior, Universitat de Girona, España, 2001.

[10] Netscape Developers Web Site. http://developer.netscape.com/

[11] Nortel Networks Corporation. *FIPA-OS V1.3.3 Distribution Notes*, Open Source, 2000. http://fipa-os.sourceforge.net/

[12] R. Oppermann, R. Rossen, Kinshuk, *Adaptability and Adaptivity in Learning Systems*, Knowledge Transfer (Volume II) (Ed, A. Behrooz), Pace, London, UK, 1997, pp. 173-179.

[13] R. Fabregat, J.L. Marzo, C.I. Peña, *Teaching Support Units*, Computers and Education in the 21st Century: Kluwer Academic Publishers, 2000, pp. 163-174.

[14] S. Oliveras. *Implementació d'un agent intel•ligent d'interfície per assistir a l'estudiant quan realitza feines d'aprenentatge en la plataforma telemàtica educativa del PLAN-G*, Proyecto final de carrera, Escola Politècnica Superior. Universitat de Girona, España, 2000.

Docencia de prácticas de Tratamiento Digital de Imágenes y de Visión Artificial en la Universidad de Almería

Francisco Guindos Rojas y José Antonio Piedra Fernández.

Departamento Lenguajes y Computación
Área Ciencias de la Computación e Inteligencia Artificial.
Universidad de Almería.
e-mail: fguindos@ual.es jpiedra@ual.es

Resumen

Este artículo busca facilitar el enlace entre los contenidos teóricos que se tratan en las asignaturas de Tratamiento Digital de Imágenes y Visión Artificial y el desarrollo mediante las prácticas de dichos conceptos. Presentamos un entorno [3] [4] que permita centrar al alumnado en los conceptos teóricos y que a su vez reduzca la complejidad a la hora de implementar los algoritmos propuestos en prácticas.

1. Introducción

En la Universidad de Almería se imparten las asignaturas de T.D.I. (Tratamiento Digital de Imágenes) y V.A. (Visión Artificial). En la tabla 1 se representa el carácter de las asignaturas en las titulaciones en las que se imparten.

Hablar hoy día de tratamiento de imágenes[2], implica hablar de informática. Los sistemas informáticos se han convertido en una herramienta indispensable en este campo. Desde las tareas más simples de procesado de imágenes a las más complejas de visión artificial, todas ellas se abordan hoy utilizando los medios que la informática pone en nuestras manos.

Así, en los estudios universitarios de informática, se han incluido asignaturas donde el alumno adquiere los conocimientos necesarios para el trabajo en esta área: adquisición, representación y manipulación de imágenes. Se trata de una disciplina donde la realización de prácticas es de suma importancia para su adecuado aprendizaje, pero cuya realización no es fácil de planificar, especialmente cuando se debe limitar a un espacio de tiempo reducido como es un cuatrimestre. El problema surge cuando se pretende realizar una práctica a un cierto nivel en el tratamiento de la imagen, pero para llegar hasta allí hay que realizar toda una serie de tareas previas, que si bien pueden enmarcarse dentro del objetivo de la propia asignatura, reducen el tiempo disponible para el desarrollo de la práctica deseada.

Por otra parte, la Visión Artificial [1] [5] se ha convertido en una de las áreas de trabajo de la Inteligencia Artificial con más futuro. Son muchas los campos de aplicación donde ya hoy se está utilizando con éxito aportando un gran avance en los métodos de trabajo. Utilizando técnicas de Visión Artificial se puede realizar un tratamiento *inteligente* de las imágenes recibidas en el ordenador en el que, más allá de la pura manipulación, se llega al análisis y reconocimientos de los objetos presentes en ellas.

Asignatura.	Ciclo	Tipo	Créditos
Tratamiento Digital de Imágenes	1º Ingeniería Informática	Optativa	6
Visión Artificial	2º Ingeniería Informática	Optativa	6

Tabla 1. Situación de las asignaturas

La docencia práctica no trata de un recorrido exhaustivo por todos los métodos empleados en estas tareas, sino que se ha escogido una metodología, con ejemplos y un entorno de trabajo donde llevar a la práctica cada una de las técnicas necesarias. Se trata de un entorno de trabajo realizado por los mismos autores y destinado a facilitar la realización e implementación de prácticas en asignaturas de Tratamiento Digital de Imágenes y Visión Artificial. Básicamente, es un conjunto de funciones de preprocesado, segmentación y descripción de imágenes que permiten al alumno centrarse directamente en las prácticas, liberándolo de la implementación de los prolegómenos necesarios para ellas.

Las principales ventajas con respecto a otros sistemas son:

- Utilización sencilla.
- Fácil de instalar.
- No necesita de nigún otro software para usarse (simplemente el compilador).
- Los tipos de datos (matrices e imágenes) verifican la validez de los datos (rango de los índices, valores iniciales, ...) en cada acceso, lo que aunque menos eficiente, es más adecuado para el desarrollo de las prácticas.

2. Contenido de las asignaturas

El contenido de los descriptores de las dos asignaturas a tratar se divide en los bloques temáticos que aparecen a continuación.

Tratamiento Digital de Imágenes:

- Captación, preprocesado y representación de imágenes.
- Transformadas en T.D.I.
- Algoritmos de realce, restauración y segmentación de imágenes.

Visión Artificial:

- Extracción y selección de descriptores.
- Representación.
- Aproximaciones al reconocimiento.
- Análisis de escenas.
- Visión en 3D.

Estos descriptores se pueden enlazar a través de un esquema de procesamiento de imágenes [2] donde quedarían englobados en la figura 1.

A través de este esquema podemos observar que los descriptores de la asignatura de T.D.I. quedarían enmarcados mediante las fases de adquisición de imágenes y preprocesado. Mientras que para la asignatura correspondiente a V.A. las fases a tener en cuentan serían segmentación, representación y descripción y reconocimiento e interpretación.

3. El entorno de trabajo IMtdi

La principal dificultad para la realización de prácticas de tratamiento de imágenes radica en que, antes de poder aplicar cualquier función sobre una imagen es necesario leerla del medio de almacenamiento, almacenarla en una estructura de datos del programa y, frecuentemente, realizar unas tareas de preproceso. Ahí es donde resulta útil este entorno, que ofrece las ayudas necesarias para realizar estas tareas previas con el mínimo esfuerzo.

Por otra parte, en la realización de prácticas de Visión Artificial la dificultad estriba en que, antes de poder abordar las tareas más propias de las prácticas de esta materia, el alumno ha de vérselas con otras que se encuadran más en las técnicas de Tratamiento Digital de Imágenes. También ocurre que para realizar una práctica de clasificación y reconocimiento de imágenes, es indispensable el contar con una serie de descriptores de los objetos que en ellas aparecen y que no son rápidos de implementar.

El entorno para prácticas IMtdi [3] [4] está ideado para resolver estos problemas. Utilizando las herramientas y librerías disponibles en el entorno de trabajo, el alumno podrá crear fácilmente en C++ sus propios programas en los que ponga a prueba las funciones de Tratamiento de Imágenes o las de Visión Artificial ambas objeto de interés, dedicando un mínimo esfuerzo a tareas de preprocesamiento.

El objetivo fundamental es que el alumno pueda centrarse en el objetivo de la práctica que pretenda desarrollar sin pérdidas de tiempo en la implementación de funciones que no forman parte de los objetivos, pero que son necesarias para poder realizarla. Por ejemplo, el alumno puede realizar una práctica de implementación de filtros sin tener que ocuparse de la lectura o escritura de la imagen.

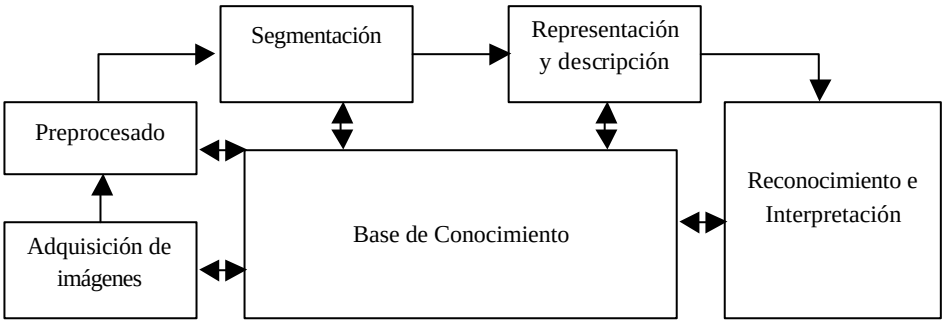


Figura 1. Sistema de procesamiento de imágenes.

Por motivos de simplicidad y, dado el hecho de que la mayoría de los algoritmos estudiados trabajan sobre imágenes representadas en niveles de gris, el entorno está especialmente diseñado para manipular este tipo de imágenes y, si bien admite imágenes en color, la cobertura para ellas es reducida.

En bs siguientes subapartados se describe de manera simplificada la potencialidad del entorno IMtdi para el desarrollo de las prácticas de las asignaturas T.D.I. y V.A.

3.1. Matrices

La estructura básica para la representación de una imagen [3] es la matriz. Por tanto, su diseño se ha hecho de forma que mejora notablemente las capacidades básicas de manipulación de estas estructuras presente en el lenguaje de programación utilizado, C++. Este módulo, implementado como una clase de C++ denominada *C_Matrix* permite definir y manipular matrices de forma cómoda y segura. Con *C_Matrix* se pueden definir matrices de números reales, acceder a sus elementos, obtener o modificar sus características, etc. También se incluye una buena cantidad de funciones, algunas de ellas claramente orientadas al tratamiento digital de imágenes. Las operaciones que podemos encontrar pueden ser: funciones estadísticas, asignación y limitación de valores, operaciones matemáticas, funciones de propósito específico (como la Gaussiana o la convolución) y métodos de entrada-salida.

3.2. Imágenes

Una imagen se representa mediante una matriz de índices que hacen referencia a los elementos de una paleta. A su vez, la paleta es otra matriz, en la que cada columna representa el valor para un componente de color y cada fila es un color utilizado por la imagen.

En el entorno de programación IMtdi se ha incluido un módulo para manipulación de imágenes. En él se encuentra definida la clase *C_Image* que contiene los datos y métodos necesarios para el trabajo con imágenes con paleta. La clase *C_Image* se ha definido como clase derivada de *C_Matrix* por lo que hereda todos los métodos definidos para las matrices. Así, si aplicamos cualquier método de la clase *C_Matrix* sobre una imagen, operará sobre la matriz principal de ésta. Además, se incluyen algunos datos específicos de una imagen, como es la presencia de la paleta o los métodos de lectura y escritura de formatos gráficos. La paleta también es una matriz, pero para utilizar sobre ella los métodos definidos para matrices, hay que hacer referencia explícita a ella.

3.3. Preprocesado

La fase del *preprocesado* está constituida por un conjunto de métodos que tienen la finalidad de realzar en la forma aquellas características que nos interesan para una aplicación concreta. La idea se basa en desechar todas aquellas características que no sean útiles para el proceso al que pretendamos someter a la

imagen, al tiempo que se deben conservar aquellas que en alguna medida se consideren importantes.

En el entorno de programación IMtdi se han incluido los siguientes métodos para el preprocesado de imágenes:

- Mecanismos de aproximación al rango de colores (como truncamiento o suavizado).

- Métodos de procesamiento de puntos.
  - Función inversa.
  - Aumento de contraste.
  - Ecualización del histograma.
  - Filtro exponencial.
  - Filtro logarítmico.
  - Función gamma.
  - Función uniforme.

- Métodos de procesamiento de máscaras.
  - Métodos de aplicación de matrices de convolución.
  - Filtro de la media.
  - Filtro de la mediana.
  - Filtro paso bajo.
  - Filtro paso alto.
  - Filtro de High-Boost.
  - Filtro de Gauss.
  - Filtro de conservación.
  - Filtro de crimmins.

3.4. Segmentación

La segmentación es una tarea de fundamental importancia en el análisis de imágenes. Su objetivo es dividir la imagen en partes, cada una de las cuales se corresponde con un objeto de la realidad. En general, los métodos de segmentación se basan en el hecho de que los píxeles que forman un objeto mantienen una cierta similitud entre ellos y, sin embargo, en las fronteras que separan dos objetos aparecen mayores variaciones.

En el entorno de tratamiento de imágenes IMtdi se han incluido algunas funciones para segmentación de imágenes. Se encuentran definidos en la clase C_Matrix, por lo que operan tanto con matrices como con imágenes, ya que se trata de una clase derivada. Sin embargo, hay que hacer notar que su ámbito de aplicación son las imágenes en tonos de gris, y siempre que la paleta de grises se encuentre normalizada.

Los métodos de segmentación que se integran son los siguientes:

- Métodos de segmentación por niveles de gris.
  - Isolíneas.
  - Floodfill.
  - Umbralizado.

- Métodos de segmentación por gradientes.
  - Gradiente y pseudogradiente.
  - Watershed.

3.5. Representación y descripción

La representación de una región en una imagen comprende dos tipos de procedimientos:
Representación mediante las características externas (contorno).
Representación mediante las características internas (los píxeles que comprenden la región).

El seleccionar un esquema u otro de representación vendrá determinado por el área de aplicación, así como, el tipo de problema que se desea resolver.

En el entorno IMtdi se han programado tanto descriptores externos o topológicos como descriptores internos. Esto facilita la labor a la hora de analizar imágenes con vista a reconocer objetos incluidos en ellas, ya que todo el proceso de extracción de características se encuentra desarrollado en dicho entorno. Además las características extraídas de una imagen permiten ser almacenadas en formatos tan comunes como el CSV también usado por Excel.

3.6. Clasificación

El problema de la clasificación consiste básicamente en lograr una partición del espacio de características en regiones mutuamente excluyentes y en asignar a cada región una clase.

El entorno IMtdi aunque no implementa esta labor, la hace posible proporcionando los descriptores en los que ésta se basa. Además, estos descriptores son almacenados en formatos legibles por cualquier procesador de hojas de cálculo. Esto permite que o bien se implementen los algoritmos de clasificación o bien se intenten resolver por las herramientas de las que dispone licencia la universidad, como puede ser

el uso del SPSS para los algoritmos tanto de clasificación supervisada como de no supervisada.

4. Conclusiones

La docencia práctica de T.D.I. y V.A. se ha planteado desde las siguientes perspectivas:

- Facilitar el aprendizaje e implementación de algoritmos propuestos en el desarrollo de las prácticas a través del entorno IMtdi.

- Centrar al alumno en el estudio de técnicas de las asignaturas y no tanto en los detalles de la implementación.

El entorno IMtdi desarrollado para la docencia práctica de las asignaturas descritas ha permitido que el alumnado por un lado refuerce los conceptos teóricos y por otro desarrolle prácticas a una complejidad impensable con la temporización propuesta de las asignaturas lo que refuerza su propia satisfacción y la de los profesores que la imparten.

Referencias

[1] Deryn Graham and Anthony Barrett. *Knowledge-Based Image Processing System*. Springer, 1997.

[2] Gonzalez, R. *Digital Image Processing*. Addison Wesley, 1992.

[3] Guindos Rojas F. y Piedra Fernández J. A. *Tratamiento de Imágenes con IMtdi*. Universidad de Almería. Servicio de Publicaciones, 2001.

[4] Guindos Rojas F., Piedra Fernández J. A. y Peralta López M. *Visión Artificial con IMtdi.* Universidad de Almería. Servicio de Publicaciones, 2001.

[5] Sonka, M. Image Processing, Analysis and Machine Vision. Champan & Hall, 1995.

Métodos pedagógicos innovadores

Enseñar informática es como...

Daniel Gayo Avello
Dpto. de Informática
Universidad de Oviedo
C/Calvo Sotelo s/n 33007 Oviedo
e-mail: dani@lsi.uniovi.es

Hortensia Fernández Cuervo
Licenciada en Psicopedagogía
e-mail: tensi_f@hotmail.com

Resumen

Un problema habitual al explicar conceptos informáticos es su alto nivel de abstracción así como la dificultad para proporcionar ejemplos adecuados y, a la vez, próximos a la experiencia personal de los estudiantes.

Una posible solución para estos problemas radica en la utilización de metáforas como organizadores previos que permitan al alumno captar la esencia de los conceptos que emulan para poder extrapolar esos nuevos conocimientos al campo informático y aprender con mayor facilidad los conceptos explicados.

En este artículo se presentan algunos aspectos de las actuales teorías sobre la metáfora así como las implicaciones que tienen en el ámbito docente. Por último, se muestran unas técnicas muy sencillas para el desarrollo de metáforas didácticas y algunas metáforas construidas mediante su aplicación.

1. Introducción

Un problema muy habitual en la docencia en general y en la didáctica de la informática en particular está en la forma en que transmitimos a nuestros alumnos unos conocimientos con un elevado grado de abstracción y, generalmente, muy alejados de sus experiencias y conocimientos previos.

Si se quiere lograr un aprendizaje verdaderamente significativo es imprescindible partir de los conocimientos que poseen los estudiantes y enlazarlos con los conocimientos que les queremos transmitir.

Naturalmente, esta tarea no es nada fácil y no hay una única manera de llevarla a cabo siendo el uso de metáforas una de las formas posibles. A lo largo de este artículo se expondrán brevemente algunas implicaciones cognitivas y didácticas sobre la metáfora y su uso en la docencia, además de esbozar una sencilla técnica para el desarrollo de metáforas didácticas y presentar algunos ejemplos de las mismas.

2. Teoría actual de la metáfora

Las teorías sobre la metáfora pueden remontarse hasta Aristóteles que en su obra *Poética* dice que "*la metáfora es la aplicación a una cosa de un nombre que es propio de otra*"; así, la metáfora sería un fenómeno puramente léxico y, aunque no se descarta por completo su valor cognoscitivo, su función primordial sería la de proporcionar al entendimiento un placer estético.

Desde entonces, el interés por la metáfora no ha hecho más que aumentar y no sólo desde un punto de vista filosófico o lingüístico sino también en los campos de la psicología cognitiva y la filosofía de la ciencia. Estos últimos serán los que más nos interesen como científicos y docentes.

Dos de los autores que más han influido en la visión que se tiene de la metáfora en ambos campos son Max Black y George Lakoff. El primero, sobre la base de los trabajos de Ivor Richards, propuso una concepción interaccionista según la cual en toda metáfora existen dos polos que interaccionan entre sí, siendo el significado de la metáfora el resultado de dicha interacción. El segundo, por su parte, reivindicó la importancia de la metáfora como elemento facilitador del aprendizaje.

Así, en la actualidad se considera que la metáfora es el principal mecanismo para la comprensión de conceptos abstractos puesto que, al proporcionar un "mapeo" entre dos dominios conceptuales diferentes, permite entender un dominio relativamente abstracto o poco estructurado en base a términos de otro más sencillo o, al menos, más estructurado. Esta concepción ha resultado fundamental para la aplicación didáctica de la metáfora que se viene llevando a cabo durante los últimos años.

3. La metáfora como organizador previo

Según Ausubel [1], el alumno alcanza un aprendizaje significativo al establecer relaciones entre los nuevos conocimientos que recibe y sus conocimientos previos (e.g., al entender el área de un triángulo a partir de la superficie de un rectángulo).

Esta reestructuración de conocimientos puede apoyarse mediante el uso de los denominados "organizadores previos". Estos organizadores son un material introductorio desarrollado por el profesor a fin de proporcionar un marco de alto nivel para los nuevos conocimientos que se van a presentar. El propósito fundamental de los organizadores previos es utilizar lo que ya saben los alumnos empujándoles a reflexionar sobre ello para ayudarles a comprender lo que desconocen y se les trata de explicar.

Los organizadores previos pueden clasificarse, a su vez, en expositivos y comparativos. Los primeros parten de una situación más general que el concepto a explicar a fin de mostrar la estructura general del mismo; los segundos, en cambio, pretenden establecer vínculos entre los conocimientos del alumno y elementos presentes en el organizador para extender los primeros hacia el nuevo conocimiento que se va a describir.

De esta forma, la metáfora puede ser utilizada como un organizador previo comparativo; esto es, mediante la metáfora el profesor puede ofrecer a sus alumnos un "puente" que les permita salvar de una forma más sencilla el salto existente entre lo que ya saben y lo que se les va a explicar y, por tanto, desconocen por completo.

A lo largo de los años han sido muchas las metáforas que han mostrado su utilidad a la hora de lograr un aprendizaje significativo de conceptos informáticos. Pueden citarse algunos ejemplos como [2] que presenta un estudio experimental sobre la influencia de un organizador previo metafórico en la comprensión de los vectores en Pascal, o [3] que describe la "metáfora de la consigna" para la explicación del concepto de memoria dinámica y punteros, así como [4] que demuestra de manera ejemplar la forma en que pueden emplearse las metáforas en un libro de texto[1].

En lo que resta de artículo se presentarán un conjunto de reglas muy sencillas que facilitarán al docente el desarrollo "sistemático" de metáforas didácticas a fin de ofrecerlas a sus alumnos como organizadores previos.

4. Desarrollo de metáforas didácticas

Hemos visto que la metáfora tiene un papel cognitivo fundamental y puede resultar una herramienta extraordinariamente útil para facilitar el aprendizaje significativo. Sin embargo, todas estas teorías no proporcionan medios para el desarrollo de nuevas metáforas; por ello, el profesor que desee crear metáforas didácticas depende de su intuición para finalizar con éxito esa tarea.

En la mayor parte de los casos la intuición del docente suele bastar para proporcionar metáforas adecuadas a los alumnos; sin embargo, no está de más proporcionar algunas pautas muy sencillas que han facilitado el trabajo de los autores con anterioridad.

La idea fundamental consiste en determinar la naturaleza básica del concepto a explicar; así, debe determinarse si el término es un sistema (e.g., la arquitectura de un ordenador), un concepto "tangible" (e.g., punteros y memoria dinámica) o un concepto "abstracto" (e.g., tipos definidos por el usuario).

[1] En [4] se presentan una serie de metáforas extraordinariamente claras para explicar, por ejemplo, el procesamiento de interrupciones (p.61) o el bloqueo mutuo (p.159); también es de destacar la forma en que las citas que abren cada capítulo actúan como organizadores previos azuzando la curiosidad del lector.

Una vez se ha determinado la naturaleza del concepto, deben seguirse una serie de pasos que conducirían a una metáfora didáctica para el mismo o, al menos, a un "prototipo rápido" de metáfora susceptible de ser "refinado".

4.1. Metáforas sobre sistemas

El caso más sencillo que se puede plantear a la hora de desarrollar una metáfora didáctica es aquel en el que el concepto a describir es un sistema[2]. En este caso se debe comenzar determinando los principales componentes que forman el sistema a describir, procurando no superar los siete elementos más/menos dos [5]; si existiera un número mayor de componentes el docente debería plantearse una simplificación del sistema o bien el desarrollo de una metáfora para cada componente individual.

Cuando ya se dispone de una lista de componentes se procede a determinar las funciones esenciales que lleva a cabo cada uno; nuevamente deberá procurarse que el número de tareas por componente sea reducido.

A continuación, para cada componente, teniendo en cuenta la lista de acciones básicas que lleva a cabo, se detalla una serie de elementos cotidianos[3] capaces de sustituirlo tal y como ha sido caracterizado (es decir, en función de las acciones básicas que lleva a cabo). La metáfora final se construirá sustituyendo con estos nuevos objetos los componentes originales del sistema.

Una de las mayores dificultades de este tipo de metáforas es el conseguir un escenario plausible; por esa razón debe procurarse que la utilización conjunta de distintos elementos resulte lo menos forzada posible.

Una vez se obtiene la metáfora se hace necesario probarla para determinar su idoneidad; para ello pueden utilizarse algunas de las relaciones que Gentner [8] definió para valorar una metáfora: especificidad, riqueza, abstracción, sistematicidad, validez, exhaustividad, transparencia y extensibilidad. Las más interesantes de cara a una metáfora didáctica son la especificidad, la claridad y la transparencia.

La especificidad indica hasta qué punto la base de la metáfora es inteligible; la claridad se refiere a la precisión de las correspondencias entre elementos de la metáfora y los elementos del concepto explicado, mientras que la transparencia indica la facilidad con que el receptor percibe qué elementos de la metáfora son aplicables al concepto explicado y cuáles no lo son.

Obviamente, una metáfora didáctica debe ser específica, esto es, emplear unos términos fácilmente comprensibles por todos los estudiantes (otra razón para recomendar la utilización de elementos cotidianos); además, el alumno debe percibir fácilmente las correspondencias entre elementos de la metáfora y del concepto así como aquellos elementos metafóricos que no son aplicables al concepto original, es decir, la metáfora debe ser lo más clara y transparente posible. Si la metáfora desarrollada es específica, clara y transparente tan sólo resta especificar claramente sus puntos fuertes y sus puntos débiles (aquellos elementos no aplicables al concepto) y preparar algunos "escenarios" para la metáfora que "recuerden" contextos comunes para el concepto explicado.

Por último, a la hora de exponer metáforas de sistemas conviene destacar las similitudes entre cada componente del sistema real y de la metáfora, mostrar el funcionamiento del sistema metafórico y compararlo con el sistema real y señalar las diferencias entre la metáfora y el concepto original. Más adelante en este artículo se muestra un ejemplo de este tipo de metáforas utilizada para explicar la arquitectura Von Neumann.

4.2. Metáforas sobre conceptos "tangibles"

En la mayor parte de los casos, el concepto a

[2] Según el diccionario de la R.A.E., un sistema es un conjunto de cosas que relacionadas entre sí ordenadamente contribuyen a determinado objeto.

[3] A lo largo del artículo se insistirá frecuentemente en que las metáforas deben construirse con elementos cotidianos. Esta recomendación no es trivial, ya [6] mantiene que un nivel de contreción elevado facilita un aprendizaje significativo, situándose en una línea similar [7] al señalar que las metáforas espaciales son las que permiten un aprendizaje más adecuado. Teniendo en cuenta esto, así como nuestra experiencia con el desarrollo y uso de distintas metáforas, creemos que construir las metáforas en base a elementos manejados habitualmente por los alumnos las hace más claras y mejora su propósito docente.

explicar no es tan claro como un sistema formado por un número fijo de componentes; sin embargo, muchos conceptos aún son lo suficientemente tangibles como para señalar varios elementos con algún tipo de relación entre sí y capaces de realizar o recibir acciones.

Si el concepto a tratar se encuadra en esta categoría se debe comenzar enumerando un conjunto reducido de elementos (sustantivos) y acciones (verbos) que describan la esencia del concepto a explicar.

Por cada elemento de la lista se escribe una lista de "sinónimos" procedentes de ámbitos no informáticos. A partir de estas listas de características y acciones genéricas se localizan elementos cotidianos que pudieran ser descritos en base a tales atributos.

Si en el caso anterior se buscaban objetos comunes para construir la metáfora y se descartaban en caso de que no pudieran integrarse de una forma plausible, en éste se trata de localizar una serie de elementos candidatos y, en una primera fase, evaluar la claridad de los mismos descartando aquellos con menor grado de correspondencia con el objeto a explicar para, en una segunda, elegir la metáfora más transparente. En la sección "Ejemplos de metáforas didácticas" se presenta una metáfora de este tipo desarrollada para explicar punteros y memoria dinámica.

4.3. Metáforas sobre conceptos "abstractos"

El caso más complicado a la hora de desarrollar una metáfora didáctica se plantea cuando el concepto a explicar es totalmente abstracto.

En semejantes ocasiones la estrategia más adecuada consiste en enumerar el mayor número posible de términos que ayuden a caracterizar el concepto que se pretende describir metafóricamente.

Una vez se dispone de un conjunto adecuado se ordena en función de la relevancia que tiene cada término de cara a describir el concepto de partida, seleccionándose finalmente el conjunto mínimo que permita describir los rasgos fundamentales del concepto a explicar.

Posteriormente se desarrolla una lista de conceptos que puedan servir como posible metáfora, todos ellos deben ser caracterizables mediante ese conjunto mínimo de rasgos. Finalmente, aplicando de nuevo los criterios de claridad y transparencia se selecciona el candidato que constituye la metáfora más adecuada.

Como último ejemplo se presenta una metáfora didáctica que pretende mostrar las ventajas del diseño descendente de programas frente al diseño ascendente.

5. Ejemplos de metáforas didácticas

En el punto anterior se han descrito algunas reglas sencillas que tienen como finalidad facilitar el desarrollo de metáforas didácticas. En este apartado se presentan, a modo de ilustración para tales técnicas, tres ejemplos de metáforas didácticas desarrolladas por los autores y que están siendo utilizadas con éxito en asignaturas de primer curso de Ingeniería Técnica en Informática y de la Licenciatura de Matemáticas.

5.1. "La Caja"

La arquitectura Von Neumann es un concepto fundamental que debería explicarse a todo alumno que vaya a enfrentarse por vez primera a una asignatura de programación. Aún cuando se trata de un concepto muy sencillo y podría explicarse directamente, introduce una serie de términos (memoria, unidad aritmética, dispositivos de entrada/salida, etc.) que, al ser nuevos para el alumno, pueden hacer la comprensión del mismo más difícil.

Por esa razón se consideró interesante desarrollar una metáfora que permitiese explicar la arquitectura Von Neumann en términos accesibles para el alumno; de este modo, una vez se establece un marco común en el que todos los estudiantes se sienten cómodos, es posible (re)explicar la arquitectura empleando sus propios términos.

A la hora de desarrollar una metáfora para explicar este concepto debemos, en primer lugar, determinar su naturaleza; al tratarse de un sistema se procede a detallar los elementos que lo constituyen:

1. Memoria.
2. Unidad aritmético-lógica.
3. Dispositivos de E/S.

4. Procesador.

Puesto que el número de componentes es reducido no es necesario simplificar el sistema ni desarrollar metáforas independientes para cada elemento. Se procede, pues, a determinar las acciones básicas que caracterizan cada componente:

- Memoria: permite leer/escribir información (datos e instrucciones) de forma no permanente.
- Unidad aritmético-lógica: realiza operaciones matemáticas y lógicas sencillas.
- Dispositivos de E/S: permiten la comunicación del sistema con el exterior.
- Procesador: procesa instrucciones y emplea los elementos anteriores de manera adecuada para resolver dichas instrucciones.

A partir de esta lista de elementos caracterizados por las acciones que llevan a cabo se enumeran una serie de elementos cotidianos que puedan "sustituirlos". Este proceso sigue una estrategia de ensayo y error puesto que no todos los elementos inicialmente seleccionados permitirían una metáfora plausible. Al finalizar el proceso de selección se decide emplear los siguientes:

1. Una pizarra en sustitución de la memoria.
2. Una calculadora en lugar de la unidad aritmético-lógica.
3. Un teléfono para representar el canal de entrada/salida.
4. Un ser humano capaz de utilizar la pizarra, la calculadora y el teléfono como sustituto del procesador.

Todos los elementos serían colocados en un recinto cerrado, "La Caja", de tal forma que la única vía de comunicación con el exterior sería la línea telefónica.

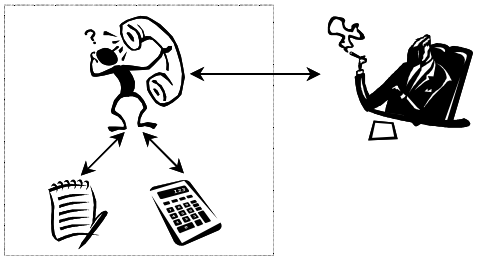


Fig. 1. "La Caja" (metáfora de la arquitectura Von Neumman).

Esta situación, aunque algo extraña, es plausible por lo que tan sólo restaría determinar la claridad y transparencia[4] de la misma y, a partir de aquí, los puntos fuertes y débiles que presenta.

Esta metáfora resulta muy clara puesto que se ha buscado un sustituto para cada componente del sistema real que lleve a cabo acciones similares a las del elemento original. Aún así, deberían comprobarse las distintas correspondencias:

- La pizarra se corresponde con la memoria, permitiendo escribir y leer información (tanto datos como instrucciones).
- La calculadora se corresponde con la unidad aritmético-lógica permitiendo realizar operaciones aritméticas sencillas.
- El teléfono se corresponde con el canal de E/S y permite intercambiar información entre el exterior y el sistema.
- El ser humano se corresponde con el procesador, como éste comprende instrucciones sencillas y, de la misma forma en que el procesador utiliza de forma adecuada la memoria, la unidad aritmético-lógica y los dispositivos de E/S, el ser humano utiliza la pizarra, la calculadora y el teléfono.

Como se puede ver, las correspondencias entre la metáfora y el concepto que pretende explicar son muchas y fácilmente apreciables.

Sin embargo, dichas correspondencias no son totalmente fieles; el concepto de transparencia indicará si las características de la metáfora no aplicables al concepto son fácilmente identificables por el alumno:

- La pizarra debe borrarse de manera explícita, sin embargo, en la implementación que conocen los alumnos de la arquitectura Von Neumann, el ordenador electrónico, la memoria se borra al cesar el suministro eléctrico.
- La calculadora permite realizar operaciones aritméticas pero no lógicas.
- El teléfono es un canal de entrada/salida

[4] Si en la construcción de las metáforas se emplean únicamente elementos cotidianos no sería estrictamente necesario comprobar la especificidad de la metáfora puesto que todos los alumnos comprenderían fácilmente los elementos utilizados.

pero no un dispositivo de E/S.
- El ser humano acepta muchas más "instrucciones" que un procesador real, además de poder actuar como memoria y como unidad aritmético-lógica.

Resulta obvio que la metáfora presenta varios puntos débiles, sin embargo, son fácilmente identificables y, aún cuando no todos fueran detectados por el alumno, serían fácilmente reconocibles al ser señalados por el profesor.

A la vista de este somero análisis resulta bastante evidente que la metáfora propuesta es plausible, clara y transparente por lo cual es susceptible de ser utilizada en el aula pudiendo emplearse de muchas formas diferentes:
- Para explicar los componentes de la arquitectura por analogía una vez explicados los de "La Caja"; en este caso es fundamental señalar los aspectos en los que se diferencian la metáfora y el concepto real.
- Para explicar el funcionamiento de la arquitectura Von Neumann directamente sobre "La Caja"; se puede pedir a los alumnos que escriban un algoritmo para dictárselo a un hipotético compañero situado en el interior de la misma y discutir la forma en que se "ejecutaría".
- La actividad anterior puede llevarse a cabo también mediante dinámica de grupos, colocando a un alumno dentro de "La Caja" y dividiendo la clase en grupos encargados de construir algoritmos que serían introducidos en el sistema.

Esta metáfora ha sido utilizada de forma muy satisfactoria con alumnos de primer curso de la asignatura de Algorítmica y Lenguajes de Programación (Licenciatura de Matemáticas) así como en una lección de introducción a la informática en el Programa Universitario para Mayores Universidad de Oviedo[5].

5.2. Calle Falsa, 123

Los conceptos de punteros y memoria dinámica son tremendamente importantes y también son fuente de muchos quebraderos de cabeza tanto para los alumnos como para los docentes a la hora de explicarlos de una manera atractiva y fácilmente comprensible.

Esta problemática llevó a los autores a buscar una metáfora alternativa a las ya existentes (e.g., la descrita en [3]). Esta metáfora es distinta de la presentada en el punto anterior puesto que en este caso no se trata de describir un sistema sino un concepto más abstracto aunque aún relativamente tangible.

Para ello se procedió en primer lugar a enumerar "sustantivos" que describiesen de manera adecuada lo esencial del concepto, se encontraron dos principales:
- Puntero.
- Variable apuntada.

Seguidamente se escribió una lista de "verbos" que hicieran referencia a acciones sobre los elementos anteriores:
- Asignar memoria.
- Liberar memoria.
- Asignar un valor a un puntero.
- Asignar un valor a una variable apuntada.
- Eliminar un puntero.

Posteriormente se procedió a elaborar listas de "sinónimos" de ámbitos no informáticos; empleándose los criterios de claridad y transparencia para su refinamiento. Finalmente los sinónimos que se obtuvieron fueron los siguientes:
- Como "sinónimo" de puntero, dirección postal apuntada en una tarjeta.
- Como "sinónimo" de variable apuntada, un solar (o un edificio).

Por lo que respecta a las acciones básicas:
- Escribir una dirección en una tarjeta sería la acción equivalente a la de asignar un valor a un puntero.
- Romper una tarjeta con una dirección escrita equivaldría a eliminar un puntero.
- Escribir la dirección de un solar vacío sería equivalente a asignar memoria.
- Construir un edificio en un solar indicado por una dirección sería el equivalente a asignar un valor a una variable apuntada.
- Demoler un edificio equivaldría a liberar memoria.

Esta metáfora es bastante plausible (salvo por la velocidad con la que se construirían y derruirían edificios) y resulta también clara y transparente.

A la hora de utilizarla en clase se hizo en dos fases y siempre mostrando tras la metáforas

[5] www.uniovi.es/Vicerrectorados/Extension/pumuo/

la contrapartida con punteros.

En la primera fase se utilizó para separar claramente el concepto de puntero del de variable apuntada. Los alumnos distinguen perfectamente una tarjeta con la dirección "Calvo Sotelo s/n 33007. Oviedo" del edificio situado en esa dirección y, a partir de éste y otros ejemplos, pueden extrapolar el mismo conocimiento al ámbito de la memoria dinámica.

En esta misma fase se puede explicar de forma similar el concepto de puntero "basura" (utilizando una tarjeta con una dirección ficticia como "Calle Falsa, 123. Springfield") o la existencia de edificios (datos) a los que no es posible llegar puesto que se ha perdido su dirección (puntero).

Llegados a este punto, los alumnos ya distinguen la diferencia entre la modificación o desaparición de una variable de tipo puntero y la modificación o eliminación de la variable apuntada. En este momento se pasa a la segunda fase en la que se explican los conceptos de asignación y liberación de memoria.

La liberación de memoria es muy sencilla, dada una dirección siempre puede demolerse el edificio allí situado para dejar un solar. La asignación de memoria puede explicarse mediante la asignación a una tarjeta de una dirección que se corresponde con un solar vacío, en el que se podrán construir diferentes edificios de distintas alturas.

Este último es el principal punto débil de la metáfora y así se les debe hacer notar a los alumnos. Las direcciones sólo indican la localización de un solar o edificio pero no la clase de edificio (el tipo de dato). Sin embargo, los punteros siempre apuntan a un tipo determinado no existiendo la posibilidad de decidir el tipo en el momento de asignar memoria.

A pesar de este inconveniente, la metáfora cumple el objetivo fundamental para el que fue creada: explicar conceptos básicos de punteros y memoria dinámica, habiendo sido utilizada en la asignatura de Algorítmica y Lenguajes de Programación.

5.3. Los fabricantes de puzzles

En todas las asignaturas de programación se introducen los conceptos de diseño descendente y ascendente, primándose generalmente el primero sobre el segundo.

Normalmente, cuando se explica esto a los alumnos éstos aún tienen muy poca experiencia y los ejercicios que han visto aún son demasiado sencillos como para poder apreciar las diferencias entre un enfoque descendente y uno ascendente.

Por esa razón, se planteó la necesidad de elaborar una metáfora para que los estudiantes pudiesen apreciar de una forma casi inmediata las características de uno y otro planteamiento y las razones que hacen preferible el diseño descendente frente al ascendente.

El problema principal con el concepto que se pretendía explicar de forma metafórica era su total abstracción; por este motivo el punto de partida consistió en el desarrollo de una lista de términos que, de una forma u otra, permitieran caracterizar el diseño descendente y/o ascendente.

Algunos de los términos que entraron a formar parte de esta lista fueron los siguientes: descomposición, integración, interfaz, acoplamiento, detalle, particularidad, generalidad, modularidad, etc.

Una vez finalizada la lista se seleccionó el menor número posible de términos para describir de manera adecuada el concepto a explicar optándose por: descomposición, integración, interfaz y detalle.

Posteriormente se procedió a buscar distintos conceptos que pudieran ser descritos en base a esos términos, seleccionándose finalmente como metáfora el proceso de fabricación de puzzles.

Esta metáfora, al referirse a un concepto totalmente abstracto, es la menos clara y transparente de los tres ejemplos presentados y precisa una explicación algo más detallada.

En la metáfora de la fabricación de puzzles están presentes los siguientes elementos:

- La imagen que aparece al construir el puzzle es el problema resuelto por el algoritmo a diseñar.
- Las piezas que forman el puzzle se corresponden con los distintos módulos que aparecen en el proceso de diseño.
- La forma de cada pieza y la manera en que encaja con sus vecinas representa los interfaces de los distintos subprogramas.

- El proceso de diseño descendente o ascendente es representado en la metáfora por la forma en que se obtienen las piezas que conforman el puzzle.

En la metáfora que se describe se explica a los alumnos que hay dos formas de obtener las piezas del puzzle: empleando un troquel que trocea la lámina en las distintas piezas (diseño descendente) o bien recortando las piezas de forma individual y dibujando en cada una la fracción de la figura total que le corresponde (diseño ascendente).

A la vista de esto resulta evidente que esta metáfora también es la menos transparente de las tres presentadas puesto que aunque son muchos los aspectos de la metáfora no aplicables al concepto explicado no son fáciles de apreciar por parte de los alumnos.

Por esa razón, esta metáfora resulta muy útil plantearla para su discusión y comentario dentro del grupo clase. La finalidad fundamental de la discusión es que los alumnos se percaten de que el diseño ascendente y, especialmente, el descendente no se parecen a la forma en que se fabrican los puzzles y que, sin embargo, empleando uno u otro pueden obtenerse efectos similares a los logrados en la metáfora: modularidad o nivel de detalle incorrectos, interfaces poco adecuados, difícil integración de los elementos, etc.

La metáfora de los fabricantes de puzzles se ha utilizado con éxito en la asignatura Metodología de la Programación I de Ingeniería Técnica en Informática.

6. Conclusiones

En este artículo se han presentado las líneas teóricas fundamentales que marcan el actual estudio de la metáfora así como una estrategia muy sencilla para el desarrollo sistematizado de metáforas didácticas.

Esta estrategia permite obtener, a partir del concepto que se desea explicar, metáforas plausibles, claras y transparentes que pueden representar tanto sistemas o conceptos tangibles como conceptos totalmente abstractos.

Para ilustrar la utilización de la estrategia propuesta se han descrito tres metáforas didácticas diferentes abarcando desde la arquitectura Von Neumann hasta el diseño descendente pasando por punteros y memoria dinámica.

Referencias

[1] David P. Ausubel. *Psicología educativa: un punto de vista cognoscitivo*. Edición en español de *Educational Psychology. A cognitive View*. 1976/1968.

[2] Katherine N. Macfarlane, Barbee T. Mynatt. *A study of an advance organizer as a technique for teaching computer programming concepts*. Proceedings of the 19th SIGCSE Technical Symposium on Computer Science Education, pp. 240-243. 1988.

[3] Ricardo Jiménez Peris, Cristóbal Pareja Flores, Marta Patiño Martínez, J. Ángel Velázquez Iturbide. *The Locker Metaphor to Teach Dynamic Memory*. ACM SIGCSE Technical Symposium'97. San José, California, EE.UU., pp. 169-173. 1997.

[4] Harvey M. Deitel. *Introducción a los sistemas operativos (segunda edición)*. Edición en español de *Operating Systems, Second Edition*. 1993/1990.

[5] George A. Miller. *The Magical Number Seven, Plus or Minus Two: Some Limits on Our Capacity for Processing Information*. The Psychological Review, vol. 63, pp. 81-97. 1956.

[6] Richard E. Mayer. *Some conditions of meaningful learning for computer programming: advance organizers and subject control of frame order*. Journal of Educational Psychology, vol. 68 (2), pp. 143-150. 1976.

[7] John M. Carroll, John C. Thomas. *Metaphor and the Cognitive Representation of Computing Systems*. IEEE Transactions on Systems, Man, and Cybernetics, vol. 12 (2), pp. 107-116. 1982.

[8] Dedre Gentner. *Structure mapping: A theoretical framework for analogy*. Cognitive Science, vol. 7, pp. 155-170. 1983.

Creatividad y resolución de problemas en las carreras de informática

Gil Gómez, Hermenegildo; Montesa Andrés, José O; Albors Garrigós, José.
Departamento de Organización de Empresas
Universidad Politécnica de Valencia
46022, Valencia
Tel. (96) 387.76.82, Fax (96) 387.76.89
e-mail { hgilgom@omp.upv.es, jomontesa@upvnet.upv.es, jalbors@omp.upv.es, }

Resumen

El siguiente artículo trata de explicar la necesidad de la enseñanza de asignaturas que faciliten las relaciones interpersonales y el crecimiento personal en carreras técnicas. En dicho contexto se presenta la asignatura de Creatividad y Resolución de Problemas en empresas de TI impartida en la Escuela Universitaria de Informática de la Universidad Politécnica de Valencia.

1.- Introducción y justificación.

En un mundo competitivo como el actual, la necesidad de buscar nuevas ideas, de hacer algo distinto, de diferenciarse de nuestros competidores, se ha convertido en esencial. Este tema ha sido muy bien tratado por Michael Porter en su libro Estrategia competitiva.[1].

El empresario innovador busca esta diferenciación, y no sólo en los productos que vende o en los servicios que presta, sino también en las tareas, en la forma como encara los proyectos, etc.

Adoptar la innovación como premisa básica de administración implica convertirla en algo sistemático. Según Peter Drucker la innovación sistemática consiste en [2]:

1. La búsqueda organizada de cambios con un objetivo determinado, y
2. El análisis de las oportunidades que ellos pueden ofrecer para la innovación social o económica.

El proceso sistemático de búsqueda de innovaciones está estrechamente relacionado con el proceso de toma de decisiones. Aquí encontramos dos elementos donde la innovación juega un papel importantísimo:

- En el problema, cuando éste se define como "el aprovechamiento de oportunidades". Este tipo de problema surge normalmente como consecuencia de una nueva idea que se traduce en un nuevo objetivo.
- En la decisión, cuando es necesario generar cursos de acción alternativos. En este caso la innovación viene dada por la cuota de creatividad aportada al proceso y la implementación de cursos de acción diferentes de los tradicionales.

En este sentido se orienta la propuesta de asignatura impartida en la Escuela Universitaria de Informática de la Universidad Politécnica de Valencia “Creatividad y Resolución de Problemas en Empresas de TI” con el objetivo de ofrecer a la sociedad personas que generen aplicaciones informáticas que permitan a sus empresas u organizaciones obtener mejoras competitivas mediante la aplicación de técnicas y habilidades específicas.

Toda la enseñanza clásica se nutre de conceptos, técnicas y herramientas principalmente basadas en el análisis, la lógica y la estructuración. Consecuentemente, muchos de nuestros alumnos se encuentran "bloqueados" para generar alternativas creativas.

Conocer el proceso creativo y sus técnicas así como la necesidad de resolver problemas permitirá a los estudiantes desarrollar una parte de

su capacidad potencial que redundará en propuestas creativas.

Sin embargo, ser creativo no es ser innovador. La innovación es creatividad aplicada. Para ser creativo e innovador es necesario que vayamos más allá del proceso creativo y sus técnicas. La innovación tiene relación directa con la implementación.

Theodore Levitt se refería a la creatividad e innovación diciendo:

> "Creatividad es pensar cosas nuevas. Innovación es hacer cosas nuevas. Las ideas son inútiles a menos que sean usadas. La prueba de su valor está en su implementación."

El proceso creativo y la innovación tienen una estrecha relación con el proceso de toma de decisiones. En general, podríamos decir que la creatividad es más importante en las primeras etapas, cuando el proceso divergente es más necesario.

Luego, cuando entramos en acción, la creatividad se convierte en innovación y esto forma parte de la implementación.

En las organizaciones hay individuos creativos e individuos innovadores, pero a veces no son la misma persona. Existen innovadores que toman ideas de otros y las llevan a la práctica.

El individuo innovador se basa en la teoría de que el cambio es algo normal y saludable. No lo ve como una amenaza, un enemigo contra el cual tiene que luchar. Adoptar la innovación implica una búsqueda continua y sistemática del cambio con el propósito de adaptar las estrategias y los planes a la nueva realidad.

La creencia común es que el innovador es arriesgado, y en cierta medida lo es. Si no, observemos la lista de víctimas en el campo de la alta tecnología, donde se invierten millones de dólares en proyectos que, en un principio, tienen un alto porcentaje de incertidumbre. Pero quizá sea más arriesgado buscar la optimización en campos donde el mercado ya está dando señales de cambio.

Un ejemplo concreto del riesgo en optimizar lo existente, en lugar de apuntar hacia las nuevas necesidades, lo tenemos en el mercado de la Informática: IBM mantuvo su mira en el hardware y Microsoft apuntó al software. Uno se orientó a optimizar lo existente, el otro a innovar atendiendo el mercado. La ventaja de la innovación en este ejemplo, por ahora, es clara.

2.- Metodología desarrollada.

El método de enseñanza se desarrolla eligiendo un modelo combinado de exposición teórica para explicar los conceptos de la asignatura, tipo lección magistral participativa y una parte que la denominamos Práctica de Aula donde mediante el planteamiento y análisis de casos prácticos reales se pretende aplicar y poner en práctica los conceptos teóricos y de aplicación comentados en la primera parte.

La Lección Magistral sigue siendo el método docente más usado en enseñanza universitaria. Bien realizada es pertinente para el logro de ciertos objetivos: adquirir información actualizada y bien organizada, facilitar la comprensión y aplicación de los procedimientos específicos de la asignatura y elevar los niveles motivacionales de los estudiantes hacia la asignatura [3].

Es fundamental en el desarrollo de una asignatura que pretende desarrollar las habilidades interpersonales de los alumnos que el modelo de Lección Magistral se convierta en un modelo de Lección Magistral Participativa donde se facilita el aprendizaje activo y cooperativo de los estudiantes. Para que en nuestras clases los estudiantes logren sus objetivos marcados de aprendizaje activo y cooperativo es necesario que cumplan las siguientes características y exigencias:

- Estar bien preparada.
- Estar bien estructurada.
- Ser impartida con claridad, expresividad y entusiasmo.
- Dar oportunidad al estudiante para intervenir.
- Manejar eficazmente las intervenciones de los alumnos.
- Despertar la necesidad de seguir aprendiendo: motivar.
- Se fomenta el aprendizaje cooperativo entre los estudiantes.

2.1.- Facilitar la participación.

Nos gustaría hacer hincapié en este punto ya que lo consideramos esencial en el aprendizaje del alumno. El aprendizaje eficaz (aquel que promueve cambios en los conocimientos, destrezas y actitudes) depende del grado de participación activa que los estudiantes tengan en el proceso enseñanza-aprendizaje, bien sea individual o grupalmente.. Para llevar a cabo una buena enseñanza y un buen aprendizaje, consecuentemente, los profesores deben fomentar la participación y cooperación en clase de sus alumnos. Hay dos recursos fundamentales que empleamos para facilitar el aprendizaje activo y cooperativo: el manejo eficaz de las preguntas y las técnicas de grupo en el aula.

2.2 Técnicas de grupo

Los objetivos inmediatos de una enseñanza en pequeños grupos es conseguir que los estudiante hablen y piensen. Los objetivos a largo plazo son el crecimiento y competencia personal [4]. Estos objetivos pueden ser expresados de la siguiente forma:

- El desarrollo de las técnicas de comunicación.
- El desarrollo de las competencias intelectuales y profesionales.
- El crecimiento personal de los estudiantes (y seguramente del docente).

Estos tres objetivos están interconectados en la práctica y cada uno de ellos tiene implicaciones en la función del tutor en la enseñanza de pequeños grupos. Si el tutor o docente pasa la mayor parte del tiempo hablando habrá menor oportunidad de que los estudiantes desarrollen sus técnicas individuales o de comunicación. De hecho la función del profesor es: dirigir la tarea de aprendizaje, los individuos y los procesos grupales.

3 Temario de la Asignatura

A continuación se presenta el temario propuesto de la asignatura de tres créditos impartida en la Escuela Universitaria de Informática de la Universidad Politécnica de Valencia.

En primer lugar se expone un bloque de Resolución de problemas y toma de decisiones donde se plantean metodologías de aplicación individual y grupal, podemos diferenciar tres temas principales:

1. El Proceso de resolución de problemas.
2. Metodologías actuales en la resolución de problemas de forma individual.
3. Metodologías actuales en la resolución de problemas en grupo.

A continuación entraríamos en la parte de creatividad la cual está programada para ocupar las dos terceras partes del tiempo de docencia y tendríamos otros dos bloques diferenciadores:

Bloque II: Enfoque del pensamiento ante conflictos

4. Lógica del pensamiento.
5. El pensamiento lateral.
6. Seis sombreros para pensar.

Bloque III: Creatividad.

7. Mitos sobre la creatividad (Cómo fomentar mi creatividad).
8. Métodos y técnicas para desarrollar la creatividad.

4 Evaluación

La evaluación de la asignatura se establecerá considerando prioritariamente la asistencia y la participación activa de los alumnos, esto tendrá un peso no sólo fundamental sino obligatorio, es decir el alumno que no haya asistido y seguido la asignatura de manera digamos que continua, no tendrá oportunidad de aprobar la asignatura.

Al final de la misma se realizará una prueba en la que se combinarán conceptos básicos teóricos y ejercicios prácticos aplicados. Está prueba determinará o diferenciará las calificaciones finales, aunque la asistencia y el interés en la asignatura y la realización de todos los ejercicios prácticos propuestos a lo largo de las sesiones puede ser suficiente para salvar la asignatura.

5 Bibliografía usada en la asignatura

DE BONO “El pensamiento lateral - Manual de creatividad” -Paidos, 1991
DE BONO “Como enseñar a pensar a tu hijo”, Paidos, 1995
DE BONO, E.: "Seis sombreros para pensar", Editorial Granica. Barcelona. 1997.
VON OECH El despertar de la creatividad Díaz de Santos, 1987
BUZAN, T.: "El libro de los Mapas Mentales", Ediciones Urano. Barcelona. 1996
MIHALKO, MICHAEL: "Thinkertoys. Cómo desarrollar la creatividad en la empresa", Editorial Gestión 2000. Barcelona. 1999.
PONTI, F.: "La empresa Creativa", Editorial Granica. Barcelona. 2001.
GOLEMAN, D., KAUFMANN. P. y RAY, M.: "El espíritu creativo", Vergara. Buenos Aires.
FABRA, M.LL., “Técnicas de Grupo para la Cooperación, CEAC, 1992
ROBBINS, “Comportamiento Organizacional”, Prentice Hall, 1996.
MICHALKO, M., “Los secretos de los Genios de la Creatividad”, Gestión 2000, 1999.
MITROFF, I. “Convierta Problemas en Soluciones Inteligentes”, Amat, 2000.

6.-Reflexiones de los profesores y los alumnos

Los profesores de la asignatura tenemos una percepción positiva de esta primera experiencia. En primer lugar, como ocurre en muchas de las asignaturas de libre elección, nos encontramos con el riesgo de que la propuesta no fuese aceptada por los alumnos y más teniendo en cuenta que es una asignatura ofertada por el departamento de organización de empresas y enmarcada en una carrera técnica como es la de Ingeniero Técnico de Informática. Pero nuestra sorpresa fue la de encontrarnos con 40 matriculados de los cuales unos 33 van habitualmente a clase. El primer día de clase hicimos la pregunta de ¿por qué se habían matriculado en la asignatura? Y la respuesta fue que el título, y más concretamente la palabra “creatividad” les llamaba la atención. A lo largo del curso hemos comprobado como los alumnos se encontraban cómodos en clase, a pesar de que el horario no era muy creativo (de 15:30 a 17:30 horas), participando activamente en los diferentes ejercicios, haciendo preguntas…Además también hemos comprobado el nivel de madurez de ellos, hay que tener en cuenta que son alumnos de primer curso, pero aún así, algunos mostraban un nivel de madurez bastante considerable, aunque también hay gente en el lado opuesto.

6.1 Proyecto futuro

Dentro de la línea que se inició con el Plan de Innovación Educativa (PIE) y que está siguiendo su nueva andadura con el desarrollo del Proyecto Europa, se hace pública la XIII convocatoria de los Proyectos de Innovación Docente (PID’s), que año tras año han ido adaptándose a las necesidades del contexto universitario. Así, esta nueva edición pretende potenciar aquellas iniciativas que el profesorado esté dispuesto a llevar a cabo, y que se orienten fundamentalmente a propiciar un aprendizaje más activo y comprometido por parte de los estudiantes. En este sentido, la experiencia nos ha motivado para presentar un Proyecto de Innovación Docente en el Instituto de Ciencias de la Educación (ICE) de la Universidad.
En la preparación del mismo hemos grabado las actuaciones prácticas de los alumnos, en concreto en un ejercicio donde se aplica la técnica de los 6 sombreros para pensar de Edward de Bono, así como las opiniones de ellos. Observamos la motivación y el interés por la asignatura en sus reflexiones y nos sugieren la necesidad de incrementar esas 30 horas lectivas con que cuenta actualmente la asignatura. El carácter fundamentalmente práctica de la asignatura es otra fortaleza que apuntan los estudiantes. Y como debilidad quizá la falta de unos apuntes de la materia. Estamos actualmente tratando de confeccionar unos apuntes para el próximo curso.

7.-Conclusiones

Cada día las empresas y organizaciones exigen y demandan habilidades particulares a sus empleados. Ya no basta con una formación técnica sin más, sino que se valora muy positivamente las habilidades interpersonales como el trabajo en equipo, la comunicación y la

aplicación de técnicas creativas que faciliten la generación de ideas y en consecuencia la resolución de problemas. Esta generación de ideas puestas en la práctica fomentan la innovación en la empresa y permiten a la misma estar en la vanguardia de la Economía y por tanto tener una posición competitiva buena en la Economía globalizada en la que actualmente se encuentran las organizaciones. En este sentido los que suscribimos la presente comunicación planteamos la asignatura de "*Creatividad y Resolución de Problemas en empresas de TI"* con la intención de facilitar a los futuros ingenieros unas técnicas y unos modelos que cada vez se están demandando más en las organizaciones.

8.-Bibliografía Referenciada

[1] Porter, M. "Estrategia competitiva"1993.

[2] Drucker, P., "Innovation and entrepreneurship" -HARPER & ROW, NEW YORK, 1986.

[3] Cruz Tomé, M.A. "Lección Magistral Participativa". Seminario de Formación Universitaria. ICE. UPV, 2000.

[4] Fabra, Mª Ll., "Técnicas de Grupo",CEAC, 1986.

Cómo conseguir que los alumnos hagan más ejercicios

Miguel Valero-García
Dept. de Arquitectura de Computadores
Universitat Politécnica de Catalunya (Barcelona)
e-mail: miguel@ac.upc.es

Resumen

En esta ponencia se describe un método de aprendizaje cooperativo informal que se ha aplicado en una asignatura de primer año de ingeniería informática. Se analiza la forma en que el método aborda varias de las dificultades experimentadas por los profesores de la asignatura para conseguir que los alumnos hagan los ejercicios del curso. El método ha sido bien valorado tanto por los alumnos como por los profesores, aunque todavía no está suficientemente implantado como para que se observen mejoras significativas en el rendimiento académico.

1. La asignatura

El método que se describe se ha ensayado en la asignatura Estructura de Computadores 1 (de ahora en adelante EC1). Esta asignatura pertenece a la Fase Selectiva de las ingenierías informáticas que se imparten en la Facultat d'Informàtica de Barcelona. Por tanto, es una asignatura con muchos alumnos en su primer año de estudios universitarios. Por ejemplo, durante el curso 2000-2001 la asignatura ha tenido alrededor de 800 alumnos, repartidos en 3 grupos en el cuatrimestre de otoño y 6 grupos en el cuatrimestre de primavera (cada grupo tiene entre 80 y 100 alumnos).

La asignatura EC1 cubre esencialmente dos temas:

- El lenguaje máquina/ensamblador de la familia de procesadores i80X86
- El subsistema de entrada/salida de un computador

Los objetivos formativos de la asignatura son esencialmente de tipo comprensión. Es decir, los alumnos deben aplicar recetas bien establecidas para obtener la respuesta correcta (normalmente única) al problema [1]. Un ejemplo típico de ejercicio es determinar el contenido de los registros del procesador después de ejecutar una secuencia de 3 ó 4 instrucciones de lenguaje ensamblador.

La asignatura tiene tres tipos de clases:

- Clases de teoría (3 horas a la semana)
- Clases de problemas (1 hora a la semana): cada grupo de teoría se divide en dos subgrupos de problemas (por tanto, el profesor imparte dos horas a la semana, una a cada grupo).
- Clases de laboratorio (2 horas cada quince días): cada grupo de teoría se divide en cuatro de laboratorio (por tanto, el profesor imparte una media de 4 horas de laboratorio a la semana).

La asignatura EC1 se aprueba, al igual que otras muchas de nuestros planes de estudios, haciendo muchos ejercicios. Por ello, la cuestión fundamental es cómo conseguir que los alumnos hagan ejercicios. Esa es la problemática que se discute a continuación y sobre la que pretende incidir el método descrito en esta ponencia.

2. Las clases de problemas

En los últimos cuatrimestres, los profesores de la asignatura solemos pasar una encuesta al inicio del cuatrimestre en la que pedimos a los estudiantes que valoren, entre otras cosas:

1. Qué características debe tener la asignatura para que pueda ser considerada "de calidad"
2. Cuáles fueron las razones del poco éxito en la asignatura el cuatrimestre anterior (sólo para los repetidores)

Habitualmente, uno de los elementos que aparecen como respuesta a la pregunta 1 es:

> "Que haya muchos ejercicios resueltos"

Por otra parte, la respuesta que domina siempre en la segunda pregunta es:

> "No hice suficientes ejercicios"

Esta encuesta resulta útil para poner de manifiesto ante los estudiantes que lo importante es que los ejercicios los hagan ellos, y no nosotros los profesores. A nosotros, los profesores, también nos ha costado un tiempo tomar conciencia clara de las implicaciones de esta cuestión, tal y como se verá en la breve descripción que se ofrece a continuación, a cerca de la evolución de las clases de problemas.

En una primera fase de la historia de la asignatura, los profesores desarrollábamos la teoría en las clases de teoría y mostrábamos cómo se resuelven los problemas en las clases de problemas. Proponíamos ejercicios a los alumnos para que hiciesen en casa (en esas horas de trabajo personal que todo alumno debe dedicar a la asignatura). Este planteamiento (bastante generalizado en la mayoría de las asignaturas) tiene las siguientes características:

- Es eficiente, puesto que el profesor puede desarrollar mucha teoría y proporcionar a los alumnos muchos ejercicios resueltos. Por una parte, el profesor tiende a sentirse satisfecho de su trabajo, especialmente si tiene la sensación de que ha explicado con claridad. Por otra parte, el alumno se siente tranquilo con una buena colección de ejercicios resueltos en la carpeta. Puesto que los ha entendido todos, piensa que será capaz de resolver ejercicios similares cuando llegue el momento.
- Aquellos alumnos que se decidían a hacer ejercicios en casa, se encontraban solos ante las primeras dificultades. Es decir, el profesor no estaba en el sitio oportuno en el momento en que más importante es su intervención en el proceso de aprendizaje.
- El profesor no tenía ninguna realimentación antes del primer examen (un examen parcial cuando el curso está bastante avanzado), lo cual limita extraordinariamente su capacidad para intervenir.

En resumen, este planteamiento ignoraba completamente una realidad que se ponía de manifiesto cuando ya era demasiado tarde. Conviene citar aquí a John Cowan [2]:

> El trabajo del profesor consiste en crear situaciones de las que el alumno no puede escapar si haber aprendido

Es obvio que en el planteamiento descrito los alumnos estaban escapando sin aprender.

En una segunda fase, se adquirió la conciencia clara de que los alumnos debían resolver ejercicios en clase de problemas, con la presencia del profesor, para que éste pudiese intervenir ante las primeras dificultades. Por otra parte, sólo así podía justificarse que los grupos de problemas fuesen reducidos (la mitad que los de teoría). Así pues, en esta segunda fase, en las clases de problemas se proponían ejercicios que los alumnos debían resolver allí mismo, de forma que el profesor pudiese intervenir al aparecer las primeras dificultades.

De forma natural, aparecieron dos planteamientos distintos. Algunos profesores adoptaban la política de ofrecer la solución correcta al final de la clase, con lo cual muchos alumnos se limitaban a esperar a que se hiciese pública esa solución para copiarla. En una variante más radical, algunos

profesores no ofrecían en ningún momento la solución correcta. Los alumnos que quisiesen, podrían entregar el ejercicio al profesor que lo devolvería corregido a la semana siguiente. Los ejercicios entregados no serían tenidos en cuenta para el cálculo de la nota, puesto que se suponía que, en caso de tenerlos en cuenta, se produciría una avalancha de entregas que desbordaría a los profesores. Un resultado de esta variante era que muchos alumnos abandonaban las clases de problemas a las pocas semanas, para asistir a las clases de los profesores que sí daban las soluciones.

En general, las características de estas estrategias son:

- Se recorta un poco el tiempo para desarrollar la teoría, puesto que una parte de las horas de teoría deben usarse para ejemplificar la resolución de problemas (que ya no se va a hacer en las horas de problemas, como ocurría en la fase anterior).
- Se consigue que algunos estudiantes hagan más ejercicios, pero muchos alumnos siguen sin hacerlos. Como se ha indicado, en los casos en los que el profesor resolvía el ejercicio al final, muchos alumnos simplemente esperaban. Si el profesor no resolvía el ejercicio, muchos alumnos dejaban de venir a las clases. Este abandono de las clases de problemas (en torno al 50%) repercute en una mejora de las condiciones para atender a los que se quedan, pero hace que el impacto de la estrategia en el rendimiento académico sea escaso, puesto que sigue habiendo muchos alumnos que "escapan" de la situación de aprendizaje creada.
- El profesor tiene más elementos de juicio sobre la marcha de la asignatura, puesto que ve trabajar a los alumnos en clase, y corrige los ejercicios que entregan los alumnos. Es curioso observar que cuando se tienen esos elementos de juicio el profesor percibe con claridad las dificultades de aprendizaje de los alumnos, lo cual puede repercutir en su moral, cosa que no pasaba en la primera fase, cuando el profesor no tenía feedback y se creaba fácilmente la ilusión de que todo va bien.
- El trabajo de corrección de ejercicios aumenta, aunque no excesivamente puesto que, como se ha indicado antes, no son muchos los alumnos que asisten regularmente a las clases de problemas en las que no se da la solución y se recogen los ejercicios.

La descripción que acabamos de ofrecer se ha organizado en dos fases para facilitar la exposición de los dilemas y dificultades a los que pretende dar respuesta el método que se describe a continuación. No obstante, la historia real no necesariamente se ha desarrollado en dos fases bien delimitadas, y posiblemente diferentes profesores de la asignatura han vivido la historia de formas parcialmente distintas.

3. El método propuesto

El método que proponemos a continuación utiliza la estrategia de aprendizaje cooperativo en grupos informales [3]. La mecánica es la siguiente:

- El profesor propone un ejercicio que los alumnos deben resolver en clase de forma individual. El ejercicio se les reparte en un papel aparte en el que deben escribir la solución (esto aumenta la relevancia que los alumnos atribuyen a la actividad). Se les explica que en la última parte de la sesión deberán reunirse con otros compañeros para comparar las soluciones.
- Durante la resolución individual, se anima a los alumnos a que discutan el problema con sus vecinos. El profesor atiende las dudas que se produzcan.
- Al finalizar la fase de resolución individual, el profesor organiza grupos de 3 ó 4 estudiantes, de forma que los miembros de un grupo no hayan interactuado entre sí durante la resolución individual. El objetivo del grupo es ponerse de acuerdo en la solución correcta. Para ello, comparan las soluciones individuales y si hay discrepancias las aclaran. Cuando llegan al consenso, escriben la solución correcta y la firman. Si alguno de

los alumnos no está de acuerdo, no firma y puede entregar su solución particular.

- El profesor recoge las soluciones consensuadas, las corrige y las devuelve la semana siguiente.

4. Qué ha cambiado

Veamos ahora en que medida el método propuesto produce un cambio de la situación en relación a las dificultades descritas en la sección 2.

- Los alumnos hacen ejercicios en clase. Además, no se produce el abandono típico que tiene lugar cuando los alumnos constatan que el profesor no dará la solución al final de la clase. Hay dos explicaciones para esto. Por una parte, los alumnos valoran la discusión con sus compañeros (que no se producía de forma tan intensa en las fases anteriores). Por otra parte, el método se ha aplicado indistintamente en las clases de teoría y de problemas, con lo cual el planteamiento "iré a las clases de problemas de otro profesor" no tiene sentido.
- Los alumnos intentan hacer bien los ejercicios. El hecho de que cada alumno tenga que exponer su solución a los compañeros del grupo al final de la clase hace que entre en juego su motivación que "quedar bien" en el grupo, lo cual le induce a esforzarse más durante la resolución individual. Esto se traduce, por ejemplo, en un mayor número de preguntas al profesor para aclarar dudas. Además, tiende a reducirse la típica situación en la que dos alumnos trabajan aparentemente juntos el ejercicio, siendo en realidad uno de ellos el que lo resuelve y el otro el que mira como se hace.
- Durante la discusión final en grupo los alumnos se aclaran entre sí muchas de sus dudas. Se sabe que en una situación así, un compañero que se acaba de enfrentar al mismo ejercicio puede ser más eficaz que el propio profesor a la hora de resolver dudas. El beneficio de la discusión es importante tanto para el que ve su duda resuelta como para el que resuelve la duda de otro, que ve de esta forma afianzados sus conocimientos y su confianza en sí mismo. En resumen, el feedback que recibe cada alumno sobre el estado de sus conocimientos se multiplica.
- El profesor corrige un número pequeño de ejercicios (uno por cada grupo de 3 ó 4) con lo cual los alumnos pueden tener un feedback regular del profesor, a un coste moderado para éste.

Además de estas mejoras, aparecen otros elementos positivos adicionales, en virtud de la naturaleza el método utilizado:

- Los alumnos se van conociendo y relacionando a medida que se avanza en el curso, lo cual hace que el clima humano de la clase mejore notablemente.
- Los alumnos ponen en práctica y mejoran sus habilidades para discutir y alcanzar un consenso.

5. Dificultades

Las dificultades que a priori puede plantear el método propuesto aquí tienen que ver con la mecánica de la formación de los grupos, la incomodidad de las aulas para el trabajo de los grupos y el mayor tiempo que se requiere para realizar la actividad.

Tal y como se ha indicado antes, es importante que los grupos estén constituidos por alumnos que no hayan interactuado durante la fase de resolución individual. Por tanto, no vale el hacer grupos con el compañero de al lado y los dos de atrás (lo cual sería óptimo en términos de tiempo). La técnica que se ha utilizado hasta ahora es asignar un número a cada alumno, de forma cíclica, y hacer que los 3 ó 4 alumnos a los que se les ha asignado el mismo número se reúnan en un punto dado de la clase (conviene, por ejemplo, dibujar un mapa de la clase indicando dónde deben reunirse cada uno de los grupos). Obviamente, esto produce un gran movimiento de alumnos durante unos segundos. Sin embargo, una vez adquirida la mecánica, el tiempo que se pierde no es excesivo (el sistema a funcionado bien con grupos de 60 ó 70 alumnos). Por otra parte, si el método se aplica en clases de dos horas, puede

aprovecharse el descanso (en el que típicamente los alumnos se mueven de su sitio) para organizar los grupos.

Ciertamente, el mobiliario de nuestras aulas no es el más adecuado para facilitar el trabajo de grupos de más de dos personas. Al aplicar el método propuesto, muchos alumnos acababan sentados encima de los pupitres, girados hacia atrás y con los papeles encima de las rodillas. No obstante, esta circunstancia no parece haber dificultado excesivamente el trabajo, si bien es cierto que siempre que se ha aplicado el método, la capacidad del aula era superior a la ocupación en ese momento. El problema sería más grave en caso de un aula completamente ocupada. En ese caso, sin embargo, existe todavía la posibilidad de hacer que los alumnos se reúnan fuera del aula (quizá en aulas cercanas desocupadas, o incluso en pasillos y otros espacios de uso general).

Finalmente, no cabe duda de que la formación y el trabajo de los grupos hacen que el tiempo que debe destinarse a la actividad aumente en relación a estrategias como las descritas en la sección 2. Lógicamente, defendemos que es un tiempo bien empleado, pero es cierto que en el caso de temarios apretados, puede plantearse un problema de falta de tiempo. En este caso, puede adoptarse una variante que consiste en proponer a los alumnos que resuelvan el ejercicio de forma individual en casa, de forma que en clase, quizá después de unos minutos de repaso, se inicia la fase de discusión en grupo. Aunque esta variante dificulta mucho la posibilidad de que el profesor resuelva dudas individuales en el momento en que se producen, retiene aún muchas de las virtudes del método tal y como se ha propuesto en la sección 3.

6. Valoración

La aplicación frecuente y sistemática del método propuesto debería propiciar un aumento en el rendimiento académico de los alumnos (el aumento de la tasa de alumnos aprobados debería ser uno de los objetivos de cualquier mejora docente). Sin embargo, esa mejora de rendimiento no se ha constatado todavía en la asignatura EC1, precisamente porque el método no se ha aplicado de forma frecuente y sistemática. En realidad, se ha aplicado de forma puntual, con el objetivo de ensayar la mecánica, analizar la reacción de los alumnos y de los propios profesores.

Los profesores han valorado positivamente el método, esencialmente porque se consiguen unas dosis de actividad por parte de los alumnos superiores a las que se consiguen con otros métodos. La mecánica ha funcionado bien, incluso en grupos de alrededor de 60 alumnos. En todo caso, se constata la necesidad de reorganizar el programa y, en particular, recortar el temario expuesto en las clases de teoría, si es que se pretende aplicar con frecuencia el método (efectivamente, el temario acostumbra a ser una barrera para la innovación docente).

La reacción de los alumnos se ha evaluado mediante dos tipos de encuesta: el CuIC y el SEEQ.

El CuIC (Cuestionario de Incidencias Críticas) [4] es una consulta que se hace a los alumnos con una cierta frecuencia a lo largo del curso. Al final de una sesión de clase, se les pide que respondan a las dos preguntas siguientes:

> ¿Qué ha sido lo más estimulante, clarificador y positivo en el último período del curso?
>
> ¿Qué ha sido lo más desconcertante, oscuro y negativo en el último período del curso?

Los alumnos deben contestar estas preguntas de forma anónima, rápida (un minuto) y concreta.

Los resultados del CuIC ponen de manifiesto que entre los aspectos más valorados aparecen siempre las discusiones entre compañeros que se propician con el método propuesto. Por tanto, puede afirmarse que los alumnos reaccionan favorablemente.

El SEEQ (Students Evaluation of Educational Quality) es una encuesta debida a H. Marsh [5], que se usa desde comienzos de los años 80 (especialmente en el ámbito anglosajón) para recoger la opinión de los alumnos sobre el profesor y la asignatura. Es ideal para identificar aspectos a mejorar. Desde hace dos cuatrimestres, los profesores de EC1 decidimos utilizar esta

encuesta para emprender un proceso de mejora continuada, consistente en:

1. Pasar la encuesta SEEQ al final de cuatrimestre
2. Identificar los aspectos peor valorados por los alumnos
3. Seleccionar alguno de los aspectos peor valorados para mejorarlo
4. Hacer un plan de mejora
5. Llevar a cabo el plan
6. Volver a pasar el SEEQ al final del cuatrimestre y analizar el impacto de las medidas adoptadas

En el cuatrimestre 1999-2000 (2) los aspectos peor valorados por los alumnos fueron (en una escala de 1 a 5):

25.- Los comentarios del profesor sobre los exámenes y trabajo corregidos fueron de mucha ayuda **(2,93)**

13.- En este curso se animaba a los alumnos a participar en las discusiones de clase **(3,11)**

26.- Los métodos de evaluación de este curso son justos y adecuados **(3,16)**

28.- La bibliografía y el material recomendado en este curso son valiosos **(3,20)**

El equipo de profesores decidió intentar una mejora del aspecto número 13. Para ello, cada profesor se comprometió a aplicar en su clase, al menos 3 veces durante el curso, el método descrito en esta ponencia en su clase. Se esperaba que con ello los alumnos percibiesen un mayor grado de participación en clase.

El resultado del SEEQ al final del cuatrimestre 2000-2001 (1) (unos 300 alumnos), por lo que respecta al aspecto que nos interesa aquí fue:

13.- En este curso se animaba a los alumnos a participar en las discusiones de clase **(3,32)**

Se pone por tanto de manifiesto una mejora apreciable en la percepción de los alumnos. Es interesante notar (no se aportan los datos aquí) que muchos de los otros aspectos evaluados por el SEEQ recibieron una valoración peor por parte de los alumnos. Esto no es de extrañar puesto que el perfil de la población de estudiantes cambia del cuatrimestre de otoño al cuatrimestre de primavera (en particular, el porcentaje de repetidores). En estas condiciones, la mejora percibida en el aspecto objeto de nuestra actuación tiene una significación importante.

En el cuatrimestre 2000-2001 (2) (unos 500 alumnos), en el que el perfil de los alumnos es comparable al del cuatrimestre en que se tomaron los primeros datos, el resultado fue:

13.- En este curso se animaba a los alumnos a participar en las discusiones de clase **(3,43)**

Este resultado muestra una consolidación de la mejora.

Por último, cabe mencionar que el uso del método propuesto también ha tenido un impacto en los resultados de la encuesta que nuestra Universidad utiliza para evaluar a los profesores. En el caso particular del autor de esta ponencia, la encuesta institucional había dado siempre un valor de entre **3,9** y **4,1** (en una escala de 1 a 5), con independencia de los innumerables esfuerzos por mejorar la calidad de las exposiciones o de los materiales producidos para los alumnos. En el cuatrimestre 2000-2001 (2), en el que se aplicó el método con frecuencia, la valoración obtenida en la encuesta fue **4,7.**

7. Conclusión

Cuando se ha propuesto el método descrito aquí a otros compañeros, con frecuencia se ha planteado reticencias acerca de:

1. La viabilidad del método, especialmente en grupos con muchos alumnos
2. El grado de aceptación de los alumnos
3. El impacto en el nivel de aprendizaje

Nuestra experiencia ha puesto de manifiesto que las reticencias (1) y (2) no están fundadas. El método ha sido operativo en grupos de más de 60 alumnos, en una asignatura de primer año. Además, los alumnos han manifestado un grado

de aceptación superior al de otros métodos ensayados previamente.

En cuanto al impacto en el nivel de aprendizaje, tal y como se ha mencionado antes, no estamos en condiciones de afirmar nada. Sólo puede esperarse ese impacto después de un uso frecuente y sistemático del método. Esto tardará más en llegar puesto que implica una reorganización del programa y, probablemente, un recorte en el número de horas de clase expositiva.

Por otra parte, no hay que olvidar los beneficios adicionales del método, que difícilmente se pondrán de manifiesto con los sistemas de evaluación clásicos. En particular:

- Los alumnos se lo pasan mejor en clase, y eso es bueno independientemente de cuál sea el nivel de aprendizaje final
- Los alumnos desarrollan habilidades sociales importantes (discutir de forma ordenada, expresarse con claridad, alcanzar un consenso, etc.).

Finalmente, un comentario que nos parece importante en cuanto a la aplicabilidad de las estrategias de aprendizaje cooperativo. Comentando el método con algún compañero de mayor experiencia docente, éste manifestó sus dudas sobre la utilidad del trabajo en grupo en el caso de ejercicios como los de nuestra asignatura, en los que la solución es única y no hay mucho terreno para el debate ni para el planteamiento de caminos y perspectivas alternativas. La experiencia nos ha mostrado que incluso para trabajar objetivos de bajo nivel de competencia como los nuestros, el trabajo en grupos cooperativos puede tener un papel fundamental, puesto que permite que muchos alumnos puedan aclarar entre ellos sus dudas sobre la forma de resolver los ejercicios, ayudando en este propósito al profesor, que siempre es un recurso limitado cuando el número de alumnos en clase es elevado.

Referencias

[1] M. Valero-García y J.J. Navarro, Niveles de competencia de los objetivos formativos en las ingenierías. *VII Jornadas sobre Enseñanza Universitaria de la Informática JENUI2001* p.149

[2] J. Cowan *On Becoming an Innovative University Teacher,* Open University Press, 1998 Publishers 1995.

[3] D.W. Johnson, R.T.. Johnson y K.A. Smith *Cooperative Learning: Increasing College Faculty Instructional Productivity,* ASHE-ERIC Higher Education Report N. 4, George Washington University, 1991.

[4] S.D. Brookfield, *Becoming a Critically Reflective Teacher,* Jossey Bass Publishers 1995.

[5] H. Marsh y M.J. Dunkin, Students' evaluations of university teaching: A multidimensional perspective, *Effective Teaching in Higher Education: Research and Practice* (New York 1997), p. 241.

Cómo NO hacer unas prácticas de programación

Agustín Cernuda del Río

Departamento de Informática - Universidad de Oviedo
Escuela Universitaria de Ingeniería Técnica en Informática de Oviedo
Facultad de Ciencias - C/ Calvo Sotelo, S/N - 33007 Oviedo
guti@lsi.uniovi.es

Resumen

En la enseñanza universitaria de asignaturas de programación los alumnos reproducen, invariablemente, multitud de actitudes altamente nocivas. Abarcan aspectos muy variados, tanto técnicos (codificación apresurada y desordenada, despreciar los mensajes de error del compilador) como de comportamiento (no acudir a tutorías o hacerlo sólo en las fechas previas a la entrega de una práctica) como de pura deontología (copia de prácticas). A pesar de las advertencias en contra, resulta extremadamente difícil erradicar estos hábitos.

Adoptando un enfoque alternativo, quizás el uso del humor y la ironía permitan que el mensaje llegue al alumno. Además, también puede ser interesante plantearse la enseñanza *en negativo*.

1 Introducción

La comunidad de alumnos resulta con frecuencia especialmente refractaria a cierto tipo de consejos sobre la programación y la realización de prácticas, cosa perfectamente lógica si se piensa en su perfil y su juventud. No hay que olvidar, además, que muchos profesionales (incluyendo los profesores y al autor de este artículo) caen en ciertos errores una y otra vez, con lo que casi no cabe culpar al alumno (más inexperto) por cometer esos mismos errores.

¿Qué se puede hacer al respecto? Está claro que de alguna manera hay que llamar la atención del alumno sobre su propio comportamiento y predisponerlo en contra de estas actitudes. Pero los consejos en positivo no suelen funcionar. Por eso hemos decidido adoptar otro enfoque: aconsejar (de manera irónica) lo peor. Esperábamos que esto ayudase por tres razones:

- El humor, la ironía, la ridiculización, resultan mucho más amenos para un alumno típico que los consejos académicos. Cabe esperar una mayor penetración del mensaje.
- Esta ridiculización *de comportamientos* (nunca de personas) puede ser una "vacuna" eficaz; el alumno estará sensibilizado contra lo absurdo de ciertas actitudes, y esto es un poderoso factor de motivación. Si el miedo al ridículo impide a los alumnos hacer preguntas, quizás les impida cometer errores[1].
- Hay ciertas habilidades que se pueden caracterizar en gran medida a partir de lo que *no* hay que hacer (por ejemplo, cómo *no* utilizar un procesador de textos [1]). En programación, una vez que se conocen las habilidades básicas, un buen programador se distingue en gran parte por las cosas que *no* hace.

Partiendo de estas premisas, decidimos redactar un documento que recogiese los errores más frecuentes, con el fin de que un alumno pudiese identificarlos con facilidad incluso cuando los cometiese de manera inconsciente, y publicarlo en forma de página web.

[1] Queda claro que esto puede aplicarse sólo con sumo cuidado y a errores evitables de actitud, no técnicos o de comprensión; sería totalmente contraproducente (e inaceptable desde el punto de vista ético) ridiculizar *al alumno* por cometer un error con un puntero, por ejemplo.

En este artículo se incluye una versión ligeramente reducida de dicho documento. Hay que advertir que el lenguaje utilizado en el mismo, lógicamente, no responde al estilo habitual en un documento científico como el presente. No obstante, hemos decidido incluir los textos aquí sin modificaciones, puesto que alterarlos para darles un aspecto más académico desvirtuaría notablemente la utilidad de este artículo.

A continuación se incluye, pues, una selección del texto final; la versión completa está disponible en [2]. En un apartado final se ofrecerán algunas conclusiones e información sobre el efecto que hasta ahora ha tenido el documento entre el alumnado de la Escuela Universitaria de Ingeniería Técnica en Informática de Oviedo.

2 De la programación propiamente dicha

2.1 Ignora los mensajes de error

Los compiladores, los sistemas operativos, etc. emiten mensajes de error sólo para que los usen sus creadores, o para justificar sus sueldos.

Leer esos mensajes, además, lleva un tiempo precioso que se podría dedicar a escribir más código, por lo que atentan contra nuestra productividad.

Y por último, para entender esos mensajes hacen falta conocimientos informáticos -cosa que no debemos cultivar, ya que en realidad todos estamos estudiando para jefes-, y encima... ¡la mayoría de las veces esos mensajes están en inglés! No caigas en la trampa. Ignóralos.

2.2 Ignora las advertencias, "warnings" o "hints"

Al igual que en el caso de los errores, estos mensajes son difíciles de entender (por lo menos más difíciles que los mensajes cortos de los móviles), y encima suelen estar en inglés.

Es más, ignorar los "warnings" le da a uno una pátina de programador profesional que no tiene miedo de los ordenadores. ¿Qué mejor demostración de madurez como programador que presentar un programa que al compilar da decenas, no, cientos de warnings, sin que su autor se inmute? Se nota que es un programador experimentado, que no se amilana por nada, y que además está demasiado ocupado para perder el tiempo con tonterías.

2.3 Escribe el código directamente sin pensar

Al final, no nos engañemos. ¿Qué es lo que estamos construyendo? Un programa. ¿Qué es lo único imprescindible en un programa? El código. ¿Qué es lo que de verdad funciona? El código. No hay que perder ni un minuto en usar medios arcaicos como lápices, bolígrafos o papel. Tú eres un miembro de pleno derecho de la generación WAP. No haces el ridículo escribiendo costosas sílabas, ¿verdad? Pues no hagas el ridículo pensando en nada, cuando hay tanto código por escribir.

Cuanto antes te quites de enmedio este rollo de programar, mejor. Y eso, muñeco, sólo lo puedes acelerar escribiendo más y más líneas.

2.4 Aunque el código no compile o no funcione, sigue escribiendo

Es sabido que los mensajes de error son una interrupción inadmisible, una traba estúpida a nuestro trabajo. ¿Qué puedes hacer si tienes un error de compilación? Ya hemos visto que leérselo y comprenderlo no es una opción válida.

Se puede intentar hacer algún cambio aleatorio en el código fuente, a ver si hay forma de engañar a ese estúpido compilador. Pero si eso no funciona, no pierdas más tiempo. NO, no caigas en la tentación de leer el mensaje de error o intentar comprenderlo (tú tienes un prestigio al que te debes). Sigue escribiendo código, que es de lo que se trata para acabar con esta asquerosa asignatura. Más adelante, ya arreglarás el error. Es sabido, además, que los errores tienden a desaparecer solos con el tiempo si no se los mira mucho. Cuando eso pase, ya tendrás el programa casi acabado. Compilarás, ejecutarás, y si probases (que tampoco hace falta) verías que todo funciona perfectamente.

Si el código compila, eso ya es el paraíso; nada nos impide seguir escribiendo código para liquidar de una vez esta odiosa práctica. El que el programa no haga lo que se quería que hiciera no es tan importante, ya lo arreglaremos más tarde, cuando esté acabado. Además, siempre puede pasar que en un golpe de suerte los profesores cambien el enunciado de la práctica y entonces sí que encaje con nuestro programa, así que no hay que arriesgarse a hacer trabajo inútil arreglando un programa que va descarriado.

2.5 Si el código tiene un error que no se produce siempre, ignóralo y sigue escribiendo

Mejor aún que en el punto anterior. Si el error no se produce siempre, va a ser difícil de encontrar, y además puede que en la demostración de la práctica no salga, y además puede que desaparezca solo si no lo miras. Así que no te preocupes, seguro que no es nada grave.

2.6 Si el código tiene un error que se produce siempre, cambia cosas aleatoriamente hasta que desaparezca

Ya hemos hablado en contra de pararse y pensar. Si hay un error que, por algún capricho, quieres eliminar, simplemente prueba a escribir el mismo código de otras formas diversas. Esto tiene la ventaja de que a lo mejor se soluciona, y además habrás conseguido que se solucione: 1) sin entender cuál era la causa, y 2) sin dejar de escribir código. Claramente, este es el comportamiento más profesional.

2.7 Construye enormes porciones de código sin compilar / ejecutar / probar

No compiles con frecuencia; no des pasos pequeñitos. Tú eres un profesional, y tienes que dar pasos de gigante. Escribe miles de líneas de código, y ya después compila. Así será mucho más entretenido buscar los errores de compilación y arreglar el código, lo que constituye un excelente ejercicio.

Respecto a ejecutar el programa que escribes, si intentas tener siempre un programa que funcione parcialmente, descubrirás los errores muy pronto, y además al haber hecho pocas modificaciones desde la última vez, te será demasiado fácil saber dónde has introducido el nuevo error. Esto sólo lo hacen los miedicas. Un verdadero programador hace el programa entero, y luego lo digiere entero, como una boa. Nada sustituye a la maravillosa sensación de buscar un error que se oculta en las últimas 10.000 líneas de código que has escrito; si sólo son 10 ó 20, la cosa no tiene ciencia.

2.8 No escribas comentarios, salvo los obligatorios

Ya lo hemos dicho antes. ¿Cuál es el objetivo de todo esto? Hacer un programa. ¿Y de qué consta un programa? De código. Todo código que no sea ejecutable no es realmente necesario. Poner comentarios explicando algo es un insulto a la inteligencia de un programador; cualquiera que vea un programa, si conoce el lenguaje de programación, sabe perfectamente lo que hace ese programa, cómo y por qué.

Si hay comentarios obligatorios (descripciones de funciones y toda esa morralla), esos sí hay que ponerlos, aunque no se tenga nada interesante que decir. A los profesores les gustan esas tonterías y te pondrán más nota.

2.9 Ignora los enunciados

Los enunciados y las especificaciones son un rollo. Hablan de problemas que no nos interesan, son largos, prolijos, y en realidad no son más que un simple ejercicio de narcisismo de los profesores para que veamos cuánto creen que saben. Limítate a echar un vistazo, decide más o menos qué te están pidiendo, y es suficiente; seguir leyendo el enunciado, o volver a leerlo más adelante, es un obstáculo en nuestra verdadera misión. Que no es otra, por supuesto, que escribir el código. Por tanto, una vez intuido más o menos lo que se pide, entierra el enunciado en el fondo de la pila de papeles más alta que tengas, o en las profundidades del directorio temporal de tu ordenador.

2.10 Ignora las normas de programación y presentación

Las normas que dicen cómo debe escribirse el código, o cómo debe presentarse la práctica, son un ejercicio de prepotencia por parte de los profesores. A ellos les gusta dominarnos, obligarnos a hacer cosas que no tienen ningún sentido, y por eso escriben normas. No te prestes a ese juego. Todas esas normas son totalmente innecesarias, ya que aun sin ellas las prácticas entregadas cumplirán los requisitos de calidad exigibles. Respecto a la facilidad de manejo por su parte, ¿no cobran para corregirlas? No te molestes siquiera en poner tu nombre o tu curso en las prácticas, ya que en realidad no tienen mayor dificultad en recordar tu cara, tu estilo inconfundible de programación, y sabrán que la práctica es tuya de todas formas.

2.11 Escribe la documentación al final

¿Cómo vas a escribir un documento describiendo un programa que no existe? Por otra parte, ¿qué sentido tiene escribir documentos para ti mismo sobre lo que vas haciendo? ¡Todo lo que puedas leer en ellos, ya lo sabes, puesto que lo habrás escrito tú! Así que el único motivo para escribir documentación de un programa es que los profesores la piden. Y eso puede solucionarse el día anterior a la entrega. Si, además, haces el programa durante los días previos a la fecha de entrega, no tendrás el problema de que se te olvide algo y necesites mirar documentos; todo estará muy reciente en tu cabeza.

2.12 No aprendas a utilizar el depurador ni otras herramientas

Si se produce un error en tu código, al fin y al cabo todo lo que sabes sobre programación te lo han enseñado los profesores. ¿De quién es, pues, la culpa de que tú tengas un error? Es del profesor; por tanto, que sea él quien lo busque y lo solucione. Los errores de programación son algo excepcional, cuando seas profesional no tendrás que enfrentarte a ellos, por lo que en tu etapa de formación es normal que no emplees tiempo ni esfuerzo en aprender a arreglarlos.

3 De la relación con el profesor

3.1 No pidas ayuda

Si no sabes hacer algo, si tienes alguna duda, si te has perdido, nunca pidas ayuda, nunca hagas preguntas en clase, ni vayas a ninguna tutoría. Hay miles de razones para esto, pero aquí daremos sólo algunas:

- Si vas a una tutoría a hacer preguntas, estás admitiendo que eres estúpido.
- Si vas a una tutoría a hacer preguntas, les darás trabajo y te cogerán manía.
- Si no entiendes algo, la culpa es del profesor. Pero ¿y si vas y te lo explican y sigues sin entenderlo? Entonces, la culpa pasará a ser tuya, ¿podrás soportar esa vergüenza?
- Si preguntas, puede quedar en evidencia que no sabes algo que deberías saber. Es mejor continuar en la ignorancia que correr ese riesgo.
- Si preguntas en clase, no sólo es que te oiga el profesor. Es que, claro, también te oirán tus compañeros. Y es evidente que tú vas a ser la única persona en la sala que necesite esa explicación, porque los demás, que nacieron aprendidos, seguro que lo entienden. Con lo cual seguro que pasan a opinar que eres estúpido.
- Si preguntas en clase, a lo mejor el profesor te resuelve la duda. Pero claro, obtendrá una valiosa información; a partir de las preguntas de los alumnos, puede que se dé cuenta de que su explicación no ha sido buena, de que la organización del curso no es adecuada, de que está yendo demasiado rápido, o cualquier otra cosa. ¿Acaso vas a darle esta información gratis? ¿Y si se le ocurre utilizar esa información para mejorar sus explicaciones? No seas insensato, no le facilites las cosas. ¿Acaso te beneficia en algo que vaya convirtiéndose en mejor profesor?.

Conclusión: nunca pidas ayuda ni acudas a una tutoría. Es mucho mejor confiar en tus compañeros, ya que al fin y al cabo están en tu situación, mientras que el profesor es tu contrincante, tu oponente, tu enemigo.

Sólo hay dos excepciones a esta regla: puedes preguntar si te has encontrado con algún problema

y no te da la gana buscarlo. Entonces, sí está justificado que acudas al profesor para que sea él el que haga tu trabajo. Si se niega, critícale en los pasillos.

La segunda es que se puede acudir a tutorías en la semana previa a la entrega del programa. Estará lleno de gente, el profesor se dedicará en exclusiva a atender alumnos obviando otras tareas, y además si no te atiende podrás criticarlo en los pasillos. Cada vez que vayas, tendrás que esperar durante horas, y eso te permitirá relacionarte con compañeros en la cola. Todo son ventajas.

3.2 Nunca describas un problema en detalle

Si a pesar de todo decides saltarte el punto anterior, y pides ayuda, ten en cuenta una regla de oro que te será también muy útil en tu vida profesional (no en vano la siguen multitud de profesionales y usuarios de la informática). NUNCA describas un problema en detalle.

Ejemplo. Si durante el desarrollo de un programa se produce algún suceso que te resulta desagradable, acude al profesor y dile: "Ayer me pasó algo que no me agrada". El pondrá cara de esperar más datos; aguanta, no digas nada más. No se te ocurra entrar en detalles como estos:

- Si el suceso desagradable se produjo durante la compilación del programa o durante la ejecución.
- Si el suceso desagradable hizo que el programa terminase súbitamente su ejecución o bien hizo que la ejecución continuase indefinidamente, o simplemente el programa no hizo lo que esperabas que hicera desde un punto de vista funcional.

Otro ejemplo. Si el suceso desagradable se produjo durante la compilación, no le digas al profesor el mensaje de error y la línea en la que se produjo dicho error. Dile sólo algo como "Me daba no sé qué error, o algo".

Otro ejemplo. Si el suceso desagradable se produjo durante la ejecución, y provocó la súbita terminación del programa, nunca anotes el mensaje producido, ni le digas al profesor el texto del mismo. Di simplemente "Me daba no sé qué error, o algo".

Evidentemente, si el suceso desagradable consistía en que el programa no hacía lo que esperabas, no se te ocurra decirle al profesor en qué situación exacta lo producía (si era al leer un fichero vacío, o antes o después de que pasase alguna otra cosa). Evita una descripción como "El error se produce siempre que cargo un segundo fichero y el primero estaba vacío". Tú di sólo la frase mágica: "Me daba no sé qué error, o algo".

¿Lo vas pillando?

3.3 Lleva siempre los fuentes equivocados

Supongamos que, contra todos nuestros consejos anteriores, acudes al profesor, y además vas a preguntarle por un problema concreto. Si el profesor se niega a examinar tus fuentes, podrás criticarle en los pasillos y advertir a futuros alumnos contra él, pero puede darse el caso de que el tío te reciba y tú acabes entrando y preguntando. Haces mal, pero la situación aún tiene arreglo: procura llevar los fuentes equivocados. Por ejemplo, si tienes un problema, y haces diversas modificaciones que no dan resultado, o incluso generan problemas nuevos, acude con los últimos fuentes, pero pregunta por el problema original. Así, el profesor buscará inútilmente un error cuando se le producirá otro. Entonces di algo como "Ah, bueno es que después probé una cosa. Quita esa línea de ahí..."

Si manejas este proceso con habilidad, puedes realizar toda una sesión de codificación allí mismo, en las tutorías. Esto es particularmente aconsejable si hay otros compañeros esperando fuera. El profesor cogerá mala fama, trabajará más tiempo, harás que el resto de los alumnos tengan más trabas para avanzar, con lo que el nivel de la clase se mantendrá en cierto equilibrio y así no pensarán en ampliar el temario en futuros años... Tómatelo como un servicio a la comunidad.

3.4 No aísles el problema

Volvamos a suponer que eres un irresponsable y acudes a una tutoría. No se te ocurra aislar el problema antes de ir. Si se produce un error en un fichero de entrada de 1 MB, no intentes ir probando con ficheros más pequeños hasta acotar qué produce el error, ni intentes crear un mini-

programa que reproduzca el mismo error. Eso podría permitir al profesor averiguar el problema de manera certera y rápida. Es mucho mejor que se lea miles de líneas de código y que haga trazas con centenares de miles de pasos. De este modo, ejercitará notablemente sus artes adivinatorias y podrás verificar su capacidad de deducción. Por supuesto, si se niega a buscar tu error, critícale en los pasillos.

3.5 Usa el correo electrónico con habilidad

Hay preguntas que son prácticamente imposibles de resolver por correo electrónico, si se hacen bien. Cultiva este arte. Haz que tu pregunta sea totalmente inconcreta. Ejemplo: "Me da no sé qué error, o algo. Aquí te mando el fuente". También puedes hacer una pregunta un tanto más concreta, pero no mandar el fuente: "En mi clase TDispositivo en el constructor me da no sé qué error, o algo". Si el profesor no resuelve el problema, critícalo en los pasillos por incompetente.

Por supuesto, escribe el mensaje de corrido y mándalo; no lo leas por segunda vez intentando ponerte en el lugar del profesor. Eso te podría llevar a detectar errores, y no es eso lo que se pretende.

3.6 Eskríbelo todo cn abrvtrs o konsonantes ekstrañas

Los profesores son unos puretas. Tú eres generación WAP. O si no, al menos serás un poko radikal. Intenta escribir mensajes que no resulten fáciles de leer. El tío, aunque crea que no, tendrá que hacer un esfuerzo extra, y además en la pantalla de un ordenador, ante la que quizás lleve varias horas. Eso facilitará su concentración y aumentará su buen humor.

3.7 Comete faltas de ortografía

La ortografía es una burda convención; hasta Juan Ramón Jiménez ponía jotas donde le daba la gana, y hasta Gabriel García Márquez abogaba por eliminar la ortografía. Evidentemente, tú no eres menos que ellos, así que tienes el mismo derecho a escribir como quieras.

¿Que no tienes costumbre de cometer faltas de ortografía? Es bien fácil. Puedes empezar por el ejercicio más fácil, más frecuente y más provechoso: "Haber si me sale". Ya, ya sé; la forma correcta es "A ver si me sale" (como diciendo "veamos si me sale"), pero ¿no encierra una brutal poesía esa brusca contracción de dos palabras en una, esa alteración semántica de utilizar el verbo "haber" sin significado alguno, esa hermosa palabra que arroja simultáneamente una B y una H (las letras más peligrosas de la ortografía española) a los ojos del lector? Si te apetece darle al profesor una bofetada y no tienes arrestos, escríbele un "Haber si puedo ir mañana a tutorías", que es lo más parecido.

3.8 No te identifiques

El correo electrónico tiene otra cosa divertida. Puedes empezar a largar sin que el tío sepa ni de qué grupo eres, ni quién eres. Todo se arreglará si en ese caso lo tuteas, porque así aumenta la familiaridad y no hace falta el nombre. Mejor aún es escribir un mensaje en el que no sepa ni de qué titulación eres. Sí, por ejemplo: "¿Cuál es la fecha de entrega para la primera práctica? Firmado: Fefu". Si el profesor da más de una asignatura en diferentes carreras (cosa extraordinariamente probable), será un buen ejercicio para él coger las listas de alumnos de todas ellas y buscar algo que pueda casar con "Fefu".

4 Y, sobre todo...

4.1 Deja el trabajo para el final

Desde el primer día, el profesor insistirá en intentar pedir cosas para la semana siguiente. Intentará avisar de que hay que ir haciendo el trabajo a un ritmo constante y decidido desde el principio.

No le escuches.

La programación de ordenadores, aunque es una disciplina joven, tiene ya sagradas tradiciones, y una de ellas es la prisa en los días previos a una entrega. Evitar ese estrés sería una nefasta preparación para tu vida profesional. Deja que el

trabajo se acumule, desoye las advertencias o los signos de retraso. No interrumpas tu vida ni dejes de ir a esquiar para recuperar tiempo perdido. Y cuando ya estés al límite del desastre, cuando falten dos semanas para la entrega de un programa estimado para cuatro meses... entonces empieza a escribir código como si no hubiera un mañana.

¿Qué aliciente tendría esta profesión si los programas se construyeran sin prisa y sin pausa? ¿Qué sería de la imagen del melenudo barbudo (o melenuda barbuda) maloliente (por no tener tiempo para afeitarse / cortarse las puntas / ducharse) con una camiseta de Megadeth (en las camisetas jeviatas se ven menos las manchas, gracias a su color negro y a los complejos dibujos), que lleva 48 horas ininterrumpidas dándole a la tecla? ¿Estarías acaso preparado para ir a la Campus Party y matar monstruos en una pantalla, con el culo sobre una silla de melamina y la cabeza bajo un techo de lona a 35ºC, durante tres días seguidos? ¿Cómo podríamos sentirnos un poquito héroes si todos los días nos vamos a dormir cuando nos entra el sueño? ¿Qué sería de las fábricas de Cocacola, qué sería de Juan Valdés, qué sería de las refinerías de cafeína que destinan la mitad de su producción a los programadores? ¿Acaso Sandra Bullock o Robert Redford cuando hacían de hackers se sentaban con sus apuntes al lado del ordenador, pensaban, tecleaban sin prisa y a la hora habitual se iban al gimnasio o al bar de la esquina, y así un día tras otro durante cuatro meses? ¿Acaso el tío ese de "Operación Swordfish" habría roto la clave del Pentágono si no hubiera estado un esbirro del Travolta apuntántole con una pistola mientras otra esbirra del Travolta lo distraía?

No, amigo, no. Para vivir tranquilo, te habrías matriculado en otra carrera. ¿Te imaginas ir a clase y que cuando el tío diga "Quién tiene hecho nosequé" tú puedas decirle "yo", y que cuando explique lo siguiente que hay que implementar tú estés atendiendo y entendiendo lo que dice porque lo anterior ya lo tienes funcionando?

Eso es para empollones y flojos de espíritu. Ya lo sabes. Deja todo el trabajo para el final.

4.2 Copia las prácticas

Copia las prácticas; ya sean las actuales por un compañero de clase, o prácticas del año pasado. Cada profesor puede tener que corregir varias decenas de prácticas; es difícil reconocer similitudes entre tantos programas. Si las reconocen, además, no es fácil demostrarlo, recurre y lleva el caso hasta el Tribunal Constitucional. Te llevará muchísimo más tiempo y esfuerzo que hacer las prácticas, pero el caso es demostrar que eres más listo que el profesor y no dar nunca el brazo a torcer.

La sensación de ahorro de trabajo que da copiar una práctica (o parte de ella) es inigualable y merece la pena. No importa que luego quedes en evidencia en la demostración, o en el examen práctico, o que en el fondo no hayas aprendido lo necesario, ya que el objetivo de las prácticas no es el aprendizaje, sino sólo la obtención de unos asquerosos créditos. Por tanto, copia lo que puedas. Si un profesor te suspende por copiar, critícale en los pasillos.

5 Efectos del documento en la comunidad

La dirección de la página web que contiene los “anti-consejos” fue comunicada, vía correo electrónico (un viernes), a los alumnos de 3 de los 14 grupos de prácticas de la asignatura Estructuras de Datos y de la Información, después de que presentasen el primer módulo de prácticas de esta asignatura. En el número del lunes siguiente del boletín informativo electrónico editado por los alumnos, InfoEUITIO [3], que cuenta con unos 300 suscriptores, apareció una reseña del documento; y varios profesores de otras asignaturas han sugerido su lectura. Desgraciadamente, esta acogida no estaba prevista y no hay estadísticas de acceso a la página, pero cabe suponer que el documento ha alcanzado un notable grado de difusión. El autor ha recibido varios mensajes al respecto, tanto de alumnos como de profesores.

En especial, resulta curioso apreciar que la respuesta de los alumnos es muy positiva; se leen el documento, les gusta, no se sienten ofendidos

en absoluto, admiten sin paliativos que se ven retratados en gran medida, y han contribuido mucho a su difusión. Un alumno sugería que, dado que InfoEUITIO no llega a todos los alumnos (sólo a los que se suscriben), debería enviarse el enlace a las listas de correo de las diversas asignaturas (Metodología de la Programación I y II, Estructuras de Datos y de la Información, etc.)[2]

Es pronto para saber si realmente estas instrucciones tendrán algún impacto en la calidad del trabajo que desarrollan los alumnos. Pero sí podemos concluir que este estilo de presentación de la información ha permitido alcanzar un sorprendente grado de penetración y una respuesta extremadamente positiva.

Referencias

[1] Agustín Cernuda del Río. *Informática - Microsoft Word (II). Cómo NO utilizar un procesador de textos*. Noviembre de 2001.

[2] Agustín Cernuda del Río. *Cómo NO realizar una práctica de programación*. http://www.agustincernuda.com/noprog.html

[3] Delegación de alumnos de EUITIO. http://petra.euitio.uniovi.es/~delegaci

[2] De hecho, su difusión se ha extendido a entornos profesionales externos a la Escuela, y es difícil conocer en este momento hasta dónde alcanza.

La autoevaluación como método de aprendizaje

Daniel Gayo Avello[1] Hortensia Fernández Cuervo[2] Fernando Torre Cervigón[1]

[1] Dpto. de Informática
Universidad de Oviedo
C/Calvo Sotelo s/n 33007 Oviedo
e-mail: {dani,torre}@lsi.uniovi.es

[2] Licenciada en Psicopedagogía
e-mail: tensi_f@hotmail.com

Resumen

Algorítmica y Lenguajes de Programación es una asignatura troncal impartida en la licenciatura de Matemáticas de la Universidad de Oviedo. Aunque algunos alumnos tienen un interés especial por la informática en general y por la programación en particular, muchos estudiantes encuentran la asignatura demasiado compleja. Esta circunstancia es especialmente acentuada en el estudio de la teoría, señalándose estos contenidos como la principal dificultad de la asignatura.

Partiendo de esta situación, los autores decidieron explorar nuevas vías para facilitar a los alumnos el aprendizaje de los contenidos teóricos. Así, se analizó la factibilidad del uso de herramientas de autoevaluación *online* como sistema de autoaprendizaje. Para ello, se seleccionaron dos tipos de herramienta distintos y se llevó a cabo un experimento de investigación educativa a fin de determinar, por un lado, las posibilidades de la autoevaluación como sistema de autoaprendizaje y, por otro, las posibles ventajas de una herramienta sobre la otra.

1. Introducción

La asignatura de Algorítmica y Lenguajes de Programación es una asignatura troncal de doce créditos que se imparte en primer curso de la licenciatura de Matemáticas de la Universidad de Oviedo. La mitad de los créditos totales se dedican a contenidos teóricos, repartiéndose los seis restantes entre prácticas de tablero y prácticas de laboratorio.

Aunque algunos de los alumnos matriculados en la asignatura ya han tenido contactos con la informática, para la mayor parte de los estudiantes la asignatura supone su primer acercamiento a los ordenadores, resultándoles relativamente compleja.

En las primeras clases se presentan los conceptos fundamentales (algoritmo, procesador, acción, variable, etc.). Sin embargo, pronto la materia alcanza un mayor grado de dificultad (estructuras de control, subprogramas, recursividad, etc.) y son muchos los alumnos que acusan el cambio.

Conociendo la dificultad de la asignatura para alumnos de primer curso y conscientes de la importancia que la motivación tiene en el proceso de enseñanza-aprendizaje, se desarrolla una página web[1] de la asignatura. A través de dicha web los alumnos pueden acceder a transparencias, apuntes, enunciados y soluciones de ejercicios y exámenes así como a un foro de discusión.

Cuando habían transcurrido unos tres meses del curso se pasó una pequeña encuesta a los alumnos a fin de determinar cómo percibían la asignatura, al profesorado, al grupo clase y a sí mismos. Las principales conclusiones que se extrajeron de este estudio fueron:

1. Los alumnos percibían de forma positiva tanto al profesorado como a la asignatura.
2. Consideraban la asignatura importante e interesante y al mismo tiempo difícil, especialmente la parte teórica.
3. Los alumnos afirmaban estudiar poco o nada, sobre todo la teoría de la asignatura.

Tales resultados causaron cierta perplejidad

[1] www.di.uniovi.es/~dani/asignaturas/index.html#alp

entre los profesores de la asignatura y les animaron a plantearse formas de abordar el principal problema que traslucía en la encuesta: el escaso estudio de los contenidos teóricos.

2. Planteamiento del problema

Simultáneamente al desarrollo de la encuesta anterior se había planteado la posibilidad de ofrecer algún tipo de sistema de autoevaluación mediante pruebas objetivas en la web de la asignatura. De esta manera, los alumnos dispondrían de un instrumento que les permitiría determinar de forma sencilla el grado de aprendizaje que estaban alcanzando.

Este planteamiento, aunque sencillo, planteaba serias dudas. En primer lugar, podía hacer creer, equivocadamente, a los alumnos que la asignatura sería evaluada mediante el uso de tales pruebas objetivas. En segundo lugar, a no ser que las preguntas de tales pruebas fueran modificadas con cierta frecuencia, se podría producir un simple aprendizaje de respuestas en lugar de un aprendizaje significativo de los contenidos. Por último, al tratarse de pruebas cerradas sin posibilidad de explicación alguna sobre la corrección o incorrección de las alternativas podría, en el mejor de los casos, disuadir al alumno de su uso y, en el peor, frustrarle en el estudio de la asignatura.

Por esas razones se decidió buscar opciones alternativas a las pruebas de respuesta múltiple. Tras un período de búsqueda bastante exhaustivo se encontró una herramienta de autoevaluación con un enfoque diferente: *Duck*[2].

Duck es una herramienta desarrollada por el *Biology Computer Resource Center* de la Universidad de Massachusetts Amherst. Básicamente se trata de un conjunto de *scripts* PHP[3] que permiten implementar un sistema de "cursos". Un curso dispone de una o más áreas incluyendo cada área varias preguntas.

Hay tres aspectos fundamentales que diferencian a *Duck* de una herramienta de respuesta múltiple: el tipo de preguntas, la evaluación y la navegación. *Duck* admite preguntas de respuesta múltiple, respuesta corta y de respuesta extensa (estas últimas pueden ser enviadas al tutor para su revisión). Por otra parte, en *Duck* no se otorga ninguna puntuación a las respuestas puesto que éstas no son ni correctas ni incorrectas; en su lugar, cada respuesta tiene un *feedback* que la matiza y sirve para resolver las posibles dudas del alumno. Por último, en *Duck* no es necesario seguir una navegación lineal, el usuario puede ir a las preguntas que más le interesen, explorar todas las respuestas, retroceder a preguntas anteriores, etc.

Estas características hacían de *Duck* una posibilidad interesante de cara a desarrollar una herramienta de autoevaluación que aportara un valor añadido para los estudiantes.

Así pues, al plantearse la necesidad de explorar nuevas formas de apoyar el aprendizaje de la teoría de la asignatura se vio la posibilidad de emplear herramientas de autoevaluación para tal fin, así como la obligación de analizar tanto herramientas tradicionales de respuesta múltiple como *Duck*. De esta forma sería posible determinar si el uso de tales instrumentos podía favorecer en alguna medida el aprendizaje y, en caso afirmativo, si existía un planteamiento más adecuado que otro.

3. Antecedentes

3.1 Evaluación y aprendizaje

En la literatura se encuentran múltiples referencias a la influencia de la evaluación en el aprendizaje; sin embargo, aunque todas constatan una mejoría en el aprendizaje de los alumnos, ninguna proporciona pruebas que permitan establecer una vinculación inequívoca entre ambos fenómenos.

Por ejemplo, [1] describe la utilización de una herramienta de respuesta múltiple como sistema de evaluación *online* así como los positivos efectos que tuvo en los resultados de los estudiantes; sin embargo, el objetivo de la aplicación no era el aprendizaje sino la evaluación del alumnado y los autores no hacen ninguna afirmación categórica sobre la relación existente entre evaluación y aprendizaje.

La referencia [2] trata una experiencia similar: el empleo de una herramienta de evaluación por ordenador como sistema de evaluación formativa; nuevamente, no se proporcionan datos que

[2] http://bcrc.bio.umass.edu/projects/duck/
[3] http://www.php.net

permitan vincular evaluación y aprendizaje.

Por lo que respecta a iniciativas en el ámbito español, [3] presenta algunas conclusiones sobre la utilización de una aplicación informática de autoevaluación con refuerzo; sin embargo, aunque los resultados parecen positivos son puramente cualitativos y, por tanto, difíciles de valorar.

Así pues, el presente artículo pretende aportar algo más de luz a este interesante método de aprendizaje mediante un estudio riguroso que vincule de forma inequívoca la influencia de la (auto)evaluación en el (auto)aprendizaje.

3.2. Aprendizaje operante vs. significativo

En este apartado explicaremos brevemente dos tipos de aprendizaje que aparecen implícitos en las herramientas de autoevaluación: el aprendizaje operante [4] y el significativo [5][6].

Podemos decir que aprendizaje operante es aquel en el que la conducta, lo que un individuo hace o dice, se produce en función de sus consecuencias (p.ej. aparcar en lugares permitidos para evitar las multas).

Entendemos, en cambio, por aprendizaje significativo el que se logra al establecerse relaciones entre los conocimientos previos del alumno y los nuevos conocimientos (p.ej. al explicar el cálculo del área de un triángulo a partir de la superficie de un rectángulo).

Al aplicar estos conceptos teóricos al experimento descrito, encontramos que la herramienta tradicional de respuesta múltiple se encuadraría dentro del aprendizaje operante puesto que cada vez que el alumno responde a una pregunta la herramienta le indica si la respuesta es correcta o no provocando el refuerzo o la extinción de la misma. *Duck*, en cambio, se adapta a un aprendizaje significativo puesto que el alumno, al responder, no se encuentra una "evaluación" sino una serie de argumentos sobre la pregunta y la respuesta seleccionada que le permiten elaborar asociaciones que le llevan a cerciorarse sobre la solución más adecuada.

4. Formulación de hipótesis

Así, se planteaban dos cuestiones: en primer lugar debía determinarse si el uso de herramientas de autoevaluación *online* mejoraba el aprendizaje de los alumnos; en segundo, había que comprobar si una aplicación con retroalimentación (aprendizaje significativo) era superior a instrumentos de respuesta múltiple (aprendizaje operante). Así, el experimento debía tratar de demostrar las siguientes hipótesis:

- La utilización de herramientas de autoevaluación en el web mejora el aprendizaje por parte de los alumnos de conceptos teóricos de programación.
- La herramienta de autoevaluación *Duck* resulta más útil para los alumnos que herramientas de respuesta múltiple.

5. Metodología

5.1. Identificación de las variables

Para el desarrollo del experimento se seleccionaron únicamente dos variables: la herramienta de autoevaluación constituiría la variable independiente mientras que el rendimiento académico en la parte teórica sería la variable dependiente; el principal objetivo de la investigación sería determinar la influencia de la primera sobre la segunda.

5.2. Población y muestra

La población objeto del estudio fueron los matriculados en la asignatura de ALP en la Licenciatura de Matemáticas de la Universidad de Oviedo en el curso 2001/2002.

Sobre esta población se llevó a cabo un muestreo aleatorio estratificado[4], para ello se dividió la población en distintos estratos teniendo en cuenta el sexo[5] de los alumnos y el número de ocasiones en que el alumno estuvo matriculado en la asignatura.

---

[4] El muestreo aleatorio estratificado implica dividir la población en subgrupos homogéneos y disjuntos para tomar una muestra aleatoria de cada subgrupo.

[5] En varias referencias [7][8][9] se afirma que mujeres y hombres se enfrentan a los ordenadores de forma diferente, no mejor o peor, sino distinta; aun cuando es muy posible que dichas diferencias no se deban tanto al sexo del usuario como a estereotipos sociales asociados a cada género, los autores consideraron una medida preventiva garantizar el mismo número de sujetos de cada sexo en cada grupo.

Así, resultó que la población estaba formada por 23 mujeres y 19 hombres. Además, exceptuando tres personas que se matriculaban por segunda vez en la asignatura, el resto formalizaban matrícula por vez primera. Este hecho también categorizaba a los alumnos por edad, 18 años para los de primera matrícula y 19 para los de segunda.

Una vez determinados los estratos se seleccionaron de forma aleatoria un número fijo de individuos de cada estrato asignándolos a los grupos 1, 2 y 3. La determinación de la naturaleza de cada grupo (control o experimental) también se decidió de forma aleatoria quedando vinculados de esta forma: 1-control, 2-duck y 3-tradicional.

Al término de este proceso se disponía de una muestra de 24 sujetos distribuidos en tres grupos en los que había 4 mujeres y 4 hombres, contando cada grupo con un alumno repetidor.

Durante el desarrollo de la investigación, se produjo muerte experimental[6] en los tres grupos (en ningún caso afectó a los repetidores), finalizando la investigación con 6 sujetos en el grupo de control y 7 en cada grupo experimental.

5.3. Diseño del experimento

El experimento es un diseño multivalente pretest-postest en el que hay un grupo de control y un grupo experimental para cada una de las herramientas de autoevaluación (grupos duck y tradicional). Cada grupo consta de 8 individuos cuya asignación ha sido aleatoria. Los tres grupos realizarían un pretest y un postest. El grupo de control no recibiría tratamiento mientras que los grupos experimentales recibirían dos niveles distintos de la variable independiente: el grupo duck utilizaría la herramienta *Duck* y el grupo tradicional la herramienta de respuesta múltiple.

5.4. Procedimiento

Las sesiones experimentales tuvieron lugar en una de las salas de ordenadores de la Facultad de Ciencias de la Universidad de Oviedo durante tres viernes consecutivos en horario de mañana.

Durante la primera sesión se pasó un cuestionario a los alumnos para determinar su actitud hacia la asignatura[7]; posteriormente, se les administró un pretest para precisar su rendimiento académico en la parte teórica de ALP con anterioridad al experimento. Dicho pretest consistió en un cuestionario de 20 ítems relativos a los temas "Funciones y subrutinas" y "Recursividad", cada ítem presentaba 4 posibles respuestas siendo una sola válida. En la evaluación del cuestionario se penalizaba el uso del azar como método de respuesta otorgando 1/2 punto por cada acierto y -1/6 a cada fallo.

A los alumnos se les notificó la forma en que se iba a puntuar dicha "prueba" aunque no se les informó[8] de la naturaleza del experimento; se les hizo creer que participaban en un programa de evaluación de la calidad de la enseñanza a fin de que las respuestas al pretest no se vieran condicionadas por un ambiente de "examen".

En esta primera sesión, justo a continuación del pretest, los grupos duck y tradicional recibieron su primera sesión de tratamiento; recibiendo la segunda el viernes siguiente.

El tratamiento recibido por el grupo tradicional consistía en la utilización durante una hora de una herramienta similar a una prueba objetiva de respuesta múltiple; dicha herramienta disponía de 40 ítems[9] con 4 respuestas cada uno y

[6] La mortalidad experimental es la pérdida de participantes a lo largo de un experimento con grupos.

[7] El instrumento utilizado fue una escala de Likert; un método donde las opiniones sobre un concepto varían desde el "total desacuerdo" hasta el "total acuerdo".

[8] El comportamiento de un individuo, en particular su productividad, puede cambiar considerablemente si es consciente de que se le observa (efecto Hawthorne); este hecho fue detectado por vez primera en una investigación realizada en una fábrica de Hawthorne (Illinois, EEUU) que pretendía determinar el efecto de la iluminación sobre la productividad. La productividad se incrementaba siempre, independientemente del aumento o disminución de la intensidad lumínica; por ello, se llegó a la conclusión de que el hecho de saberse observados era lo que afectaba a la productividad de los trabajadores. Por otro lado, si los individuos saben que pertenecen a un grupo determinado, en especial al de control, pueden producirse efectos de desmoralización (puesto que no se espera mejoría por su parte) o rivalidad (al esforzarse mucho más que de costumbre para "demostrar" al investigador que son "mejores" que los grupos tratados).

[9] Los ítems de las herramientas de autoevaluación, al igual que el pretest y el postest, trataban los temas "Funciones y Subrutinas" y "Recursividad"; un 25% de los ítems de las herramientas aparecían en el pretest y en el postest.

permitía al usuario saber si había "acertado" o "fallado" en su respuesta así como calcular una puntuación total o ver las respuestas correctas.

El grupo duck utilizaba, durante el mismo período de tiempo, una herramienta similar a la del grupo tradicional puesto que disponía de los mismos ítems y respuestas. Sin embargo, presentaba importantes diferencias respecto a la herramienta tradicional; en primer lugar las respuestas no eran correctas o incorrectas sino que su selección proporcionaba una explicación acerca de su validez; además, el alumno no obtenía ningún tipo de puntuación.

Al finalizar la primera sesión se pasó a los alumnos de los grupos duck y tradicional un cuestionario para determinar el grado de satisfacción respecto a su herramienta.

Para terminar el experimento, una semana después de la conclusión del tratamiento se pasó a los alumnos de los tres grupos un cuestionario[10] para determinar el trabajo personal que habían dedicado a la asignatura durante el mes en que se había llevado a cabo la experiencia. Además, se administró a los tres grupos un postest que contenía los mismos ítems que el pretest aunque tanto las preguntas como las respuestas a cada pregunta aparecían en un orden distinto.

En la Tabla 1 se muestra la forma en que se llevaron a cabo las distintas sesiones.

1º Viernes	09.00	Pretest grupos tradicional y duck.
	09.40	1º Tratamiento grupos tradicional y duck.
	11.00	Cuestionario satisfacción grupos tradicional y duck.
	11.15	Salida grupos tradicional y duck. Pretest grupo control.
2º Viernes	09.00	2º Tratamiento grupos tradicional y duck
3º Viernes	10.00	Cuestionario para determinar trabajo personal grupos duck y tradicional.
	10.15	Postest grupos duck y tradicional.
	10.45	Cuestionario para determinar trabajo personal grupo control.
	11.00	Postest grupo control.

Tabla 1. Temporalización de las sesiones

[10] El instrumento utilizado fue una escala de Thurstone.

6. Discusión de los resultados

Mediante el experimento se obtuvieron datos para determinar la actitud de los alumnos ante la asignatura, los conocimientos teóricos antes del tratamiento (pretest), los conocimientos teóricos tras el tratamiento o en ausencia de éste (postest), así como el esfuerzo dedicado a la asignatura.

A partir de dichos datos se debía determinar si existían o no diferencias significativas entre las puntuaciones medias obtenidas por los distintos grupos antes y después del tratamiento. Si las diferencias eran estadísticamente significativas y los tres grupos eran aptitudinal y actitudinalmente equivalentes entonces dichas diferencias sólo podrían ser atribuidas al tratamiento. Para el análisis de datos se usó el paquete SPSS.

En primer lugar, hubo que determinar si los grupos eran equivalentes en cuanto a su actitud hacia la asignatura, sus aptitudes (pretest) y el trabajo personal realizado durante las semanas que llevó el experimento. Para comprobar esto se realizó un ANOVA[11] de un solo factor para cada una de las medidas: actitud, aptitud y trabajo personal.

A continuación se debía determinar si la situación del grupo de control en el postest era similar a la mostrada en el pretest, así como comparar los grupos experimentales con dicho grupo de control y con su situación original en búsqueda de mejorías. Todas estas comparaciones se limitan, básicamente, a contrastar las puntuaciones medias obtenidas por los distintos grupos; para ello se emplea la prueba T de Student[12].

Por último se tenía que comprobar si el grupo

[11] Análisis de la varianza.

[12] La prueba T permite comparar muestras independientes entre sí (p.ej. las notas de dos grupos distintos) o relacionadas (p.ej. las notas de un mismo grupo antes y después de un tratamiento) y determinar si las diferencias entre las medias son significativas o no. Este contraste debe aplicarse de forma unilateral si se pretende determinar una mejoría o superioridad y de forma bilateral si sólo se quieren determinar diferencias entre las muestras. Tanto ANOVA como la prueba T requieren que las muestras sigan una distribución aproximadamente normal, para garantizar este supuesto puede emplearse el test de normalidad de Shapiro-Wilk; asímismo, es necesario que las varianzas de las muestras sean homogéneas, la prueba de Levene suele utilizarse para esta verificación.

duck (que, recordemos, siguió un aprendizaje significativo) había obtenido mejores resultados que el grupo tradicional (aprendizaje operante).

En aras de la brevedad los resultados numéricos se presentan en un apéndice limitándonos aquí a discutir las conclusiones que se pueden extraer de los mismos.

Como se puede apreciar en la primera parte del análisis estadístico, las puntuaciones alcanzadas por los tres grupos en todas las medidas (pretest, postest, actitud y trabajo) seguían distribuciones normales y presentaban varianzas homogéneas; de esta forma se cumplían los requisitos para aplicar tanto ANOVA como la prueba T de Student.

La aplicación de ANOVA a los resultados obtenidos en el pretest, prueba de actitud y nivel de trabajo mostraron que los tres grupos eran equivalentes en todos los aspectos; es decir, sus aptitudes y actitudes hacia la asignatura eran equivalentes y durante el período de tiempo que duró el experimento los tres grupos estudiaron la asignatura en la misma medida (más bien poco).

Esto quiere decir que las diferencias existentes entre las puntuaciones obtenidas en el postest no serían achacables ni a diferencias aptitudinales o actitudinales ni a un esfuerzo diferente por parte de los alumnos, tan sólo a la única variable que los diferenciaba: la herramienta de autoevaluación empleada.

La aplicación de la prueba T de Student demostró que, en el postest, los grupos duck y tradicional presentaron una mejoría significativa respecto al grupo de control y respecto a su situación inicial en el pretest; por el contrario, el grupo de control, aunque empeoró ligeramente no lo hizo de forma significativa (es decir, permaneció sin cambios).

Así pues, queda demostrada nuestra primera hipótesis: la utilización de herramientas de autoevaluación en el web mejora el aprendizaje por parte de los alumnos de conceptos teóricos.

Por lo que respecta a la superioridad de *Duck* frente a herramientas tradicionales, no se demostró, puesto que la diferencia entre los dos grupos experimentales no fue significativa. Sin embargo, esto no quiere decir que ambas herramientas sean equivalentes, sería necesario realizar más experiencias a fin de demostrar o refutar definitivamente la hipótesis.

En las gráficas que se proporcionan también en el apéndice se pueden apreciar de una forma muy intuitiva los resultados del experimento.

7. Conclusiones y trabajo futuro

A la vista de los resultados se puede afirmar, sin lugar a dudas, que la utilización de herramientas de autoevaluación en el Web facilita a los alumnos el aprendizaje de conceptos teóricos de asignaturas de programación.

La cuestión de si es preferible emplear una herramienta de respuesta múltiple o, por el contrario, resulta más adecuada una herramienta que proporcione un *feedback* al alumno aún queda abierta.

Es cierto que el grado de satisfacción fue mayor en el caso del grupo que utilizó el segundo tipo de herramienta, *Duck*, que en aquellos alumnos que emplearon una herramienta tradicional. Sin embargo, los resultados obtenidos en el postest por ambos grupos fueron estadísticamente equivalentes por lo que no se puede afirmar que haya diferencias significativas en cuanto al aprendizaje adquirido mediante el uso de una u otra herramienta.

No obstante, es preciso señalar que los estudiantes utilizarán estos instrumentos de una forma totalmente voluntaria; por ello, es posible que en una aplicación real sí existan diferencias cuantificables en los resultados obtenidos por los alumnos en función de la herramienta elegida. Por otro lado, habrá que tener en cuenta que iniciativas como la descrita tienden a ser aprovechadas únicamente por los alumnos más destacados [10].

Por tales razones, una vez demostrada la utilidad de las herramientas de autoevaluación como sistemas de autoaprendizaje, los autores pretenden llevar a cabo una nueva investigación que determine en una situación real cuál de los dos enfoques es más adecuado.

Referencias

[1] Phil Davies. *Learning Through Assessment: OLAL. On-Line Assessment and Learning*. Conference Proceedings -Flexible Learning- 3rd Annual Computer Assisted Assessment Conference, Loughborough, UK. 1999.

[2] Leith Sly, Léonie J. Rennie. *Computer managed learning: Its use in formative as well as summative assessment*. Conference Proceedings -Flexible Learning- 3rd Annual Computer Assisted Assessment Conference, Loughborough, UK. 1999.

[3] J.M. Serra, Rafael Serra Simal. *Autoevaluación con refuerzo, como herramienta informática individual de apoyo en el aprendizaje*. III Congreso Edutec. 1997.

[4] Burrhus F. Skinner. *La conducta de los organismos*. Edición en español de *The Behavior of Organisms: An Experimental Analysis*. 1938/1975.

[5] David P. Ausubel. *Psicología educativa: un punto de vista cognoscitivo*. Edición en español de *Educational Psychology. A cognitive View*. 1976/1968.

[6] Mario Carretero. *Constructivismo y educación*. 1993.

[7] Janet Carter, Tony Jenkins. *Gender and Programming: What's Going On?* Proceedings of the 4th Annual SIGCSE/SIGCUE conference on Innovation and Technology in Computer Science Education. 1999.

[8] Charles M. Ray, Carolee Sormunen, Thomas M. Harris. *Men's and Women's Attitudes Toward Computer Technology: A Comparison*. Information Technology, Learning and Performance Journal. 1999.

[9] Leslie J. Francis. *The Relationship Between Computer Related Attitudes and Gender Stereotyping of Computer Use*. Computers & Education, vol. 22(4) 283-289. 1994.

[10] Curtis A. Carver, Richard Howard. *An Assesment of Networked Multimedia and Hypermedia*. 1995 IEEE/ASEE Frontiers in Education Conference. 1995.

Apéndice: Análisis estadístico

Nota: Se emplea un nivel de significancia del 95% (0,05)

Aproximación a la distribución normal de las puntuaciones en el pretest (Shapiro-Wilk)

Grupo	W	Sig.	Conclusión
control	0,938	0,559	0,938 ≅ 1 y 0,559 >> 0,05 ➔ La distribución puede considerarse normal
duck	0,898	0,365	0,898 ≅ 1 y 0,365 >> 0,05 ➔ La distribución puede considerarse normal
tradicional	0,947	0,672	0,947 ≅ 1 y 0,672 >> 0,05 ➔ La distribución puede considerarse normal

Homogeneidad de las varianzas entre las puntuaciones en el pretest (Levene)

Estadístico de Levene	gl1	gl2	Sig.	Conclusión
0,879	2	19	0,431	0,431 >> 0,05 ➔ Las varianzas son homogéneas

Equivalencia entre las puntuaciones en el pretest (ANOVA de un solo factor)

	Suma de cuadrados	Grados libertad	Media cuadrática	F	Sig.	Conclusión
intergrupos	17,537	2	8,769	2,645	0,097	0,097 > 0,05 ➔ Aptitudes equivalentes
intragrupos	62,977	19	3,315			
total	80,514	21				

Aproximación a la distribución normal de las puntuaciones en la escala de Likert para la actitud

Grupo	W	Sig.	Conclusión
control	0,972	0,899	0,972 ≅ 1 y 0,899 >> 0,05 ➔ La distribución puede considerarse normal
duck	0,967	0,857	0,967 ≅ 1 y 0,857 >> 0,05 ➔ La distribución puede considerarse normal
tradicional	0,921	0,471	0,921 ≅ 1 y 0,471 >> 0,05 ➔ La distribución puede considerarse normal

Homogeneidad de las varianzas entre las puntuaciones en la escala de Likert para la actitud

Estadístico de Levene	gl1	gl2	Sig.	Conclusión
0,391	2	19	0,682	0,682 >> 0,05 ➔ Las varianzas son homogéneas

Equivalencia entre las puntuaciones en la escala de Likert para la actitud

	Suma de cuadrados	Grados libertad	Media cuadrática	F	Sig.	Conclusión
intergrupos	0,305	2	0,153	0,014	0,986	0,986 > 0,05 ➔ Actitudes equivalentes
intragrupos	200,286	19	10,541			
total	200,591	21				

Aproximación a la distribución normal de las puntuaciones en la escala de Thurstone para trabajo personal

Grupo	W	Sig.	Conclusión
control	0,930	0,522	0,930 ≅ 1 y 0,522 >> 0,05 ➔ La distribución puede considerarse normal
duck	0,864	0,214	0,864 ≅ 1 y 0,214 >> 0,05 ➔ La distribución puede considerarse normal
tradicional	0,909	0,416	0,909 ≅ 1 y 0,416 >> 0,05 ➔ La distribución puede considerarse normal

Homogeneidad de las varianzas entre las puntuaciones en la escala de Thurstone para trabajo personal

Estadístico de Levene	gl1	gl2	Sig.	Conclusión
1,345	2	17	0,287	0,287 >> 0,05 ➔ Las varianzas son homogéneas

Equivalencia entre las puntuaciones en la escala de Thurstone para trabajo personal

	Suma de cuadrados	Grados libertad	Media cuadrática	F	Sig.	Conclusión
intergrupos	8,568	2	4,284	1,013	0,384	0,384 > 0,05 ➔ Trabajo equivalente
intragrupos	71,879	17	4,228			
total	80,448	19				

Aproximación a la distribución normal de las puntuaciones en el postest

Grupo	W	Sig.	Conclusión
control	0,912	0,441	0,912 ≅ 1 y 0,441 >> 0,05 ➔ La distribución puede considerarse normal
duck	0,908	0,410	0,908 ≅ 1 y 0,410 >> 0,05 ➔ La distribución puede considerarse normal
tradicional	0,985	0,978	0,985 ≅ 1 y 0,978 >> 0,05 ➔ La distribución puede considerarse normal

Homogeneidad de las varianzas entre las puntuaciones en el postest

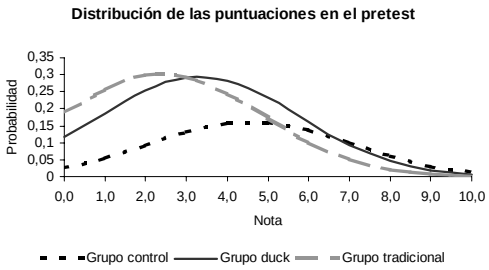
Estadístico de Levene	gl1	gl2	Sig.	Conclusión
0,341	2	17	0,716	0,716 >> 0,05 ➔ Las varianzas son homogéneas

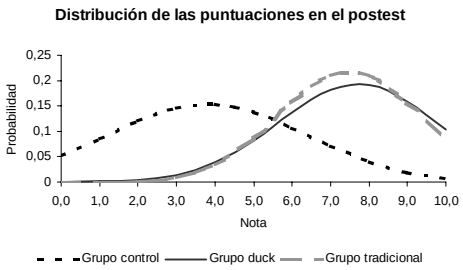
Mejoría en el postest frente al grupo control y de duck frente al tradicional (prueba T para muestras independientes)

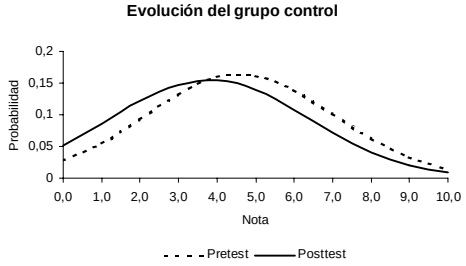
Grupos comparados	T	Grados libertad	Sig. unilateral	Conclusión
duck vs. control	3,027	11	0,0060	0,0060 << 0,05 ➔ Mejoría significativa
tradicional vs. control	2,965	11	0,0065	0,0065 << 0,05 ➔ Mejoría significativa
duck vs. tradicional	0,265	12	0,796	0,7960 >> 0,05 ➔ Mejoría **no** significativa

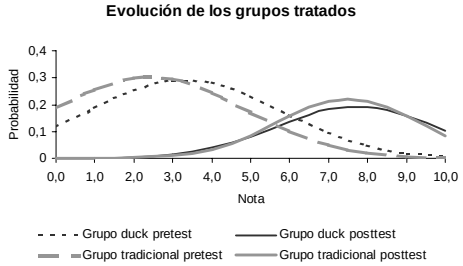
Cambios respecto a la situación inicial (prueba T para muestras emparejadas)

Grupos	T	Grados libertad	Sig. bilateral	Conclusión
duck vs. duck	11,664	6	0,000	0,000 << 0,05 ➔ Diferencia significativa
tradicional vs. tradicional	20,457	6	0,000	0,000 << 0,05 ➔ Diferencia significativa
control vs. control	-0,706	5	0,512	0,512 >> 0,05 ➔ Diferencia **no** significativa









Nuevas técnicas de aprendizaje: Cybergymkhana

Pedro José Lara Bercial, Juan José Escribano, David Atauri
Dep. de Programación e Ingeniería del Software
e-mail:{pedro,jje, atauri}@dpris.esi.uem.es
Universidad Europea CEES
28670 Villaviciosa de Odón

Resumen

En esta ponencia se presenta una nueva concepción en la utilización de Internet para la realización de ejercicios y el aprendizaje de técnicas de búsqueda en la red, basada en la posibilidad de realizar pequeños juegos de pistas utilizando las nuevas tecnologías.

1. Introducción

Es un hecho que las Nuevas Tecnologías de la Información y la Comunicación (NTIC) han cambiado tanto la forma en que se imparten las clases, como los hábitos de estudio del universitario del siglo XXI.

Según el acta del debate sobre nuevas tecnologías celebrado en Sevilla en el año 2000, en el marco de las Jornadas Andaluzas de Calidad[5], estas nuevas tecnologías se aplican para apoyar la docencia de la siguiente manera:

- Proporcionar información sobre aspectos relacionados con la gestión de la asignatura: programa del curso, fechas de exámenes, horarios de tutoría, etc.
- Ampliar y dar una nueva perspectiva a la comunicación entre el profesor y los estudiantes, así como de los estudiantes entre sí. Esta comunicación mediada por el ordenador incluye las modalidades sincrónica (mediante el chat) y asincrónica (a través del uso del correo electrónico; que se puede realizar de forma simple o con la implementación de listas de distribución. Otro apoyo consiste en la realización de tutorías electrónicas y foros electrónicos de las asignaturas.
- Como fórmula alternativa y complementaria a la docencia presencial, las NTIC son utilizadas para desarrollar estrategias de aprendizaje y actividades de formación a través de Internet. Este tipo de uso de las NTIC incluye la realización de materiales didácticos en formato hipermedial.

Centrando la discusión en este último punto, los esfuerzos por utilizar las NTIC como apoyo a la docencia presencial, van desde el aprendizaje en tiempo real, a la pura utilización de Internet como biblioteca universal. En la dirección primera van los esfuerzos del proyecto Numina [6], que se está llevando a cabo conjuntamente por la UNCW (University North Carolina at Wilmington), Pearson Education y Hypercube. Dicho proyecto se basa en la utilización redes inalámbricas en el aula que establecen entre el docente y los estudiantes un vehículo inmediato de transmisión de preguntas y respuestas. Se permite de este modo la exposición casi instantánea de los resultados de las respuestas realizadas, generando estadísticas y todo tipo de análisis y discusiones "al vuelo".

Otro ejemplo es nuestro propio proyecto Incampus [10][11] que comenzó a implantarse el año pasado y ha convertido a la Universidad Europea de Madrid en el primer campus inalámbrico de España, en el cual es posible acceder a la Intranet, Internet y a todos los recursos de la red —impresoras, discos duros, etc.— desde un ordenador portátil en cualquier lugar del dicho campus. Apoyándonos en esta infraestructura, pero más orientados hacia la búsqueda de información en Internet de manera rápida y efectiva, van los esfuerzos que han dado origen a la cybergymkhana.

Es indudable que la red se ha convertido en una de las herramientas de búsqueda de información y apoyo al aprendizaje, fundamentales para estudiantes de todo corte que aprenden a utilizarla en la Universidad y continúan haciendo uso de ella durante su carrera profesional. Si nos centramos además en los estudiantes de Ingeniería o carreras más técnicas, el uso de la red de redes durante sus estudios y posteriormente, una vez finalizados estos, se hace aún mucho más frecuente y esencial para ellos.

Por otra parte la red ha experimentado un crecimiento tan acelerado que se han pasado de los 2.100 millones de páginas en Internet [1] y más de 2 millones de sitios web [3] en el año 2000 a 2.1 billones de páginas [4] y más de 122 millones de dominios [11] registrados a principios del 2002. Este desmesurado aumento de la información distribuida en la red, convierte el proceso de búsqueda en una aventura que a veces acaba con la cancelación de la misma por la imposibilidad de encontrar, entre los cientos de resultados obtenidos, aquello que necesitamos. Los problemas inherentes a cualquier sistema de información -desorientación, navegabilidad ineficiente y sobrecarga de conocimientos recuperados- se multiplican en Internet hasta el punto de que los internautas novatos y los no tan novatos, se confunden y frustran con cómo buscar y con lo que encuentran [12].

Sin embargo y siendo realistas, tal y como dice Scout Brandt, profesor de la Universidad de Purdue, solo hay una cosa peor que intentar buscar algo en Internet, y es enseñar a otro a hacerlo[2]. A pesar de ello, las discusiones sobre cómo enseñar a buscar información, apuntan a la idea de que, en concreto en entornos hipertextuales, el aprendizaje debe ser experimental [13], asemejándose en cierta manera a la conocida memoria fotográfica. Es decir, un internauta experimentado, basa las construcciones de sus búsquedas y el filtrado de los resultados, en experiencias anteriores, utilizando lo que podríamos llamar patrones de búsqueda que empíricamente suelen dar buen resultado.

En un intento de enseñar a los alumnos de primero de la asignatura de Introducción a la Informática en Red (IIR) de las diferentes carreras de la Escuela Superior de Informática de la UEM, las mejores técnicas de búsqueda asociadas con la localización de contenidos en Internet y entrenarles en las mismas, surgió la idea de realizar un juego basado en la entrega sucesivas de pistas cuyas soluciones debían ser obtenidas a través de Internet. Para ello se desarrollaron los mecanismos hardware y software necesarios para que de forma global y totalmente automática todos los alumnos matriculados pudieran competir de manera remota y en cualquier momento, a cambio de pequeñas recompensas en la nota final de la asignatura.

2. Cybergymkhana

El juego de la gymkhana podría definirse como un juego de pruebas que han de ir realizándose una por una, y en el que la resolución de una prueba concreta depende directamente de la solución encontrada para la prueba anterior.

Las pruebas normalmente consisten en localizar un lugar y encontrar o conseguir algo en dicho sitio. Para ello el organizador reparte una o más pistas para cada prueba y los concursantes deben interpretarlas adecuadamente para conseguir hallar la solución que como ya se ha comentado es necesaria para seguir adelante con la gymkhana.

En el caso de la asignatura de IIR, la cybergymkhana cumple un doble objetivo. Por un lado, como se dijo anteriormente, entrenar a los alumnos en la búsqueda efectiva de información en Internet. Y por otro aprovechar el juego en si, para generar cybergymkhanas relacionadas con un tema concreto de los que componen el temario de la asignatura. Así, la propia búsqueda sirve también de apoyo al estudio.

Dado el carácter lúdico y no obligatorio de la participación en las diferentes cybergymkhanas, no es posible asegurar un número alto de alumnos participantes, por lo que, para animarles a jugar, se decidió este año 2001-2002, recompensar a los dos primeros que consiguieran terminar la cybergymkhana con dos “vidas” para la parte de test del examen final. En este examen la puntuación es la suma de todas las preguntas

acertadas menos las no acertadas, por lo que la posibilidad de fallar dos preguntas sin descontar puntos fue bastante bien aceptada por los alumnos

Funcionamiento

Dado el carácter automático del proceso de administración y resolución de la cybergymkhana, se crearon un sitio de Internet y una serie de páginas desde las que es posible realizar las siguientes operaciones:

Figura 1. Página de Inicio de la cybergymkhana

- Desde el punto de vista de la administración:
  - Configuración de la Cybergymkhana: Permite especificar el nombre de la tabla de la base de datos de cybergymkhanas que almacenará todo lo referente a la actual y algunas propiedades más de la misma (Figura 2).

Figura 2. Página de administración

  - Construcción de la Cybergymkhana: El usuario administrador puede añadir nuevas pistas a la prueba, especificando la solución y un posible mensaje de ayuda (Figura 3).

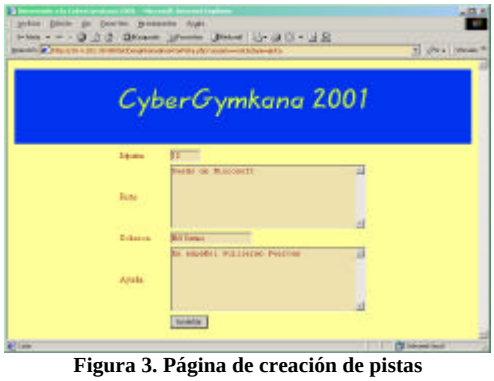


Figura 3. Página de creación de pistas

  - Monitorización del avance de los participantes: Permite saber en cualquier momento el estado de un participante indicando la pista que actualmente está intentando resolver, las ayuda que ha pedido, los errores cometidos y la fecha de la última pista resuelta (Figura 4).

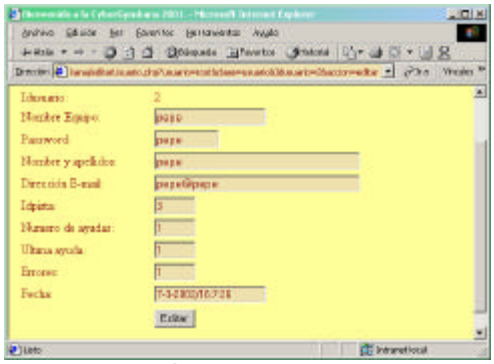


Figura 4. Página de alta de participante

- Desde le punto de vista de la participación
  - Alta de participantes: Cualquier persona que quiera participar, debe registrarse en la cybergymkhana conectándose a la página principal y dando entre otros datos su dirección de correo, dirección que se utilizará para enviarle su contraseña de acceso al juego, que le identifica como participante y le distingue del resto de jugadores.

- Lectura y resolución de la pista siguiente: Cada participante puede leer e intentar solucionar una pista concreta también a través de Internet (Figura 5). Si acierta, inmediatamente se le mostrará la pista siguiente, si falla se le contabilizará el error para más tarde evaluar su participación. Por cada error se le suman cinco horas al tiempo final dedicado a esa pista.

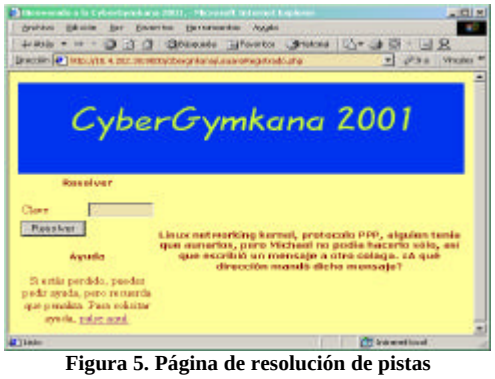


Figura 5. Página de resolución de pistas

- Petición de Ayuda: Si se es incapaz de encontrar la solución se puede pedir una ayuda por pista pero el participante será penalizado con tres horas que se sumarán al tiempo final dedicado a resolver la pista.

Ejemplo de Pistas

A modo de ejemplo en una de las cybergymkhanas de este año, cuyo tema central eran los manuales en línea que se pueden encontrar en Internet, aparecían las siguientes pistas:

Ejemplo 1:

Pista 2: *La compañía de diseño web de la pista uno, proporciona además servicios de consultoría y entrenamiento en diversas herramientas de diseño así como en un par de lenguajes de programación muy usados en diseño web. Uno de ellos, capaz de trabajar con "objetos" utiliza concretamente para trabajar con botones un objeto que tiene un método que quita el foco del ratón sobre dicho objeto. Este método se llama igual que un famoso grupo musical británico de los 90.*

Uno de sus componentes era Ingeniero de Sistemas Computacionales antes de pasar a formar parte del grupo. ¿Cuál es el primer apellido de dicho miembro?

(La compañía de la pista uno era Kodiac Graphics y, obviamente, no se decía en el texto de la pista dos)

Para resolver esta pista, había que utilizar al menos dos buscadores distintos, ya que en el mismo buscador en el que aparecía información del grupo musical, no aparecía la información acerca del método del objeto al que se refiere la pista. Y además, obligaba de alguna manera a ir familiarizándose con la siempre ardua labor de localizar información técnica en dichos manuales.

Ejemplo 2:

Pista 3: *De las dos tecnologías de programación nombradas en la pista 3, una fue creada por Rasmus Lerdorf en 1996. Sin embargo la versión 3 de dicho lenguaje, que introdujo un nuevo "parser" bastante mejorado, fue desarrollada fundamentalmente por otros dos ingenieros en un país de oriente medio. Uno de ellos ha participado recientemente en un congreso en Frankfurt.*

¿Cuál es el dominio del servidor de correo de la cuenta del administrador del web del congreso?

En esta ocasión la pista obliga, además de a bucear en la historia de un lenguaje de programación, a relacionar conceptos estudiados en la asignatura como por ejemplo: dominio, servidor de correo y cuenta de usuario.

Aspectos técnicos

Desde el punto de vista técnico, la arquitectura del sistema se corresponde con la de la figura 6. En ella puede observarse que existe una máquina servidora en la que está ejecutando un servidor

http, encargado de recibir las peticiones hechas por los usuarios e interpretarlas para devolver las correspondientes páginas web. Estas páginas son páginas activas capaces de obtener y modificar información de una base de datos que también está instalada y funcionando en dicha máquina servidora.

El usuario conectado a Internet desde su PC de sobremesa o desde su portátil, puede conectarse al servidor web, registrarse, y comenzar a participar en la última cybergymkhana que el administrador haya colgado utilizando el mismo mecanismo remoto.

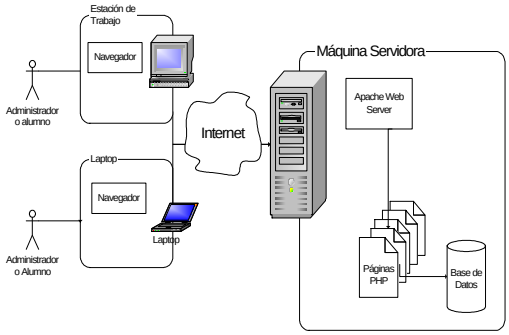


Figura 6. Arquitectura de la cybergymkhana

El servidor web instalado en la máquina servidora es un Apache [7]: un servidor de libre distribución construido en C y con código fuente libre. La base de datos es MySQL [8], uno de los sistemas de base de datos con código abierto más populares.

Las páginas encargadas de solicitar y mostrar información al usuario, tanto al administrador como al participante, están desarrolladas utilizando PHP [9], que es un conocido lenguaje de programación basado en guiones y orientado al desarrollo de aplicaciones Web. PHP puede ir embebido en páginas HTML y ser interpretado en el lado servidor.

3. Conclusiones

Desde el punto de vista docente, la experiencia llevada a cabo durante el primer cuatrimestre del curso 2001-2002 tuvo un gran éxito, con un alto número de participantes en la competición y con un gran número de preguntas de marcado carácter educativo, surgidas a raíz de los diferentes intentos de los alumnos por encontrar las soluciones a las pistas.

A modo de ejemplo, de las pistas comentadas anteriormente, surgieron cuestiones tales como qué es consultoría, qué es un objeto de programación, qué es un método de un objeto, qué es un *parser*, y quién es el *webmaster* entre otras. Preguntas todas ellas directa o indirectamente relacionadas con la asignatura de IIR.

El número de alumnos que completó la cybergymkhana antes del examen final fue de 6 de un total de 75, mientras que el total de alumnos que llegaron a resolver al menos una pista fue de 40. Aunque el sistema de penalizaciones hizo que el ganador o ganadores de la recompensa no fuesen rápidamente identificados, esto no evitó que un alto porcentaje de alumnos abandonasen al ver que dos o más de sus compañeros ya habían resuelto gran cantidad de pistas. En próximas cybergymkhanas se probaran nuevos modelos de recompensa que palien el abandono sin influenciar demasiado en la evaluación real que ha de hacerse de lo aprendido por el alumno.

Además, en futuras experiencias relacionadas con el juego, podría pensarse en la posibilidad de que los propios alumnos divididos en grupos diseñaran las pistas a resolver por sus compañeros. De esta manera, no solo aprenderían jugando sino preparando el juego, ya que como quedó patente este año, preparar la Cybergymkhana lleva un trabajo extra basado en hacer búsquedas en la red y en relacionar conceptos de la propia asignatura que perfectamente puede complementar la docencia de ésta.

Referencias

[1] B.H. Murray y A. Moore, *Sizing the Internet*, CyveillanceInc.,julio2000,(www.cyveillance.com)
[2] Brandt Scott, *Constructivism: Teaching for Understanding of the Internet*. Communications of the ACM. Octubre 1997.
[3] http://www.qwentes.com
[4] http://www.news-star.com

[5] http://www.ice.uma.es/ieev/biblos/jac.htm
[6] http://aa.uncwil.edu/numina
[7] http://www.apache.org
[8] http://www.mysql.com
[9] http://www.php.net
[10] http://www.cisco.com/global/ES/solutions/smb/webmarketing/wm_casestudy_home.shtml
[11] http://www.ganar.com/edicion/noticia/0,2458,9270,00.html
[12] Internet Software Consortium (http://www.isc.org)
[13] Oliver, R. *Interactive information systems: Information access and retrieval*. *Electronic Library 13*, 3 (June. 1995), 187–193.
[14] Spiro, R., Feltovich, P., Jacobson, M., and Coulson, R. *Cognitive flexibility, constructivism, and hypertext: Random*

Aplicación de las directivas EUROPA en la asignatura de Sistemas de Transmisión de Datos (Programas AME2 Y 3)

José Luís Poza Luján, Alberto Bonastre Pina, José Salvador Oliver Gil
Departamento de Informática de Sistemas y Computadores
Escuela Universitaria de Informática - Universidad Politécnica de Valencia
Camino de Vera S/N 46020 VALENCIA
e-mail: {jopolu, bonastre, joliver}@disca.upv.es

Resumen

En este artículo se describe la implantación de las directivas EUROPA en la asignatura de Sistemas de Transmisión de Datos (STD).

Estas directivas tienden a mejorar los sistemas de enseñanza-aprendizaje y evaluación. Las mejoras deben ir encaminadas a lograr los objetivos añadidos de preparación del alumno para el desarrollo de su actividad profesional.

Las acciones que se han puesto en marcha se dividen en dos tipos: actividades y seminarios. Por medio de las actividades se aprende a colaborar, cooperar y descubrir los problemas de coordinación que supone la cooperación. En los seminarios se fomenta el espíritu analítico, sintético y crítico que todo ingeniero debe poseer.

Describimos aquí pues, la filosofía de estos dos tipos de acciones que se van a realizar durante el segundo semestre del curso 2001-2002.

1. Marco de desarrollo de la asignatura

La asignatura de Sistemas de Transmisión de Datos (STD) se centra en los primeros niveles del modelo ISO/OSI (nivel físico y enlace de datos), profundizando en las técnicas de transmisión existentes y describiendo diferentes opciones de implementación.

Es una asignatura optativa, correspondiente a los plantes 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). Los objetivos planteados en la asignatura son:

- Conocer los sistemas de transmisión de Datos por medio de aspectos fundamentales (codificación, modulación, etc.) y de sus características Físicas.
- Aprender a comprender y realizar el diseño de sistemas de transmisión de datos.
- Programar los dispositivos físicos involucrados en la transmisión de datos.

Estos objetivos dan lugar a un desarrollo temático expuesto 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 la transmisión de datos de las aplicaciones reales que estos fundamentos tienen en la actualidad. Esta división proporciona dos visiones de los sistemas de transmisión de datos; una teórica donde se adquiere la base para comprender la segunda; y la segunda visión, la práctica y real que desarrolla la primera. De esta forma, la asignatura cubre tanto los objetivos de formación como los de preparación del alumno/a.

2. Marco de desarrollo del proyecto EUROPA

2.1. Precedentes

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. Las principales propuestas del PIE se resumen a continuación:

- Las enseñanzas deben enfocarse a la consecución del saber hacer del alumno.
- El sistema de evaluación debe responder a criterios como el control continuo del trabajo, el dominio de las técnicas, la capacidad de interrelación con los compañeros/as y la iniciativa del alumno.
- Es fundamental que el alumno de nuevo ingreso lo haga motivado y con una formación adecuada.

Durante estos doce años transcurridos se han aprobado numerosos PIE's, los cuales, en su mayoría, han producido un efecto positivo en la mejora docente.

Por otra parte, la acción del PIE y otras iniciativas han conducido a un incremento muy apreciable de la docencia en laboratorio (el 45% de los créditos impartidos en la UPV se dedican a ello) pero, por desgracia, ello se ha producido a costa de los créditos de prácticas en aula (sólo representan un 20% del total de créditos impartidos) manteniéndose muy elevados los de clases teóricas (35% de los créditos impartidos).

2.2. Proyecto EUROPA

Todo ello nos sitúa en unas coordenadas diferentes que exigen cambios en los objetivos y en la forma de abordarlos. Por ello se propone el Proyecto EUROPA [7].

El Proyecto EUROPA (Una Enseñanza ORientada al APrendizaje) se apoya en el Plan de Innovación Educativa (PIE) y toma de él parte de sus objetivos, en algunos casos los amplía y en otros los complementa con nuevas propuestas. Estos objetivos se resumen a continuación.

- Enfocar las enseñanzas a la consecución del saber hacer del alumno y lograr, además, que el alumno desarrolle al máximo su capacidad de autoaprendizaje.
- Incidir decididamente en la mejora de los sistemas de evaluación, favoreciendo la evaluación continua y la medida del saber hacer real del alumno.
- Poner la enseñanza al servicio de la sociedad, orientando la docencia hacia el empleo.
- Promover mejoras en la docencia que impliquen de un modo global a los Centros y Departamentos, por un lado, y de un modo personalizado e individual a cada profesor y a cada alumno.

La forma de abordar estos objetivos se concreta en cinco programas.

- Programa De Ayuda Complementaria A La Enseñanza (ACE).
- Programa De Ayuda A La Organización Docente (ADO).
- Programa De Ayuda A La Formación Integral Del Alumno (AFI).
- Programa De Ayuda A La Mejora En El Aprendizaje (AMA).
- Programa De Ayuda A La Mejora De La Enseñanza (AME).

En el presente curso se ha solicitado, y ha sido concedido, un proyecto AME, en concreto los subproyectos AME-2 sobre nuevos métodos de enseñanza-aprendizaje y AME-3 sobre mejora de los sistemas de evaluación. Dentro de esta concesión está la colaboración, como becarios, de dos alumnos que realizarán labores de apoyo a los profesores.

3. Descripción del proyecto de asignatura

Para la realización del proyecto EUROPA en la asignatura de Sistemas de Transmisión de Datos, se escogieron diversos objetivos que permitiesen alcanzar el objetivo. Este objetivo principal consiste en mejorar la asimilación, por parte del alumno, de los conceptos relacionados con la asignatura. La herramienta empleada es la introducción de nuevos métodos de enseñanza basados en el papel activo del alumno. Y esto se

hace por medio de actividades tutorizadas de trabajo en grupo y seminarios de discusión.

3.1. Metodología docente.

La metodología docente clásica, empleada habitualmente en el sistema de enseñanza-aprendizaje tradicional, divide las acciones formativas en teoría, dividida a su vez en clases magistrales y sesiones de problemas, y prácticas, asociadas, respectivamente, al aula y al laboratorio.

Para contrastar estas acciones con las que posteriormente se expondrán se explica a continuación cómo se han planteado en la planificación de la asignatura.

- *Sesiones de teoría*: Explicación de los fundamentos teóricos por parte del profesor, generalmente en el aula. Es lo que constituye la clásica sesión magistral. El alumno normalmente asimila estos conceptos por medio de la toma de apuntes.
- *Sesiones de problemas*: El profesor plantea ejemplos prácticos y los resuelve en pizarra. Estas sesiones también se suelen realizar en el aula donde el alumno resuelve y corrige los problemas.
- *Sesiones de prácticas en laboratorio*. Donde el profesor plantea ejercicios de desarrollo procedimental y donde el alumno adquiere las habilidades y destrezas complementarias de la aplicación práctica de los conceptos teóricos.

Para que los alumnos puedan desarrollar habilidades útiles en el mundo empresarial [3], se han planteado dos nuevas acciones formativas. Estas nuevas metodologías son las siguientes:

- *Seminarios*: Consisten en plantear y realizar alguna de las distintas técnicas de trabajo en grupo, e individual, que se desarrollan por parte de los alumnos.
- *Actividades*: Consisten en la realización obligatoria por parte de los alumnos de trabajos teórico-prácticos relativos a la asignatura. Dichos trabajos serán tutorizados por el profesor_en colaboración con uno o varios alumnos-tutores de actividades.

3.2. Metodología de evaluación.

Para la evaluación de los seminarios de discusión se valorará por parte del profesor la participación activa del alumno en los seminarios, el grado de comprensión y asimilación de la materia que se deduce de sus intervenciones y el grado de preparación de los materiales.

La evaluación de las actividades en grupo (trabajos) se realizará mediante evaluación personal por parte del profesor en base a los encuentros de tutorización (evaluación continua) y mediante encuesta a la clase, donde cada alumno puntuará sobre un máximo de 10 puntos varios aspectos del trabajo, como son el contenido, la claridad de la exposición o la aplicación práctica.

Dentro de la evaluación de la asignatura, también se debe tener en cuenta los conocimientos teóricos y prácticos adquiridos por los medios tradicionales. Esta ponderación se puede observar en la tabla 1.

Método	Área	Evaluación	Peso
Clásico	Teoría	Cuestiones	25%
	Problemas	Problemas	15%
	Prácticas	Cuestiones	20%
Innovador	Seminarios	Evaluación continua	20%
	Actividades	Presentación oral	20%

Tabla 1. Ponderación de las evaluaciones de las distintas metodologías.

Cabe destacar que a la parte innovadora de la asignatura se le ha dado un 40% de peso en la nota final, debido a que hasta el final del curso no se conocerá el impacto que esta parte pueda tener en el alumnado. Dentro de la parte innovadora se ha distribuido equitativamente (20%) el peso correspondiente a seminarios y actividades.

4. Descripción de la innovación docente.

4.1. Seminarios.

Los seminarios tratan de desarrollar capacidades y actitudes que no se cubren sólo con los métodos

tradicionales de teoría y prácticas o con la nueva metodología de las actividades.

Las actitudes principales que se intentan desarrollar son:

- Sentido y espíritu crítico.
- Capacidad de análisis y síntesis.
- Capacidad de discusión y argumentación.

Para lograr estas capacidades se ha planteado la realización de diversas acciones -individuales o en grupo-, en las que el alumno sea partícipe activo de las mismas. Por unificar la terminología, a estas acciones se les ha denominado seminarios.

Las acciones que se han propuesto, derivan de diversas metodologías de trabajo en grupo ampliamente empleadas. Entre ellas cabe destacar las expuestas a continuación, que son las que se realizarán en los seminarios.

- *Seminario*. Grupo reducido que estudia un tema intensivamente en varias sesiones en las que todos participan aportando sus indagaciones. Es a la vez una técnica de grupo y una técnica de conocimiento e investigación.
- *Mesa redonda*. Se parte de exposiciones sucesivas de alumnos que tienen diferentes puntos de vista acerca de un mismo tema. A continuación se mantiene una discusión en la que interviene un moderador.
- *Foro*. En el que el grupo en su totalidad realiza un debate abierto en torno a un tema, hecho o problema. La participación de cada uno se reduce a 2 ó 3 minutos.
- *Métodos de casos y proyectos*. Donde se estudia un caso real, se discute, se sacan conclusiones y se propone un proyecto de implantación o de abordaje del tema.

Los seminarios se realizan en el aula de teoría, lo cual es lógico puesto que son una ampliación innovadora de ésta y no precisan de un material complementario de laboratorio.

La planificación de los seminarios a lo largo del curso se expone en la tabla 2. En la citada tabla se observa cómo se ha dispuesto un primer seminario a la conclusión del bloque I.

Este primer seminario es un *estudio de caso* que se realiza para afianzar los conocimientos básicos de la asignatura. El estudio de casos ha sido ampliamente empleado en las enseñanzas técnicas como método activo de enseñanza-aprendizaje [4].

Sesión	Contenido
1	Presentación
2	Sistemas Teleinformáticos.
3	Medios de transmisión
4	Señal I. Conceptos.
5	Señal II. Aplicaciones.
6	Modems
7	Transmisión Serie
8	SEMINARIO
9	ADSL - Modem Cable.
10	Tarjetas de Red - Ethernet.
11	Transmisiones inalámbricas.
12	SEMINARIO
13	SEMINARIO
14	SEMINARIO

Tabla 2. Planificación de los seminarios.

Los posteriores seminarios están encaminados a realizar actividades de grupo más relacionadas con los conocimientos adquiridos en el segundo bloque, por lo que se plantean especialmente algunas actividades de las seleccionadas entre la amplia bibliografía existente al respecto [6].

Cabe destacar el intento de participación activa por parte del alumnado que se intenta dar con esta técnica. En posteriores estudios se irán añadiendo los resultados que deriven de aplicarla en la asignatura, así como la opinión de los alumnos sobre esta metodología.

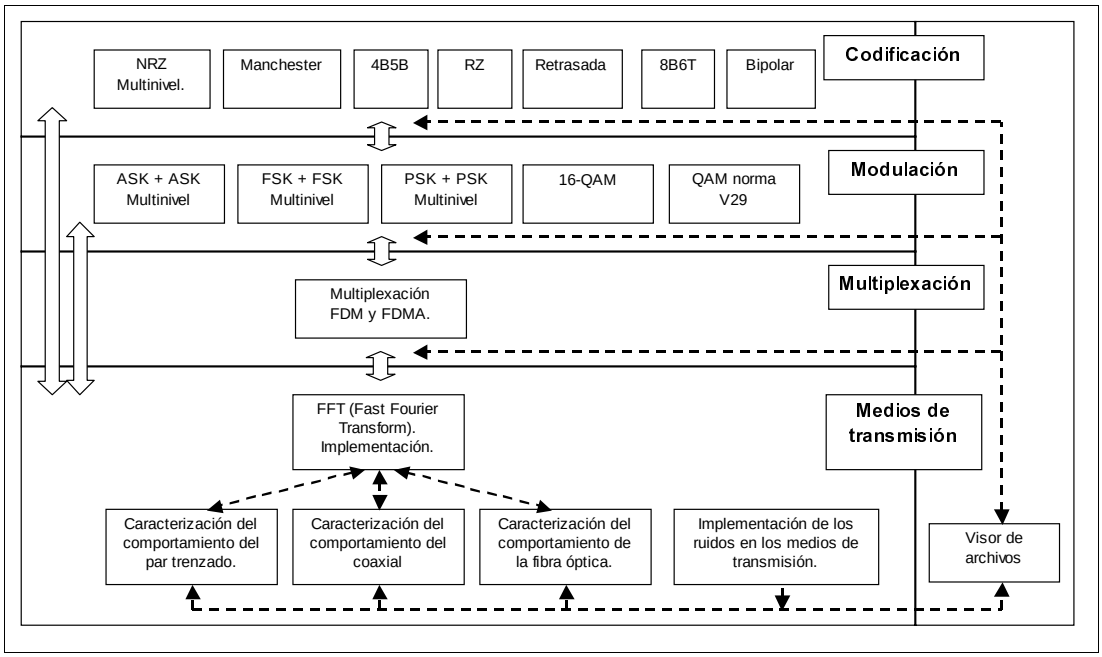


Figura 1. Conjunto de las actividades

4.2. Actividades.

Las actividades tratan de unificar la comprensión de los conocimientos teóricos expuestos en el aula con el desarrollo de los mismos en el laboratorio.

Esto en sí mismo no es ninguna innovación, pero en el presente curso se planteó la posibilidad de lograr algunos de los objetivos EUROPA por medio de un método docente innovador.

Las actividades consisten en la realización obligatoria por parte de grupos de alumnos de trabajos teórico/prácticos relativos a un área del temario del Bloque I de la asignatura. Dichos trabajos son tutorizados por el profesor, en colaboración con los alumnos tutores de actividades y seminarios.

Los contenidos organizados de las actividades se pueden ver en la figura 1. Estos contenidos se extraen del Bloque I de teoría, y están basados en el libro de referencia de la asignatura [1]. El componente práctico consistirá en la implementación de un código que, a partir de la entrada de un archivo de datos, los codifique, module o simule la transmisión y vuelque el resultado en otro archivo de datos. Este proceso se hará en ambos sentidos.

El funcionamiento de cada módulo deberá ser similar, teniendo una entrada y salida de archivos por medio de las cuales se comunican los datos con los otros módulos que estén a un nivel superior o inferior. Este modo de funcionamiento se ve en la figura 2.

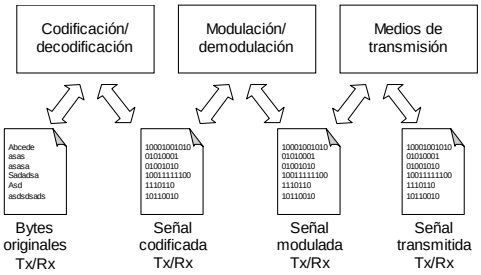


Figura 2. Interactuado de las actividades

Para la realización de las actividades, los alumnos se organizan en grupos que trabajan en

horario de prácticas. El funcionamiento de los grupos de actividades se expone a continuación.

1. Se forman los grupos de actividades, que deberán tener cinco componentes. En casos excepcionales justificados se podrá admitir grupos de menos de cinco componentes.
2. En la composición de los grupos debe de asumirse la responsabilidad de un área de trabajo. Las áreas que se han creado son las siguientes.
    a. Coordinación interna y externa
    b. Documentación.
    c. Implementación: análisis.
    d. Implementación: programación.
    e. Implementación: pruebas.
3. A lo largo del semestre, los grupos acuden al laboratorio para desarrollar el trabajo asignado. A su vez se realizan reuniones periódicas para resolver los problemas de compatibilidad de archivos, funciones y labores similares de coordinación. Se trata de que conozcan la realidad de la comunicación en un proyecto en el que están involucrados varios grupos de trabajo.

La planificación de las actividades a lo largo del curso se realiza de la manera expuesta en la tabla 3.

La primera de las sesiones de actividades está dirigida a mostrar el sistema de trabajo y a enseñar los esqueletos de las aplicaciones sobre el que los alumnos deben programar el módulo correspondiente. A lo largo del curso, los alumnos desarrollan las labores de documentación y programación para, en la última sesión, mostrar los resultados a los compañeros.

Sesión	Contenido
1	Medios de Transmisión.
2	ACTIVIDADES.
3	Modems
4	Señal Transmisión Serie.
5	ACTIVIDADES

Tabla 3. Planificación de las actividades.

Durante el curso, los profesores y los alumnos tutores son los encargados de revisar la evolución de las actividades, así como de controlar la coordinación entre los grupos. Esto último es necesario ya que este aspecto también se evalúa.

Se destaca_que las labores de programación son escasas ya que los algoritmos a implementar están ampliamente desarrollados, por lo que la labor del grupo debe centrarse principalmente en aportar su parte de código al conjunto global, así como a documentar todo el proceso y presentar los resultados públicamente.

4.3. Coordinación y comunicación en las actividades

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 y la coordinación 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 de comunicarse entre ellos, tanto a nivel vertical como a nivel horizontal [5].

- *Comunicación vertical.* En el proyecto de actividades expuesto anteriormente, los grupos escogen un área de la transmisión de datos en la que trabajar: codificación, modulación o medios de transmisión, Entre estas “capas” de las actividades deberá existir una comunicación para consensuar formato de los archivos, configuración de aplicaciones y paso de datos entre unas capas y otras. Además deberá haber una coordinación para las pruebas comunes de varias capas.
- *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 la faena común entre todos.

Con este sistema se pretende estudiar las interactuaciones 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. Esto último hace que uno de los componentes de los grupos deba asumir el papel de coordinador y comunicador. Se pretende con ello que los alumnos sepan concederle la importancia necesaria a la figura de coordinador de proyectos.

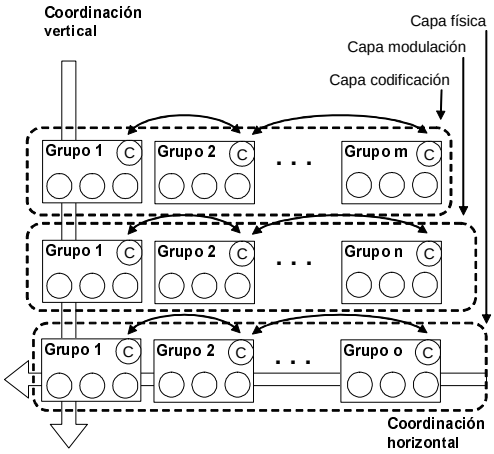


Figura 3. Relaciones de comunicación entre los diversos grupos de las actividades.

Otro de los objetivos consiste en comprobar si emerge de forma natural la figura del coordinador de capa horizontal y el coordinador vertical entre las diversas capas. Esta figura es necesaria y, en caso de no surgir, deberá ser adoptada por los alumnos-tutores bajo la coordinación de los profesores.

5. Conclusiones

En este artículo se ha propuesto una metodología docente en la que el alumno desarrolla unas habilidades por medio de seminarios y actividades innovadoras. Estas actividades o seminarios están dirigidos por los profesores y alumnos tutores que facilitan la coordinación y comunicación entre los grupos.

Metodología	**Bloque I**	**Bloque II**
Clásica	Teoría y práctica	
Innovadora	Actividades	Seminarios

Tabla 4. Aportación de la innovación en la estructura de la asignatura.

Al final del curso se realizará una encuesta a los alumnos que, junto a las calificaciones, será analizada para poder extraer conclusiones sobre la conveniencia o inconveniencia de esta metodología.

Cabe destacar la ilusión puesta por parte de los profesores de la asignatura y de los alumnos en emplear metodologías que favorezcan el desarrollo de habilidades y capacidades añadidas a las propias que se obtienen al cursar la asignatura.

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] James L. Gibson. *Las Organizaciones: Comportamiento, Estructura, Procesos*. McGraw-Hill. 1996.

[4] Leoncio Jimenez C., Pablo Kendall J. Angelica Urrutia S., Jorge Ibañez E. *Propuesta Metodológica para analizar casos como Modelo de Aprendizaje*. La Revista electrónica del DIICC. Edición Nº7. 2002.

[5] Musitu, G. *Psicología de la comunicación*. Nau Llibres. 1987.

[6] Newstrom, John W. Edward E. Scannell. *100 ejercicios para dinámica de grupos: una estrategia de aprendizaje y enseñanza*. McGraw-Hill, cop. 1989

[7] Vicerrectorado de coordinación académica y alumnado. *Una enseñanza orientada al aprendizaje: proyecto EUROPA*. UPV. 2001

Una herramienta de apoyo para la enseñanza de informática en estudios empresariales

Pedro L. Pérez Serrano, Luis Arévalo Rosado
Departamento de Informática
Universidad de Extremadura
10071 Cáceres
e-mail: { plperez,ljarevalo}@unex.es

Resumen

En este artículo se presenta una experiencia llevada a cabo en la Facultad de Ciencias Económicas y Empresariales de Badajoz. Esta se centra en la metodología utilizada para la realización, impartición y seguimiento de las prácticas de las asignaturas de informática que existen en dicho Centro, si bien esta metodología se puede poner en práctica en cualquier asignatura de informática de cualquier titulación. Con ese objetivo se han adaptado las características de un software de administración remota de libre distribución, comentando las ventajas e inconvenientes que hemos encontrado para el desarrollo de una clase interactiva. Por una parte desde el punto de vista de rendimiento de la computadora, y por otra parte, las ventajas e inconvenientes que tiene para los alumnos el seguimiento de la clase de prácticas utilizando esta metodología.

1. Presentación

La presencia de informática en las titulaciones de empresariales en esta Facultad [1] es, con relación a otras facultades de estudios empresariales, bastante relevante. Se puede decir que en todas las titulaciones que impartimos docencia, la informática dentro de los distintos planes de estudios empresariales, se encuentra siempre presente. En los nuevos planes renovados en 1998, se implantó como asignatura troncal en 3º de Diplomatura en Empresariales, obligatoria en 1º de Licenciatura en Economía y optativa en 2º y 3º de Diplomatura en Relaciones Laborales y Licenciatura en Administración y Dirección de Empresas. Además de estas asignaturas, el alumno tiene la posibilidad de matricularse en otra asignatura de libre elección, Informática Básica, en la cual se explican conceptos generales de informática, procesador de texto y diseño de página web, asignatura que todos los años el porcentaje de ocupación de la misma es del 100% por parte de los alumnos, quedando incluso lista de espera por posibles anulaciones de matrícula.

Con todo esto, unido a los continuos cambios en el campo de la informática, junto con la implicación directa que tiene, cada vez mas, en el mundo empresarial, nos obliga a esforzarnos y a actualizarnos para mejorar la calidad docente de tal forma que el alumno recién titulado tenga los conocimientos suficientes como para valerse por si mismo en la utilización de las herramientas mas comúnmente utilizadas en las empresas. No podemos consentir ya, en los tiempos que corren actualmente que un alumno no posea conocimientos básicos de manejo [2] de hojas de cálculo, bases de datos, internet entre otros.

2. Objetivos

Mas que un objetivo, lo que nos planteamos fue una nueva metodología de impartir las clases de prácticas, frente a posibles problemas tales como el alto número de alumnos con los que nos encontramos por grupo debido bien de espacio, disponibilidad de profesorado, número de créditos asignados a prácticas o por cuestiones de incompatibilidad de horarios, problemas muy comunes en muchas facultades. Con todo ello, junto con la precariedad que a veces nos encontramos en las salas de prácticas, en las cuales no siempre podemos encontrar un cañón de

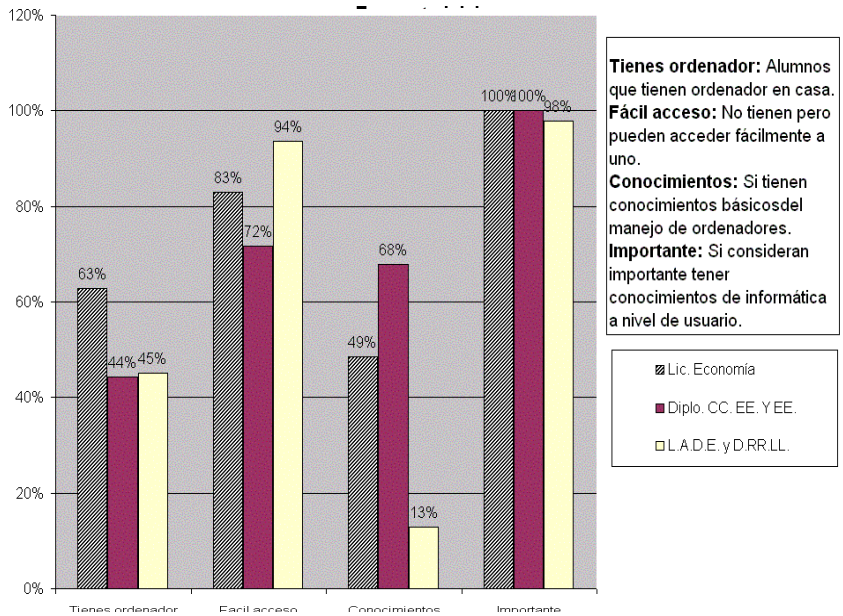


Figura 1. Encuesta realizada el primer día de clase

vídeo, imprescindible a nuestro parecer, fue lo que nos motivó a realizar esta idea y ponerla en práctica y ver los resultados.

Esta claro que el objetivo marcado por nosotros como profesores a la hora de dar una clase de prácticas es que sean lo más interactiva y eficiente posible, y que el alumno siga la clase al mismo ritmo que nuestras explicaciones. Esto, con el cañón de vídeo se puede conseguir, de tal forma que el alumno observa las explicaciones que se están realizando, proyectadas con el, al mismo tiempo que las aplica en su ordenador.

Pero, ¿cómo explicar conceptos, herramientas, ejercicios sin la disponibilidad (muchas veces) de un cañón de vídeo?, ¿no damos la clase?, ¿utilizamos la pizarra para las explicaciones o vamos ordenador por ordenador explicando a los alumnos?. Evidentemente la respuesta está clara, no. Hay que pensar que los alumnos, en nuestro caso en particular tienen muy pocas horas de prácticas (según la distribución de créditos de prácticas de la asignatura) y que en el poco tiempo que tienen deben de aprovecharlas lo mejor posible. Debemos hacer pues las prácticas lo más efectivas posible, sobre todo observando además que, aun siendo no obligatoria la asistencia a prácticas, el porcentaje de asistencia ronda el 100% (está visto que en la Universidad, el concepto de obligatoriedad a veces es más negativo que el de no obligatoriedad).

Además hay que tener en cuenta que existen muchos alumnos que cuando empieza el cuatrimestre poseen muy pocos conocimientos o ninguno acerca de informática, tal y como se refleja en la encuesta realizada el primer día de clase, figura 1. Por este motivo a estos alumnos es imposible enseñarle algo práctico sin que tengan una pequeña guía (bien sea cañón de video, transparencia, apuntes muy detallado) de ayuda. Por otra parte en determinadas prácticas debido al gran número de alumnos, las últimas filas no observan con suficiente claridad lo proyectado en el cañón de video.

Todos estos motivos nos obligaron a intentar desarrollar una nueva metodología que motivarán lo máximo a los alumnos y les distrajera lo mínimo. Ésta se muestra a continuación.

3. Metodología

3.1. Introducción

La idea consiste en aprovechar los recursos de los que dispone nuestra Facultad y las nuevas tecnologías para desarrollar una clase interactiva. Esta consiste en el desarrollo de un monitor virtual que muestra la información del monitor del profesor en cada uno de los equipos de los alumnos. Esta tecnología puede ser aplicada como elemento adicional al cañón de video. De esta manera los alumnos disponen de una ayuda con la cual visualizan de una forma más clara y concisa lo mostrado por el cañón o simplemente puede ser utilizado como un medio de exposición. El ordenador del profesor actúa como servidor y los equipos de los alumnos como clientes, donde el servidor se encarga de capturar la pantalla y enviarla a cada uno de los ordenadores de los alumnos. A la hora de desarrollar esta metodología dos posibles soluciones pueden ser tomadas:

- Solución mediante hardware
- Solución mediante software

La primera solución actualmente es utilizada en la Facultad de Ciencias del Deporte en la Universidad de Extremadura. Esta consiste en la existencia de un cuadro de control que recibe la señal del monitor del profesor y se encarga según la configuración del cuadro de mando de enviarla a cada uno de los alumnos seleccionados. Mediante este cuadro de control el profesor es capaz de canalizar la información a toda la sala, a un grupo determinado de alumnos o incluso a uno sólo. Además es capaz de invertir los papeles, es decir, que sea el monitor de un alumno el que pueda ser visualizado por el resto de los compañeros. Esta solución es idónea, sin embargo, el principal inconveniente que tiene es el coste. Por una parte se debe realizar la compra del cuadro de mando y los correspondientes interfaces de conexión con el monitor y por otra parte realizar una instalación de cableado que conecte correctamente todos los equipos de la sala.

La otra solución consiste en aprovechar los medios de transmisión que dispone cada uno de los ordenadores. La información procedente del monitor del profesor es recogida por una aplicación que se encarga de enviarla vía ethernet a cada uno de los equipos clientes que se encuentran conectado. Para desarrollar esta solución buscamos en la red aplicaciones de libre distribución con las características comentadas anteriormente. Existía una característica primordial que debe poseer la aplicación utilizada, la velocidad de transferencia entre el servidor y los clientes. Si esta velocidad no es lo suficientemente rápida puede producir que el refresco de la pantalla del alumno sea inadecuado, provocando una visión incomoda del monitor y que el alumno se puede distraer con mayor facilidad.

Por este motivo la selección del protocolo es muy importante. El lenguaje seleccionado para la comunicación entre los equipos fue IPX porque el servidor solamente tiene que enviar un único paquete de información, el cuál, se distribuye mediante difusión o “broadcast” a todos los equipos de la red. Esta solución es la más eficiente porque es independiente del número de equipos que se encuentren conectado en ese momento al servidor de pantalla. Los frutos de esta búsqueda no fueron los esperados y esta situación nos obligó a la búsqueda de aplicaciones basadas en el protocolo TCP/IP. La principal desventaja de esta solución es que el servidor tiene que enviar el mismo paquete de información a tantos equipos como clientes se encuentren conectados al servidor, siendo más lento la transferencia de la información.

Aun cambiando nuestras los objetivos de la búsqueda no se ha encontrado ninguna aplicación para el desarrollo de clases interactivas, motivo que nos obligo a la búsqueda de aplicaciones con distintas funcionalidades que se pudieran adaptar a los fines docentes marcados. En este caso se trata de una aplicación, RTVNC[3] cuya ventana principal se muestra en la figura 2. El principal

Figura 2. Programa RTVNC

objetivo de la aplicación es la administración remota de servidores. Ésta consiste en mostrar la información del servidor para que pueda ser administrado desde otro sitio físico. La idea es la misma para la enseñanza mediante ordenador, cada cliente se conecta al servidor y puede visualizar la pantalla de este. El funcionamiento es muy simple, a la hora de comienzo de la clase, cada uno de los alumnos ejecuta la aplicación cliente y el profesor queda residente el programa servidor en su ordenador.

3.2. Funcionalidades

Una vez conocida la aplicación a utilizar para el desarrollo de esta nueva metodología, debemos conocer cual son las características que necesitamos para la impartición de una clase interactiva. Estas funcionalidades se muestran a continuación:

- Se busca que el alumno pueda continuar trabajando con el resto de las aplicaciones aunque el monitor virtual se encuentre en funcionamiento. En aquellos casos en los que el alumno no entienda alguna explicación o necesita visualizar con mayor claridad el monitor del profesor para continuar con la práctica, puede recurrir a esta ventana de ayuda y posteriormente minimizarla.

- Otra funcionalidad sería la posibilidad de ver la ventana a pantalla completa. Esta puede ser utilizada cuando el profesor este desarrollando una clase teórico-práctica o una presentación. El alumno puede observar su monitor y no necesita esforzar la vista para ver con claridad el cañón de video para realizar el seguimiento de la explicación realizada por el profesor y sin la distracción de otras aplicaciones.

- Relacionada con la anterior, sería conveniente que se pudiera bloquear el ratón y el teclado de los equipos de los alumnos. Esta es necesaria en muchas ocasiones para que el alumno no se distraiga con los programas instalados en el ordenador.

- Por último la posibilidad de realizar una clase corporativa, es decir, el desarrollo de un ejercicio en el ordenador del profesor por parte de todos los alumnos. De esta forma el profesor puede motivar a los alumnos en la realización del ejercicio.

Todas estas funcionalidades son proporcionadas por la aplicación seleccionada. Esta aplicación además dispone de otras características que son mostradas a continuación:

- Posibilidad de seleccionar distintos algoritmos de compresión que se utiliza en el envió de la información. En determinadas situación la utilización de un algoritmo u otro puede proporcionar enviar mayor información a cada uno de los programas clientes.

- Posibilidad de seleccionar la captura de la pantalla con 8 bits. Esta característica es muy interesante para su uso en redes lentas cuyo tráfico de información puede estar reducido por el número de ordenadores o el ancho de la red.

3.3. Resultados de la metodología. Encuesta

Esta metodología ha sido utilizada este año en la Facultad de Ciencias Económicas y Empresariales. Este centro dispone de dos salas formadas por 25 equipos cada una. Las características son bastantes dispares. La primera de ella se encuentra formada Pentium I 133 con 32 MB y la otra por Pentium III 500 con 64 MB. Esta diferencia tan notable en los ordenadores, se comprueba en cuanto al nivel computacional de la aplicación. A continuación se muestran los resultados obtenidos comentando las ventajas e inconvenientes que hemos encontrado, por una parte desde el punto de vista de rendimiento de la computadora, como por otra parte, las ventajas e inconvenientes que tiene para los alumnos el seguimiento de la clase de prácticas utilizando esta metodología.

Los resultados que se han obtenido a nivel computacional han sido satisfactorios sabiendo que el desarrollo de esta metodología podía suceder que no funcionara debido al uso de un protocolo y una aplicación inadecuada para estos objetivos. El desarrollo de esta metodología ha sido probada en la segunda sala debido a la mayor capacidad computacional que disponen. El primer problema observado de la utilización de esta aplicación viene provocado por una sobrecarga computacional del equipo que actúa como servidor de pantalla. Esta situación viene provocada por el número tan elevado de información que tiene que enviar el programa servidor a cada uno de los clientes. A mayor

ENCUESTA INFORMATICA BASICA
Número de alumnos encuestados: 130

Interesante	Fácil de usar	Cañón vídeo	Util	Fácil seguimiento	Importantes
90	80	45	87	82	130
69%	62%	35%	67%	63%	100%

ENCUESTA INFORMATICA APLICADA
Número de alumnos encuestados: 310

Interesante	Fácil de usar	Cañón vídeo	Util	Fácil seguimiento	Importantes
234	280	89	256	230	300
75%	90%	29%	83%	74%	97%

Interesante: Si les parece interesante esta metodología.
Fácil de usar: Si les ha sido fácil de enterder y usar el programa.
Canón de vídeo: Si prefieren el cañón o esta nueva opción.
Util: Si les parece o no útil esta forma de dar las clases.
Fácil seguimiento: Si les parece fácil la realización de los ejercicios junto con la utilización de este programa.
Importantes: Si consideran importante las clases de practicas de informática.

Tabla1. Datos de la encuesta de la metodología

número de clientes la carga computacional del servidor es mayor y disminuye el rendimiento del mismo. Esta situación no ha provocado en ningún momento el mal desarrollo de esta metodología, si bien está claro que con un servidor más potente funcionaría mejor, ya que estamos hablando de Pentium a 500Mhz.

Otra característica que se ha observado, es según se incrementa el número de clientes el retardo de la información en los clientes aumenta. Se ha probado con 20 clientes y el seguimiento de la clase era permisible, con un cierto retardo, retardo que se puede disminuir con equipos más potentes. A un número reducido de clientes, la visión en los equipos de los alumnos se producía en tiempo casi real, mientras que a mayor número de clientes se producía un pequeño retardo. La resolución de la pantalla tiene que ser la misma para todos los equipos o inferior a la del servidor. Este no es tan importante como los anteriores pero debe ser tenido en cuenta. Además si utilizamos una resolución muy alta provoca que el retardo en los clientes sea mayor debido a que tiene que enviar mayor información. Si se produce un cambio cuando se esta utilizando la aplicación, provoca el cierre de la misma.

Los resultados de esta metodología en los alumnos fueron realizados mediante encuesta que se puede observar en la tabla 1. Esta ha sido realizada en dos asignaturas que se imparten en la Facultad de Económica. Los resultados gráficos de la encuesta puede visualizarse en la figura 3. Como se observa los alumnos de la asignatura informática básica fueron un poco más reacios al uso de esta tecnología. Esta situación puede producirse porque estos alumnos normalmente escogen esta asignatura en el primer curso de la carrera y no han trabajado normalmente con el ordenador. Los alumnos de aplicada recibieron esta aplicación con mayor curiosidad debido a que para el desarrollo de las prácticas necesitan observar el cañón con mayor claridad.

Por los datos obtenidos en la encuesta podemos concluir que el uso de esta metodología ha sido una experiencia positiva en el desarrollo de una clase interactiva. Se observó que la distracción por parte del alumnado ha disminuido por el uso de una herramienta que consiguió llamarles la atención. Los resultados de las notas

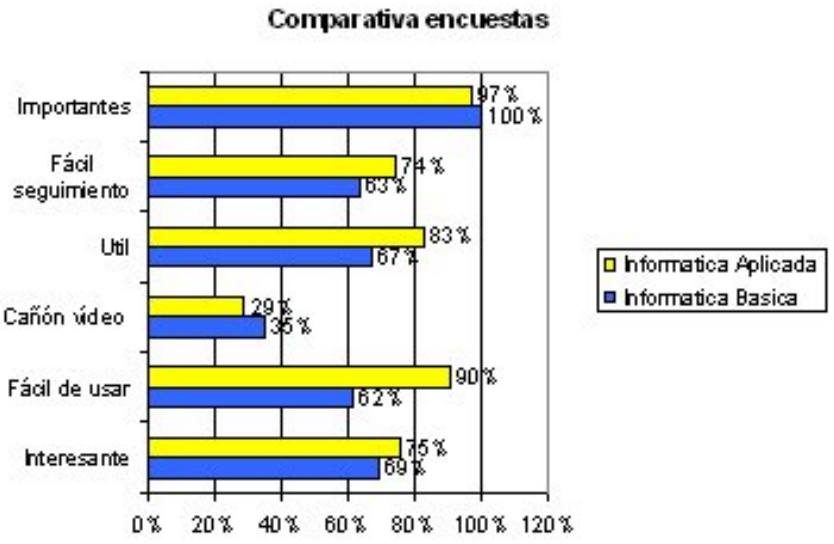


Figura 3. Resultado de la encuesta de la metodología.

de prácticas también han mejorado.

4. Conclusiones

Salvando los posibles inconvenientes técnicos comentados anteriormente, en este artículo hemos propuesto una forma de subsanar problemas con los cuales nos encontramos a veces en las salas de prácticas así como una posible metodología nueva para impartir clases prácticas de informática.

Con la posible utilización de esta técnica el alumno puede seguir perfectamente las explicaciones del profesor, sin necesidad de, por una parte, el profesor moverse de su ordenador, pudiendo resolver la duda monitorizando en una ventana del ordenador del alumno, el ordenador del profesor, y este explicar mediante la acción correspondiente en la aplicación u ejercicio que se esté realizando, la duda al alumno, el cual ve en la pantalla de su ordenador cómo se realizaría la acción, pasando rápidamente el alumno a realizarlo en su práctica, simplemente minimizando la ventana del profesor y maximizando la suya. Esta explicación podrá ser visualizada en todos los ordenadores de los alumnos que lo deseen. Por otra parte es una forma de que el alumno esté aprovechando al 100%, a nuestro modo de ver la clase, ya que podríamos hacer a la inversa, ver desde el ordenador del profesor, en todo momento, qué está realizando cada uno de los ordenadores de los alumnos.

Referencias

[1] Pérez Serrano, P. L. La asignatura de informática aplicada a la gestión de la empresa en Diplomatura en Ciencias Empresariales. VI JENUI 2000.

[2] Lapeña Marcos, María Jesús. La enseñanza de la informática en estudios empresariales. VII JENUI 2001.

[3]RTVNC. http://zone.syracuse.net/winme/adnlo ad / 18374 9_63804.html

Representación interna y aritmética de los números en computadores: Actividades para el laboratorio

Ester M. Garzón, Inmaculada García, José-Jesús Fernández

Dpto. Arquitectura de Computadores y Electrónica
Universidad de Almería. E-04120 Almería
e-mail: {ester,inma,jose}@ace.ual.es.

Resumen

En este trabajo se describe una estrategia para abordar el tema de la representación de los números y la aritmética en los computadores. Este tema está incluido en los planes de estudios de la mayoría de los títulos de ingeniero, especialmente en Ingeniería Informática. Esta iniciativa educativa se basa en un conjunto de ejercicios prácticos, planteados para subrayar las principales características de la representación interna de los números y su aritmética. Los ejercicios se desarrollan con un entorno computacional que constituye una valiosa herramienta para que los estudiantes reflexionen sobre este tema, abordando los aspectos más importantes, eliminando actividades tediosas. Esta propuesta puede incluirse en cursos introductorios relacionados con Estructura de Computadores, Programación, Matemática Discreta o Métodos Numéricos.

1 Introducción

La computación numérica es una herramienta esencial en todos los campos de la Ingeniería y la Ciencia [14]. Ingenieros y científicos hacen un uso intensivo de métodos numéricos y computadores para resolver problemas complejos. Para el desarrollo de la computación numérica, es de enorme importancia entender cómo se representan los números internamente en un computador, cómo se llevan a cabo las operaciones aritméticas y cuáles son sus limitaciones. Algunas de las causas más importantes de fallos de los sistemas están relacionadas directamente con errores de computación numérica [3,4]. Por ejemplo, algunos desastres de la vida real que se atribuyen a errores de de ese tipo son: el fallo del misil Patriot durante la Guerra del Golfo en 1991 [8], la explosión del Ariane 5 lanzado al espacio por la Agencia Espacial Europea en junio de 1996 [1] y otros.

Aunque la computación numérica se basa principalmente en la aritmética en punto flotante, la representación interna de enteros y su aritmética también tienen importancia [10], ya que las operaciones con enteros están incluidas en casi todas las aplicaciones de un computador. La representación interna de enteros más extendida entre los computadores modernos es el complemento a dos. Las principales características de este formato son los rangos de representación y los algoritmos para llevar a cabo las operaciones aritméticas.

Debido a la gran importancia de la computación en punto flotante en el contexto de la computación numérica [7], los científicos e ingenieros deben tener un profundo conocimiento de la aritmética en punto flotante y en particular del formato estándar IEEE 754. Desde el punto de vista de la computación numérica es importante conocer aspectos tales como rango, precisión y propiedades algebraicas relacionadas con cualquier representación finita en punto flotante.

Por otra parte, diversos aspectos del diseño de computadores también requieren un profundo conocimiento de la aritmética en punto

flotante: (1) desde el punto de vista de la arquitectura del computador, que aborda el diseño del repertorio de instrucciones, incluyendo aquellas instrucciones que involucran operaciones en punto flotante; (2) desde el punto de vista del compilador y diseño de lenguajes de programación, en el sentido de que la semántica del lenguaje debe ser suficientemente clara para efectuar una adecuada traducción de los programas a lenguaje máquina (3) desde el punto de vista del sistema operativo, en lo que se refiere al tratamiento de excepciones, en el sentido de que los manejadores de excepciones pueden ser definidos por el propio usuario de acuerdo con el problema que se esté tratando.

Las recomendaciones sobre los curricula de los estudios de Informática establecidas por The Joint IEEE Computer Society y ACM Task Force han considerado siempre el tema de la representación interna de los datos como fundamental en el área de Arquitectura de Computadores en la fase introductoria del curriculum; también han considerado este tema relacionado con Lenguajes de Programación y Ciencias de la Computación.

A pesar de su gran importancia y del hecho de ser un aspecto fundamental en el curriculum de Ingeniería Informática, la representación en punto flotante es poco conocida por los estudiantes y sólo es bien entendida por expertos. Ha habido diversas iniciativas en los años noventa [12,2,13] y otras más recientes [6,9] para hacer más asequible este tema a los estudiantes. Algunas de ellas [2,6,9] se centran directamente en el estándar IEEE de punto flotante, explicando la propia representación y los aspectos relacionados con ésta. Otras [12,13] hacen uso de una representación intermedia en punto flotante para ilustrar los principales conceptos. Los trabajos [12] y [9] también incluyen ejercicios para clarificar los conceptos de precisión y rango presentes en cualquier representación finita en punto flotante. Finalmente, la referencia [9] ofrece una serie de programas para mostrar claramente las propiedades y limitaciones del estándar IEEE, desde la perspectiva de la computación numérica.

En este trabajo, describimos nuestra propuesta para la enseñanza de la representación interna de los números y la aritmética en los computadores, tanto enteros como reales, con el objetivo de ser incluida en asignaturas de los primeros cursos de Ingeniería Informática. Con el fin de facilitar la ilustración de los principales conceptos relacionados con la representación interna de los números enteros y reales, esta propuesta hace uso de (1) un conjunto de ejercicios prácticos; (2) un entorno computacional constituido por una serie de programas auxiliares y (3) un formato en punto flotante reducido que sirve para introducir el formato IEEE.

2 Representación interna de enteros. Aritmética entera

La representación más extendida para los números enteros es el complemento a dos [10]. El rango de la representación de enteros en complemento a dos, con cadenas de p bits, es $[-2^{p-1}, 2^{p-1}-1]$. Con este código se tiene una única representación para el cero por lo que se facilita su detección. Por otro lado, esta representación es especialmente adecuada desde el punto de vista del hardware porque no requiere circuitos adicionales para la operación de la resta.

Con el objetivo de ilustrar los aspectos relacionados con esta representación, se ha diseñado un conjunto de ejercicios prácticos, basados en un entorno computacional que se presenta mas adelante. Estos ejercicios se pueden clasificar en dos grupos relacionados con la representación y con la aritmética.

2.1 Nuestra propuesta para enseñar representación de números enteros

Los ejercicios incluidos en este grupo proponen al alumno: (1) convertir de código decimal a complemento a dos, diversos enteros empleando distintos valores de longitud de palabra (p); (2) determinar el número mínimo de bits que

se necesita para representar un número dado; (3) determinar el rango de la representación en complemento a dos para diferentes longitudes de palabra.

Un ejemplo de ejercicio incluido en este grupo es:

(a) Determinar la representación en complemento a 2 de los enteros siguientes: $-15, 127, 128, -300$*, con una longitud de palabra* $p = 8$*.*

(b) Con la misma longitud de palabra, efectuar la conversión inversa aplicando como entrada los resultados del apartado anterior.

(c) En principio los resultados obtenidos en (b) deberían ser iguales a las entradas de (a). Analizar por qué en algunos casos eso no ha ocurrido.

2.2 Nuestra propuesta para enseñar aritmética entera

Los ejercicios que se clasifican en este grupo incluyen el desarrollo de las operaciones aritméticas usando diferentes longitudes de palabra. El software que se ha diseñado para estos ejercicios ilustran los algoritmos que se implementan en las unidades aritméticas de enteros (suma/resta, algoritmo de Booth, división con y sin restauración), ofreciendo el detalle del desarrollo de dichos algoritmos y teniendo en cuenta la situación excepcional overflow.

En el desarrollo de este tipo de ejercicios se sugiere a los alumnos que efectúen las diferentes operaciones aritméticas con diversos valores de longitud de palabra, haciendo uso del entorno computacional.

3 Representación y aritmética en punto flotante

El estándar IEEE ha sido ampliamente adoptado para la representación interna de los reales. Se utiliza prácticamente en todos los procesadores y coprocesadores numéricos de hoy en día para llevar a cabo los cálculos en punto flotante [2,5]. Esta sección comienza revisando los aspectos más destacables de este estándar y a continuación se centra en la descripción de nuestra propuesta para facilitar la enseñanza de estos conceptos.

3.1 El estándar IEEE 754 para la representación binaria en punto flotante

El estándar IEEE define un formato de simple precisión, de forma que el número real r expresado en notación exponencial como

$$r = (-1)^s \times (1.m) \times 2^{E-S}$$

se representa por una cadena de 32 bits que se organizan como sigue: un bit para el signo (s), $n_e = 8$ bits para el exponente sesgado $E = e + S$ (denotando e el exponente y $S = 2^{n_e-1} - 1$ el sesgo) y $n_m = 23$ bits para la mantisa. La base implícita que se considera en esta representación es 2. El formato de doble precisión dispone de 64 bits, de los cuales 11 están destinados al exponente, 52 a la mantisa y uno al signo.

Existe una gran variedad de conceptos relacionados con la representación en punto flotante que cualquier Ingeniero Informático debe conocer:

- El número de bits del exponente y mantisa definen el rango y la precisión de la representación.

- La distancia entre dos datos en punto flotante consecutivos varía a lo largo de la recta real. A medida que se consideran valores mayores, esa distancia es mayor y viceversa.

- Existe distancia relativamente significativa entre el cero y el menor real distinto de cero. No obstante, el uso de números denormalizados permite reducir esta distancia.

- Se define la precisión de la máquina, ϵ_{mach}, como la distancia entre 1.0 y el menor número mayor que 1.0 representable por el formato. Este parámetro se puede tomar como una medida de la precisión de las operaciones en punto flotante.

- El cero se representa por una cadena de bits de valor cero.

- Los errores de redondeo, truncamiento y cancelación están presentes en la aritmética en punto flotante.

- Se asocian códigos especiales (Infinito, Not-a-Number) a situaciones excepcionales.

- Las rutinas que tratan las excepciones están bajo el control de usuario/programador.

- La representación en punto flotante no cumple exactamente las reglas fundamentales del álgebra convencional. Propiedades como asociativa o distributiva no se verifican exactamente.

3.2 Nuestra propuesta para enseñar aritmética en punto flotante

Por sencillez, en el desarrollo de los ejercicios prácticos se va a utilizar una versión reducida del formato IEEE 754 de representación de números reales. En términos generales, nuestro formato sigue todas las convenciones del estándar IEEE 754, pero emplea sólo 12 bits, de los cuales $n_e = 5$ bits están destinados al exponente, $n_m = 6$ bits a la mantisa y uno al signo. El empleo de este formato de representación tan particular, nos va a permitir ver más fácilmente todos los conceptos y problemas de aproximación, redondeos y precisión asociados a la representación de los números reales. La Tabla 1 resume las principales características de este formato reducido.

El uso de esta versión reducida del formato estándar IEEE 754 trata de alcanzar los siguientes objetivos: (1) identificar fácilmente la relación entre el número real y su representación en punto flotante así como los efectos de redondeo en la conversión; (2) ilustrar claramente la relación de compromiso rango/precisión; (3) desarrollar los algoritmos de las operaciones aritméticas en punto flotante; (4) amplificar las anomalías propias de la computación en punto flotante, de forma que el uso de este formato reducido permite identificarlas más fácilmente; (5) permitir cálculos con lápiz y papel poco tediosos, si son necesarios. En resumen el formato que proponemos es bastante adecuado para ilustrar los aspectos

Tabla1. Reprentación en punto flotante reducida

Características más importantes de la representación

Longitud representación:	12 bits: signo: 1 bit exponente: n_e =5 bits mantisa: n_m =6 bits
Sesgo del exponente:	$2^{n_e-1} - 1 = 15 = 01111_2$
Rango del exponente:	$[-14, 15]$
Precisión de la máquina:	$\epsilon_{\text{mach}} = 2^{-n_m} = 2^{-6}$
redondeos empleados:	*Trunc., más cercano*
Números denormalizados:	Considerados
Excepciones consider.:	*Overflow*, *Underflow*, *Invalid*, *División por cero*

Valores especiales positivos más importantes

Valor especial	Representación	Valor
Mayor normal.:	0 11110 111111	$+1.984375 \times 2^{15}$
Menor normal.:	0 00001 000000	$+1.000000 \times 2^{-14}$
Mayor denormal.:	0 00000 111111	$+0.984375 \times 2^{-14}$
Menor denormal.:	0 00000 000001	$+0.015625 \times 2^{-14}$
+ cero	0 00000 000000	$+0.0$
+ Infinito:	0 11111 000000	---
NaN:	0 11111 111111	---

más relevantes de la representación en punto flotante.

Por otra parte, se ha diseñado un conjunto de ejercicios prácticos que tienen como objetivo ilustrar tales aspectos. Estos ejercicios se resuelven haciendo uso de un entorno computacional que se convierte en una valiosa herramienta, desde el punto de vista pedagógico. Este conjunto de ejercicios se puede clasificar en diferentes categorías:

1. Conversión de decimal a punto flotante, analizando los efectos de los distintos modos de redondeo.

2. El siguiente grupo de ejercicios, trata de facilitar el aprendizaje de conceptos tales como precisión, rango y compromiso rango/precisión. Incluye las actividades siguientes:

- Determinación de la representación de diferentes números reales usando diversos formatos, es decir variando el número de bits dedicados al exponente y mantisa.
- Determinación de la distancia entre dos números en punto flotante consecutivos en algunos intervalos, para diversos formatos de representación.

- Determinación de los rangos de representación de varios formatos en punto flotante, incluyendo los sub-rangos de números denormalizados y normalizados.
- Cálculo de los números mínimos de bits para exponente y mantisa sujetos a ciertas restricciones, por ejemplo, ser capaz de distinguir la representación de dos números muy próximos.

Como ejemplo, la Figura 1 resume el proceso para determinar el numero mínimo de bits en exponente y mantisa, para distinguir los reales 15.9 y 15.925 y a la vez representar el número 100000.

Determinación del número mínimo de bits en la mantisa para distinguir 15.9 y 15.925

	15.9	15.925
Formato Float	Representación	Representación
$n_e = 5$, $n_m = 6$	010010 111111	010010 111111
$n_e = 5$, $n_m = 7$	010010 1111110	010010 1111110
$n_e = 5$, $n_m = 8$	010010 11111100	010010 11111101
$n_e = 5$, $n_m = 8$	15.9→15.875	15.925→15.90625

Determinación de el número mínimo de bits para que además 100000. representable

	Número Real: 100000.	
Formato Float	Representación	Valor
$n_e = 5$, $n_m = 8$	0 11111 00000000	+Infinito
$n_e = 6$, $n_m = 8$	0 101111 10000110	99840.0

Figura 1. Proceso para determinar el formato en punto flotante más corto en el que los números 15.9 y 15.925 se distinguen y el real 100000.0 es representable. Solución obtenida: $n_e = 6$, $n_m = 8$

3. El siguiente grupo de ejercicios tiene como objetivo que los estudiantes analicen los procedimientos que se llevan a cabo para efectuar operaciones aritméticas con datos en punto flotante. Los ejercicios desarrollan sumas, restas, productos y cocientes con diferentes operandos, haciendo uso de distintos modos de redondeo y midiendo el error relativo de los resultados. Los operandos que se proponen se han escogido de forma que se presenten diversas situaciones. Por ejemplo, sumas de parejas de reales de magnitud muy distinta (en nuestro formato reducido $1000.0 + 2.5 \rightarrow 1000.0$).

Por otra parte, los programas incluidos en el entorno generan, como salida, una explicación detallada del proceso que se lleva a cabo para efectuar la operación aritmética en cuestión. De esta forma los estudiantes pueden distinguir fácilmente los distintas acciones asociadas a dichas operaciones.

4. Otra categoría de ejercicios analiza las distintas anomalías algebraicas que tienen su origen en la naturaleza finita de la representación en punto flotante. Nos parece importante que los alumnos reflexionen sobre la aritmética en punto flotante del computador y también sobre las limitaciones de ésta. Los ejercicios de este grupo incluyen:

- Análisis del error de cancelación que se produce en la sustracción de dos operandos de magnitud muy próxima.
- Ejercicios que muestran el efecto de acumulación de errores de redondeo. Por ejemplo, se multiplica un real por un entero mediante la suma sucesiva del primero, aplicando distintos modos de redondeo. Los estudiantes analizan la evolución del error durante el proceso de la suma. La Figura 2 muestra la salida que ofrece el entorno cuando se pide efectuar el producto 1.999×10.

Multiplicación 1.999×10

	Truncamiento		Redondeo más cercano	
Iter	Representación	Value	Representación	Valor
1 ->	0 01111 111111 ->	1.99900	0 01111 111111 ->	1.99900
2 ->	0 10000 111111 ->	3.96875	0 10000 111111 ->	3.96875
3 ->	0 10001 011111 ->	5.93750	0 10001 011111 ->	5.93750
4 ->	0 10001 111110 ->	7.87500	0 10001 111111 ->	7.93750
5 ->	0 10010 001110 ->	9.75000	0 10010 001111 ->	9.87500
6 ->	0 10010 011101 ->	11.62500	0 10010 011111 ->	11.87500
7 ->	0 10010 101100 ->	13.50000	0 10010 101111 ->	13.87500
8 ->	0 10010 111011 ->	15.37500	0 10010 111111 ->	15.87500
9 ->	0 10011 000101 ->	17.25000	0 10011 000111 ->	17.75000
10->	0 10011 001100 ->	19.00000	0 10011 001111 ->	19.75000

Figura 2. Producto 1.999×10 mediante sumas sucesivas, haciendo uso del formato en punto flotante reducido, con dos modos de redondeo.

- Ejercicios que demuestran que algunas reglas fundamentales del álgebra convencional no se verifican en computación en punto flotante: (1) adición y producto en punto flotante no son exactamente asociativas; (2) la propiedad distributiva del

producto respecto de la suma no se verifica exactamente; (3) la propiedad de cancelación no siempre es válida, es decir si los números positivos en punto flotante A, B, C verifican $A+B = A+C$, no siempre se verifica $B = C$; (4) el producto de un número en punto flotante por su inverso no siempre es igual a 1; (5) es casi siempre erróneo preguntar si dos números en punto flotante son iguales.

Un ejemplo muy interesante relacionado con las anomalías algebraicas en punto flotante está relacionado con la computación de la Serie Armónica:

$$H_N = 1 + 1/2 + 1/3 + \ldots + 1/N.$$

Hacemos uso de esta serie para mostrar que el orden distinto de sucesivas sumas produce resultado distinto, es decir, no se verifica la propiedad asociativa para la suma. Se propone calcular la serie en el orden que aparece en la definición de la serie y, por otra parte, calcular ésta en orden inverso y comparar los resultados. La Figura 3 muestra la salida que genera el entorno computacional, cuando se desarrollan estos ejercicios.

Por otra parte, está demostrado matemáticamente que esta serie diverge; no obstante, en aritmética en punto flotante no diverge debido a los errores de redondeo. Se han propuesto también ejercicios relativos a esta cuestión.

Computación de la Serie Armónica

$\sum_{n=1}^{10} \frac{1}{n}$ $\sum_{n=10}^{1} \frac{1}{n}$

Index	Number	Sum 1..N	Number	Sum N..1
1	1/1 = 1.00000	1.000000	1/10 = 0.10000	0.100000
2	1/2 = 0.50000	1.500000	1/9 = 0.11111	0.210938
3	1/3 = 0.33333	1.828125	1/8 = 0.12500	0.335938
4	1/4 = 0.25000	2.062500	1/7 = 0.14286	0.476562
5	1/5 = 0.20000	2.250000	1/6 = 0.16667	0.640625
6	1/6 = 0.16667	2.406250	1/5 = 0.20000	0.843750
7	1/7 = 0.14286	2.562500	1/4 = 0.25000	1.093750
8	1/8 = 0.12500	2.687500	1/3 = 0.33333	1.421875
9	1/9 = 0.11111	2.812500	1/2 = 0.50000	1.921875
10	1/10 = 0.10000	2.906250	1/1 = 1.00000	2.937500

Figura 3. Computación de la serie armónica con $N = 10$, haciendo uso del formato en punto flotante reducido y distintos ordenes de los sumandos

5. Finalmente, los ejercicios relacionados con las excepciones constituyen otro grupo de actividades. Puesto que el estándar IEEE 754 permite que las rutinas que tratan las excepciones estén bajo el control del usuario/programador, es extremadamente importante que el alumno esté familiarizado con las situaciones que producen excepciones, cómo tratarlas y qué valores en punto flotante generan excepciones. Los ejercicios que están dentro de esta categoría están relacionados con las siguientes cuestiones:

- Algunas operaciones aritméticas con datos numéricos en punto flotante pueden generar excepciones, debido a que el resultado sea no representable. Por ejemplo, el producto de dos números de muy pequeña magnitud puede generar una excepción de tipo *Underflow*, o la suma de dos números de gran magnitud puede generar una excepción de tipo *Overflow*, etc. Dos ejemplos concretos en el formato en punto flotante reducido producen excepciones: $0.0009 * 0.001$ genera *Underflow* y $1875 * 35$ genera *Overflow*.
- Dado un número en punto flotante A, determinar el resultado de las operaciones en punto flotante $A + \infty$, $A * \infty$, $A/0$, A/∞.
- determinar el resultado de las operaciones en punto flotante $\infty + 0$, $\infty + \infty$, $\infty - \infty$, $\infty * 0$, $\infty * \infty$, $\infty * (-\infty)$, $0/0$, ∞/∞, $\infty/0$.

La Figura 4 muestra diferentes situaciones excepcionales, las operaciones con las que se asocian y el valor en punto flotante que generan. Esta correspondencia debe verificarse por todo formato en punto flotante que se rija por el estándar IEEE.

Cabe destacar, para concluir esta sección, que los ejercicios prácticos descritos se desarrollan con un soporte computacional que se describe en la siguiente sección. Los programas que forman parte de este entorno están diseñados para generar las respuestas y justificaciones de los ejercicios que acaban de ser descritos.

Singular operation	Exception	Returned value
$A - \infty$	Overflow	-Infinito
$\infty + \infty$	Overflow	+Infinito
$\infty - \infty$	Invalid	NaN
$A * \infty$	Overflow	±Infinito
$\infty * 0$	Invalid	NaN
$\infty * (-\infty)$	Overflow	-Infinito
$A/0$	Division-by-zero	±Infinito
A/∞	Underflow	±Cero
$0/0$	Invalid	NaN
∞/∞	Invalid	NaN
$-\infty/0$	Overflow	-Infinito

Figura 4. Diversas situaciones que producen excepciones en nuestro formato. El símbolo ± significa que el signo del resultado depende del signo del operando.

4 El entorno computacional auxiliar

Hemos desarrollado un entorno computacional auxiliar para que los estudiantes desarrollen actividades prácticas sobre la representación y la aritmética de los computadores, facilitándoles la comprensión de los principales aspectos relacionados con este tema. El entorno está constituido por un conjunto de programas escritos en lenguaje de programación C estándar. Con el objetivo de construir una herramienta independiente de la plataforma, en la medida de lo posible, los programas internamente aplican técnicas relacionadas con aritmética de múltiple precisión, para generar las representaciones y las operaciones aritméticas en punto flotante. Hemos comprobado los programas en diferentes plataformas (PC, SGI, Sun, Alpha) y diferentes sistemas operativos (Windows, Linux, IRIX, Solaris) obteniendo resultados coherentes. Los programas pueden ser ejecutados en consola, especificando en la línea de comandos los operandos de entrada y las opciones. Para los programas relacionados con la representación y aritmética en punto flotante, las opciones incluyen la posibilidad de especificar el número de bits para el exponente y para la mantisa; la opción por defecto está asociada con el formato reducido que hemos descrito en la sección anterior ($n_e = 5$, $n_m = 6$).

También hemos desarrollado un entorno más amigable y basado en web para los programas. Con este entorno los alumnos no necesitan trabajar directamente en una ventana consola y la línea de comandos, sino que pueden acceder al entorno a través de su navegador y, de forma transparente, ejecutar los programas en el servidor web.

El entorno web está basado en formularios HTML (HyperText Markup Language) y tecnología CGI (Common Gateway Interface) [11]. Los estudiantes hacen uso de su navegador para: (1) seleccionar el programa que debe ejecutarse, (2) especificar los operandos y las opciones y (3) enviar los datos al servidor. La tecnología CGI se encarga de obtener los datos enviados por el cliente web, ejecutar el programa en el servidor con los parámetros de entrada especificados y, finalmente, enviar los resultados al cliente.

Nuestro entorno, que está disponible en la URL `http://www.ace.ual.es/RAC/`, está estructurado en dos bloques principales. Por un lado, todos los programas relacionados con la representación y aritmética de los enteros y por otro, los relativos a representación y aritmética en punto flotante. Además, existe un manual explicativo que contiene un resumen de los conceptos teóricos y los enunciados de todos los ejercicios, que están relacionados con el amplio espectro de conceptos asociados a la representación y aritmética de los números en un computador, como se ha descrito en este artículo.

Desde el punto de vista pedagógico, este entorno constituye una herramienta auxiliar ideal para los estudiantes. El entorno y el conjunto de actividades diseñadas facilitan enormemente, y desde una perspectiva eminentemente práctica, la compresión de todos los conceptos relacionados con la representación de números de longitud finita y su aritmética en computadores.

5 Conclusiones

En este artículo hemos descrito una estrategia educativa para desarrollar el tema de representación y aritmética de los números en un computador, tanto enteros como reales. Con el

objetivo de preparar a los estudiantes de ingenierías informáticas, para que puedan abordar problemas computacionales de la vida real, se desarrollan un conjunto de actividades relacionadas con los conceptos de computación numérica. Nuestra estrategia también está dirigida a ofrecer a los estudiantes conceptos elementales que forman parte de cursos sobre arquitectura de computadores, lenguajes de programación y métodos numéricos.

Nuestra propuesta consiste, principalmente, en un conjunto de ejercicios prácticos cuidadosamente diseñados para destacar los aspectos más importantes relacionados con la representación finita de los números. Además, ofrecemos un entorno computacional para que los estudiantes desarrollen dichos ejercicios. Después de trabajar siguiendo esta estrategia educativa durante varios cursos, en el desarrollo de la asignatura Estructura de Computadores de Ingeniería Técnica Informática de Sistemas, podemos decir que nuestra iniciativa facilita el aprendizaje de dicho tema.

Referencias

[1] Ariane 501 Inquiry Board. "Ariane 5. Flight 501 failure." *Inquiry Board Report, 1996*, [Online] Available: http://www.esa.int/export/esaCP/ /Pr_33_1996_p_EN.html.

[2] D. Goldberg. What every computer scientist should know about floating-point arithmetic. *ACM Computing Surveys*, 23(1):5–48, 1991.

[3] L. Hatton and A. Roberts. How accurate is scientific software? *IEEE Trans. Software Eng.*, 20(10):785–797, 1994.

[4] N.J. Higham. *Accuracy and Stability of Numerical Algorithms*. SIAM, 1996.

[5] W. Kahan. IEEE standard 754 for binary floating-point arithmetic. World-Wide Web document, 1996. [Online] Available: http://www.cs.berkeley.edu/∼wkahan/ /ieee754status/ieee754.ps

[6] W. Kahan. Ruminations on the design of floating-point arithmetic. World-Wide Web document, 2000. [Online] Available: http://www.cs.nyu.edu/cs/faculty/ /overton/book/docs/

[7] D.E. Knuth. *The art of computer programming, third edition.*, volume 2, Seminumerical Algorithms. Addison-Wesley, 1998.

[8] General Accounting Office (Information Management and Technology Division). "Patriot Missile Software Problem," General Accounting Office Report, 1992 [Online] Available: http://www.fas.org/spp/starwars/gao/ /im92026.htm

[9] M.L. Overton. *Numerical Computing and the IEEE Floating Point Standard*. SIAM, 2001.

[10] D.A. Patterson and J.L. Hennessy. *Computer Organization and Design. The Hardware/Software Interface*. Morgan Kaufmann Pub., 1998.

[11] G. Birznieks S. Guelich, S. Gundavaram. *CGI Programming on the World Wide Web*. O'Reilly, 2000.

[12] T.J. Scott. Mathematics and computer science at odds over real numbers. *ACM SIGCSE Bulletin (ACM Special Interest Group on Computer Science Education)*, 23(1):130–139, 1991.

[13] C.W. Steidley. Floating point arithmetic basic exercises in mathematical reasoning for computer science majors. *Computers in Education Journal*, 2(4):1–6, 1992.

[14] C.W. Ueberhuber. *Numerical Computation. Methods, Software, and Analysis*. Springer-Verlag, 1997.

Un enfoque algorítmico para enseñar "Visión por Computador" en las titulaciones de Informática

A. B. Moreno, A. Sánchez, J. Vélez
Dept. de Ciencias Experimentales e Ingeniería
Universidad Rey Juan Carlos
28933 Móstoles (Madrid), Spain
e-mail: a.b.moreno@escet.urjc.es, an.sanchez@escet.urjc.es, j.velez@escet.urjc.es

Resumen

Este artículo describe el enfoque y desarrollo de un curso sobre "Visión por Computador", impartido en el tercer curso de la titulación Ingeniería Técnica en Informática de Sistemas de la Universidad Rey Juan Carlos de Madrid. El artículo refleja nuestro punto de vista y experiencia sobre cómo enseñar esta materia interdisciplinar a los alumnos de titulaciones informáticas, con vistas a una mejor comprensión de las operaciones habituales sobre imágenes así como de los algoritmos más apropiados para implementarlas. Las aplicaciones de "Tratamiento de Imágenes" y de "Visión por Computador" han adquirido una gran importancia por sus múltiples usos en entornos industriales. Al mismo tiempo, ofrecen a los estudiantes la posibilidad de utilizar sus conocimientos sobre algoritmia para abordarlas. Mostrar la relación entre lo aprendido en cursos previos sobre "Algoritmos y Estructuras de Datos" con los diversos problemas de "Visión por Computador" estudiados en esta asignatura, contribuye a que el curso resulte más interesante para los alumnos.

1. Introducción

En los cursos introductorios sobre "Tratamiento Digital de Imágenes" y "Visión por Computador" se estudian los procesos sistemáticos que, aplicados sobre una imagen digital producen otra imagen modificada más apropiada para cierto propósito (por ejemplo, para ser visualizada, analizada, comprimida, etc.). Se trata de un área con múltiples aplicaciones en diversas disciplinas tanto científicas (biología, astronomía, meteorología, medicina...) como industriales (robótica, procesamiento de documentos, arte, biometría...)[5]. Como en otras áreas científicas, los problemas de "Visión por Computador" se pueden resolver de una manera metódica y estructurada. El análisis y el diseño eficiente de los algoritmos usados en las aplicaciones que trabajan sobre imágenes son componentes críticos en toda metodología de procesamiento de imágenes. Dado el elevado número de aplicaciones que manipulan imágenes, resulta apropiado para un estudiante de Ingeniería Informática, conseguir conocimientos prácticos sobre las operaciones más importantes de tratamiento de imágenes y sobre la manera eficiente de implementar dichas operaciones. Estos conocimientos se pueden utilizar y ampliar en materias afines, por ejemplo, en cursos más avanzados de "Visión Artificial" o de "Robótica".

El enfoque propuesto en este trabajo se ha aplicado a la asignatura "Visión Computacional" de Ingeniería Técnica en Informática de Sistemas de la Universidad Rey Juan Carlos. Esta asignatura es optativa, y se imparte en el tercer curso de la titulación durante un cuatrimestre (14 semanas) con tres horas lectivas por semana. El curso está estructurado en clases teóricas y prácticas. La teoría debe ser aprobada mediante un examen escrito. La parte práctica está organizada en dos partes principales: (1) una práctica de tratamiento de

imágenes que cada alumno realiza de forma individual, y (2) un proyecto consistente en una aplicación de "Visión por Computador" para ser realizado por grupos de tres alumnos. Para las prácticas se requiere el conocimiento de un lenguaje de programación estructurada de alto nivel (preferentemente Pascal o C). Dicho conocimiento se consigue cursando otras asignaturas troncales como "Metodología de la Programación" y "Estructuras de Datos y de la Información", impartidas respectivamente en los cursos primero y segundo de la carrera. Para la realización de los proyectos propuestos, es posible usar las librerías de software disponibles para el curso: MATLAB [9] y la biblioteca de funciones MIL de la tarjeta digitalizadora Meteor 2/4 Matrox [10] para prototipado y desarrollo rápido de programas.

A continuación se describe la estructura del resto del artículo. La sección 2 resume nuestras ideas generales sobre la enseñanza de la "Visión Computacional" en titulaciones informáticas. La sección 3 describe los elementos de teoría del curso. En la sección 4 se presentan varios ejemplos de prácticas propuestas en el curso. En la sección 5 se describen algunas propuestas de proyectos en grupo sobre aplicaciones reales de visión por computador, donde se pone énfasis en los requerimientos de corrección y eficiencia de las soluciones programadas. Finalmente, la sección 6 resume las conclusiones extraídas de nuestra experiencia.

2. Reflexiones sobre la enseñanza de "Visión por Computador" en titulaciones informáticas.

Como hemos indicado, la "Visión por Computador" es una *materia interdisciplinar*. Está relacionada con la informática, las matemáticas y la ingeniería, y también con otras áreas donde se puede aplicar (por ejemplo, la medicina o la geología). Este carácter interdisciplinar convierte a la asignatura en atractiva y de gran utilidad profesional pero, por otro lado, hace difícil el diseño de cursos basados en tratamiento de imágenes y visión computacional. En consecuencia, el profesorado debe optar inevitablemente por diferentes enfoques, por lo que los simples objetivos de la asignatura pueden variar muchísimo. En nuestro caso, al encontrarnos en una titulación de informática y tratarse además de un primer curso en el área de la "Visión por Computador", nuestro planteamiento consiste principalmente en *introducir los problemas y técnicas del tratamiento digital de imágenes y visión artificial desde una perspectiva algorítmica*. Otras consideraciones a tener en cuenta al planificar los contenidos y la orientación del curso son las siguientes:

- *La visión computacional es una materia "en auge".*
  Actualmente muchas empresas tienen una opinión bastante positiva sobre el futuro de la tecnología del tratamiento de imágenes. Su importancia está no solamente en el reconocimiento de objetos y en la interpretación de imágenes, sino en otros campos como: la realidad virtual, Internet y las bases de datos de imágenes.

- *Los estudiantes necesitan algo más que teoría.*
  Aunque existe una parte teórica de la visión computacional que está bien definida, se necesita imperativamente un trabajo de laboratorio y de proyectos de visión con una orientación sobre los sistemas. Hacen falta prácticas cortas para familiarizar a los alumnos con cámaras, tarjetas digitalizadoras, etc. Se necesita dedicar tiempo a explicar a los alumnos proyectos reales que se puedan resolver (así como su solución) usando técnicas de "Visión por Computador". En las soluciones es importante hacer énfasis en los aspectos de eficiencia y tiempo de respuesta del sistema implementado.

- *¿Qué enseñar?*
  Este curso suele ofertarse como materia optativa en los últimos años de titulaciones informáticas (ingenierías técnicas, ingeniería superiores o licenciaturas). La manera tradicional de ofrecer la materia es mediante clases magistrales de teoría y la realización de prácticas de laboratorio. En nuestra propuesta, además de considerarse el enfoque tradicional, se aporta como novedad el uso de técnicas de diseño de algoritmos y de otros elementos

algorítmicos para analizar y resolver problemas relacionados con los contenidos de la asignatura.

- ¿Cómo *enseñar?*
  Al impartir cursos relacionados con esta materia hay que usar presentaciones interactivas que permitan la visualización del problema y ofrecer prácticas que tuviesen relación con problemas reales de visión artificial.

3. Elementos de teoría del curso

Como se ha señalado, este curso está orientado a estudiantes de Ingeniería Técnica en Informática, durante un cuatrimestre y tiene asignadas tres horas lectivas semanales (de las cuales se destina aproximadamente el 50% del tiempo a teoría y a prácticas). Aunque nuestro enfoque es bastante práctico, también consideramos necesario explicar los fundamentos teóricos básicos del tratamiento de imágenes digitales y la visión por computador. Durante el curso se utilizan varios textos, aunque no encontramos ninguno "ideal" para el curso completo. El libro más utilizado ha sido González y Woods [4]. Otros libros de texto complementarios son Jain et al. [6], Baxes [1], y Maravall [8]. A continuación, se muestra un esbozo de los temas cubiertos en las clases de teoría junto con el número de semanas dedicadas a cada tema (entre paréntesis):

1. Introducción: problemas y aplicaciones (1)
2. Adquisición de imágenes digitales (1)
3. Muestreo y cuantización de imágenes (1)
4. Realce y restauración de imágenes (3)
   · operaciones de tratamiento a nivel de píxeles y de regiones,
   · transformaciones geométricas sobre imágenes,
   · tratamiento en el dominio de la frecuencia.
5. Análisis de imágenes (3):
   · segmentación,
   · extracción de características,
   · reconocimiento de formas.
6. Compresión de imágenes (2):
   · métodos de compresión con y sin pérdida.
7. Aspectos avanzados (3):
   · morfología matemática,
   · introducción a la visión tridimensional,
   · visualización de datos.

Al explicar el temario del curso, nos podemos beneficiar del conocimiento recibido por los alumnos en cursos previos como los de "Metodología de la Programación" y "Estructuras de Datos y de la Información". Por ejemplo, las técnicas de diseño de algoritmos se podrían relacionar con ejemplos de algoritmos representativos explicados en nuestro curso de "Visión Computacional". Pensamos que esta integración, cuando se pueda llevar a la práctica, resulta ventajosa para el aprendizaje de las diferentes asignaturas interrelacionadas. A modo aclaratorio, la Tabla 1 muestra algunos ejemplos sobre cómo integrar las técnicas de diseño de algoritmos en nuestro curso.

Cuando las diferentes operaciones sobre imágenes se explican en las clases teóricas, se motiva a los alumnos con ejemplos prácticos y se muestra el carácter interdisciplinar de la asignatura mediante diversos ejemplos de aplicaciones industriales reales. Algunos de estos ejemplos simplificados se han utilizado como proyectos, para ser realizados por grupos de alumnos al final del curso.

Nosotros consideramos importante recalcar los aspectos de eficiencia de los algoritmos de imágenes. Tal vez, un algoritmo inicial diseñado usando una técnica particular no es aplicable directamente en la práctica. Por ejemplo, esto ocurre cuando se usa "backtracking" con objeto de implementar un método de etiquetado de regiones conexas en una imagen digital. En este caso, se identificará y explicará un algoritmo más eficiente, basado en el uso de la programación dinámica. Es útil para los estudiantes aplicar esta estrategia de "refinamiento" de la solución para descubrir un mejor algoritmo con respecto a los requerimientos de tiempo de ejecución y memoria.

Técnica de diseño algorítmica	Ejemplo de su uso en la asignatura
Divide-y-vencerás	Filtro de mediana (realzado) *Split-and-merge* (segmentación)
Técnica voraz	Códigos de Huffman (compresión) Árbol de recubrimiento (clasificación)
Backtracking	Componentes conexas (reconocimiento) Crecimiento de regiones (segmentación)
Programación dinámica	Contorno óptimo (extracción de características) Detección de líneas (extracción de características)
Transformación del dominio	FFT (filtrado en el dominio frecuencial) Transformada de Hough (segmentación)

Tabla 1. Técnicas de diseño de algoritmos asociadas a distintos ejemplos de procesamiento de imágenes.

4. Prácticas del curso

El curso requiere ser complementado con prácticas de laboratorio y con proyectos prácticos. En nuestra universidad, este curso se impartió por primera vez entre Octubre de 1999 y Febrero de 2000. Debido al reducido número de estudiantes (aproximadamente veinte siguieron el curso), los recursos de hardware y software disponibles eran reducidos. Estos han consistido en cuatro PC (Pentium III) cada uno con una tarjeta digitalizadora Matrox, dos cámaras de vídeo convencionales, MATLAB y el software de digitalización y tratamiento de imágenes Matrox Imaging Libraries (MIL).

Las prácticas de laboratorio consisten en la realización de una o varias operaciones básicas, descritas previamente en las clases de teoría, sobre imágenes digitales. Tras la descripción en profundidad de una clase de operaciones sobre imágenes y sus algoritmos asociados, el énfasis se pone en ayudar a los alumnos a descubrir cómo implementar estas operaciones. Es muy importante recalcar la eficiencia de los algoritmos de imágenes [7]. Tal vez un algoritmo diseñado inicialmente, de forma menos eficiente, usando una técnica particular no sea aplicable en la práctica. En este punto, se explicará un nuevo algoritmo más apropiado para la resolución del mismo problema. Nosotros consideramos útil que los estudiantes aprendan "por refinamiento" a identificar un mejor algoritmo en relación con los requerimientos de tiempo y memoria.

Estas prácticas propuestas son individuales y diferentes para cada alumno. Cada enunciado de práctica contiene una descripción corta de su propósito, los datos de entrada, los resultados que debe proporcionar, y en algunos casos la(s) técnica(s) de diseño de algoritmo(s) que se han de utilizar con objeto de tener una implementación más eficiente. Muchas de las prácticas están orientadas a una técnica de diseño específica (por ejemplo, "divide y vencerás", búsqueda con retroceso, etc.), y los estudiantes debían explicar también por qué la técnica elegida resulta aplicable en relación con la eficiencia de la solución. De las prácticas propuestas en el curso, resumimos tres a continuación.

- *Componentes conexas de una imagen*:
  Una de las operaciones más comunes en tratamiento de imágenes digitales consiste en encontrar las componentes conexas en una imagen [5]. Por ejemplo, en tareas de segmentación, los puntos (píxeles) en una componente conexa forman una región

candidata para representar un objeto (o parte de él) en una imagen. Un algoritmo de etiquetado de componentes conexas encuentra todas las componentes de una imagen y asigna una única etiqueta a todos los puntos que están en la misma componente. En esta práctica, se pide a los alumnos que diseñen e implementen un algoritmo recursivo para el problema propuesto siguiendo el esquema de "backtracking" o búsqueda con retroceso [2], dado un píxel de comienzo. También se pide a los estudiantes que identifiquen otro algoritmo equivalente y más eficiente (no recursivo) usando una estructura de datos auxiliar (árbol cuaternario o "quadtree") para almacenar los vecinos de cada píxel. La Figura 1 muestra una imagen binaria y su correspondiente representación usando un "quadtree".

- *Filtro de mediana:*
  El filtro de mediana es un filtro de rango, que sustituye cada píxel de una imagen por la mediana de los valores de gris en su vecindad local [1]. Si la vecindad de un píxel se considera de tamaño 3×3, esta región se puede convertir en un vector, que se ordena primero por valores de intensidad no decreciente, y luego se selecciona el valor de intensidad situado en la posición mediana de dicho vector. Se pide a los estudiantes que diseñen un algoritmo de ordenación eficiente que siga la técnica "divide y vencerás" [2] para obtener la mediana, y luego se aplica dicho filtro a toda la imagen original para obtener una nueva imagen filtrada.

- *Códigos de Huffman:*
  El objetivo de esta práctica es construir la representación comprimida de una imagen mediante la técnica de los códigos de Huffman [4]. La compresión de Huffman [3] de una imagen es una técnica de codificación sin pérdida. Los valores de intensidad cada píxel son sustituidos por códigos de longitud variable basados en la frecuencia de ocurrencia para cada valor de intensidad en la imagen dada. Una tarea inicial para los estudiantes consiste en obtener el histograma de la imagen con las frecuencias de ocurrencia de cada valor de intensidad en la imagen. Se sugiere, a continuación, un algoritmo voraz [2][3] que permita asignar los códigos más cortos a los valores de intensidad con ocurrencias más frecuentes. Finalmente, hay que realizar un análisis de la eficiencia del algoritmo desarrollado.

5. Proyectos del curso

En esta sección se describen de forma simplificada algunos proyectos propuestos que deben ser realizados por los alumnos, en grupos de tres personas, asignándose a cada grupo de trabajo un proyecto diferente. El principal objetivo de estos proyectos, es abordar la realización de aplicaciones reales simplificadas. Estos proyectos en grupo son una excelente oportunidad para que los alumnos intercambien conocimientos de la asignatura con sus compañeros y practiquen las técnicas y algoritmos estudiados durante el curso. También hemos recalcado que en las aplicaciones industriales el tiempo de cálculo es un factor crítico [7]. Para motivar a los estudiantes sobre la importancia de implementar programas

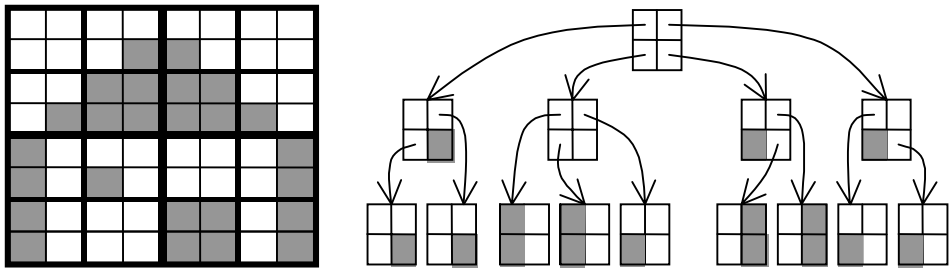


Figura 1. Ejemplo de una imagen binaria y su correspondiente representación "quadtree" para segmentarla.

eficientes, se les pide algún tipo de medida de los tiempos de ejecución. En particular, cada grupo de alumnos debe estimar la complejidad computacional de los algoritmos de imágenes involucrados en su proyecto particular. Además, el tiempo de ejecución de los programas desarrollados se debe calcular (usando una función de tiempo, como el comando UNIX `time` o el más sofisticado `gethrtime`). De los proyectos propuestos resumimos brevemente dos de ellos a continuación.

- *Reconstrucción morfológica de imágenes de códigos de barras:*
  Leer un código de barras de una imagen que ha sido escaneada previamente es un problema bien conocido en la práctica. Dicho problema lleva asociados otros debidos a la técnica no especificada usada por los escáneres para capturar las imágenes. Generalmente, éstos capturan las imágenes en niveles de gris, mientras que los lectores de códigos de barras trabajan sobre imágenes binarias (monocromas, o en blanco y negro). La conversión de una imagen en niveles de gris a una imagen binaria se denomina umbralización. Los escáneres avanzados usan procedimientos de umbralización regional para tal operación. Dichos métodos realzan la calidad visual de las imágenes eliminando zonas oscuras homogéneas. Sin embargo, tienen efectos perniciosos sobre los códigos de barras, precisamente porque los códigos de barras son zonas oscuras homogéneas (como se observa en la Figura 2).

  Para el realce de la región de la imagen que contiene el código de barras, los alumnos han de proponer e implementar un procedimiento de reconstrucción. Primero, el sistema debe encontrar la región de la imagen que contiene el código de barras (por ejemplo usando un algoritmo de "*split-and-merge*"). Luego, se aplica un procedimiento de reconstrucción (por ejemplo basado en la operación de dilatación morfológica). El tiempo total de respuesta debe ser menor que un segundo por imagen en un PC estándar.

- *Extracción automática de ayudas visuales en imágenes de pistas de aterrizaje:*
  Un sistema que realiza extracción automática de objetos relevantes en imágenes de pistas de aterrizaje tiene numerosas aplicaciones. Puede ser útil en aplicaciones de importancia práctica, como el mantenimiento automático de la iluminación de las pistas de vuelo o para asistir al piloto en aviones no equipados con ILS (Sistemas de Aterrizaje Instrumental, en inglés "Instrumental Landing Systems"). Para simplificar el problema, sólo se consideran imágenes nocturnas de pistas de vuelo, de manera que aparecen iluminadas artificialmente para destacar los elementos de interés (ver Figura 3).

  Se pide a los alumnos que construyan un sistema que haga una extracción automática de ciertos objetos luminosos relevantes en imágenes de pistas de aterrizaje, agrupando los objetos en líneas (ver Figuras 3 y 4). La organización de este proyecto tiene dos etapas

Figura 2. En la imagen de la izquierda, se muestra un código de barras degradado debido al uso de umbralización regional. En la imagen de la derecha, se muestra el resultado de la construcción del mismo código de barras usando una dilatación morfológica.

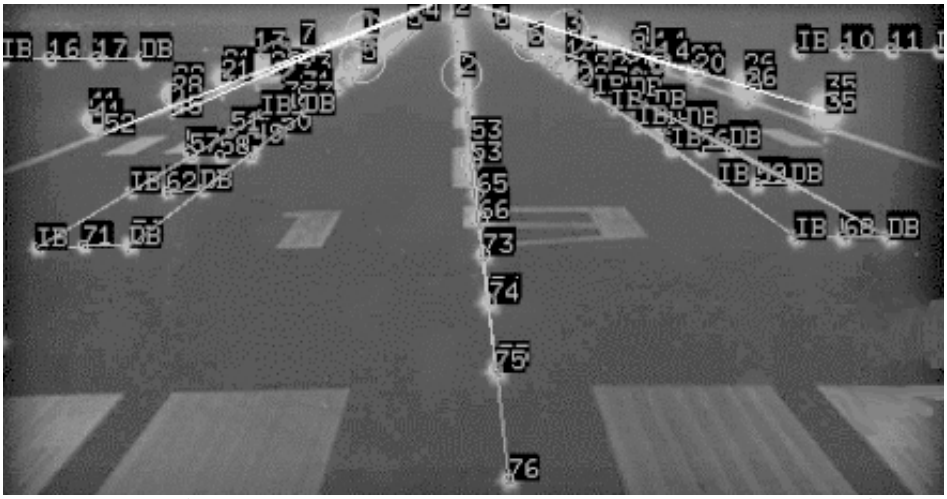


Figura 3. Imagen final segmentada con las ayudas visuales luminosas detectadas y agrupadas en líneas.

principales: 1) segmentación de las ayudas visuales luminosas (por ejemplo usando un procedimiento de umbralización), y 2) asociación en líneas de elementos de luz encontrados (por ejemplo utilizando la transformada de Hough) según un modelo propuesto de pista de aterrizaje.

6. Conclusiones

Este trabajo presenta un curso introductorio a la "Visión por Computador" con una marcada orientación algorítmica y práctica. Los requisitos de eficiencia (en tiempo de respuesta y memoria) de las aplicaciones son muy importantes y restrictivos en los algoritmos de

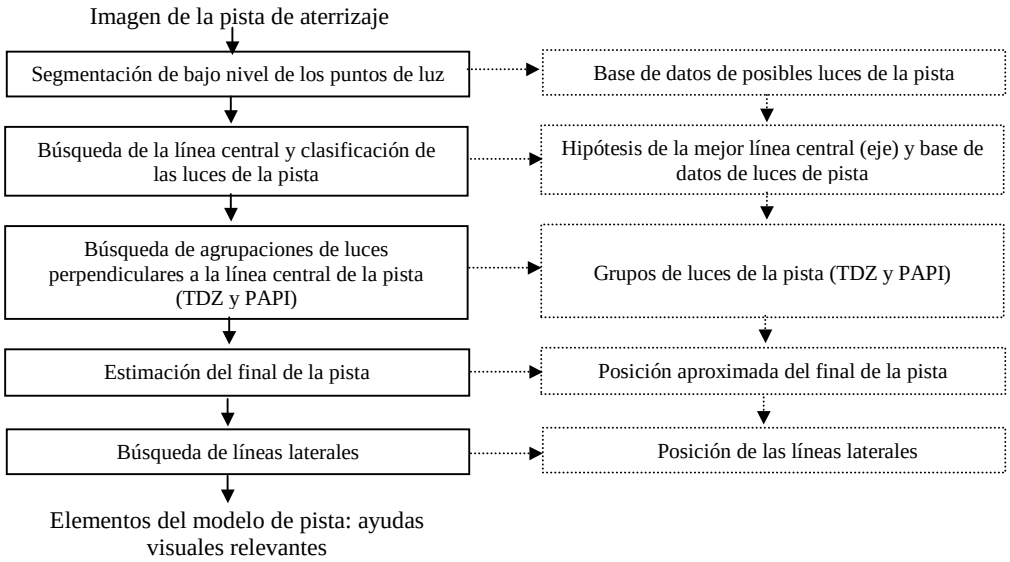


Figura 4. Diagrama de bloques del método propuesto de extracción automática de ayudas visuales relevantes.

visión. Por ello, los alumnos tienen que ser conscientes de lo importante que es el uso de algoritmos eficientes en sistemas de tratamiento de imágenes. Las técnicas de diseño de algoritmos pueden ayudar a idear dichos algoritmos y, en algunos casos, también ofrecen las soluciones óptimas para un problema de tratamiento de imágenes. Las propuestas de proyectos para ser realizados por grupos de alumnos, estaban pensadas como simplificaciones de aplicaciones industriales reales, lo que motivó enormemente a los alumnos de la titulación informática. Esto también les ha permitido analizar e implementar de forma sistemática y estructurada soluciones realistas. En algunos casos, parte de los proyectos de este curso han sido también diseñados y implementados eficientemente usando técnicas de diseño de algoritmos.

El énfasis puesto en la relación entre imágenes y algoritmos ha hecho que el curso sea más atractivo para los alumnos informáticos, quienes además se han acercado a aplicaciones de la industria. El enfoque algorítmico adoptado ha dado como resultado un porcentaje alto de alumnos aprobados en la asignatura.

Esperamos mejorar este curso cuando se vuelva a impartir en los próximos años.

Referencias

[1] G.A. Baxes, *Digital Image Processing. Principles and Applications*, J. Wiley, 1994.
[2] G. Brassard y P. Bratley, *Algorithmics. Theory and Practice*, Prentice-Hall, 1988.
[3] T. H. Cormen, C. E. Leiserson y R. L. Rivest, *Introduction to Algorithms*, The MIT Press, 1990.
[4] R.C. González y R.E. Woods, *Digital Image Processing*, Addison-Wesley, 1992.
[5] A.K. Jain, *Fundamentals of Digital Image Processing*, Prentice-Hall, 1989.
[6] R. Jain, R. Kasturi y B.G. Schunk, *Machine Vision*, McGraw Hill, 1995.
[7] I. Kabir, *High Performance Computer Imaging*, Manning, 1997.
[8] D. Maravall, *Reconocimiento de Formas y Visión Artificial*, Ed. Ra-ma, 1993.
[9] The MathWorks Inc., *Matlab v. 5.1*, 1997.
[10] Matrox Electronic Systems Ltd, *Matrox Imaging Library Reference Manual*, Ver 6.1, 1999, http://www.matrox.com/imaging/prod/mil/.

Organización curricular y planes de estudio

Enfoque diacrónico para la enseñanza de la programación imperativa

L. Fernández Muñoz (a), R. Peña (b), F. Nava (c), Á. Velázquez Iturbide (c)
(a) Dept. LPSI. Universidad Politécnica. 28071 Madrid e-mail: setillo@eui.upm.es
(b) Facultad de Documentación. Universidad de Alcalá. 28871 Madrid e-mail: rpr@uah.es
(c) Escuela Superior de Ciencias Experimentales. Universidad Rey Juan Carlos. 28933 Madrid
e-mail: {f.j.nava,a.velazquez}@escet.urjc.es

Resumen

Presentamos una alternativa para enseñar la programación imperativa no orientada a paradigmas; proponemos un enfoque diacrónico basado en la exposición justificada de cada constructor de los lenguajes de programación a través de su evolución histórica, motivada por los conceptos recurrentes que subyacen a los mecanismos particulares de cualquier paradigma.

Especificamos los objetivos que debe perseguir esta docencia, así como los contenidos que debe cubrir. Para alcanzar los objetivos empleamos herramientas de desarrollo y visualización del software, sobre un lenguaje flexible que se adecua sintáctico-semánticamente al avance de la exposición de la materia. La adecuación del lenguaje a los conceptos impartidos minimiza el tiempo dedicado al aprendizaje de su sintaxis y los recursos pedagógicos empleados ayudan a asentar los conocimientos.

La estructura propuesta elimina la dificultad que supone el cambio de paradigma, encontrada en experiencias que presentan el paradigma procedimental en primer curso y el orientado a objetos (OO) en segundo, y la sobrecarga inicial de conceptos que implica empezar directamente con el paradigma orientado a objetos.

1. Introducción

En el momento actual existe un fuerte debate en torno a la planificación de la enseñanza de la programación imperativa en los primeros cursos de los estudios de informática. En un trabajo previo [6] hemos analizado las experiencias docentes publicadas, tropezando con inconvenientes que impiden seleccionar, entre ellas, el enfoque adecuado.

Entendemos que la discusión sobre la planificación no debe centrarse en el paradigma a presentar en primer lugar, o en el lenguaje empleado para las prácticas, sino en los objetivos perseguidos y en el modo de conseguirlos, por lo que realizamos una nueva propuesta docente.

Este artículo se inicia presentando muy brevemente, en su apartado 2, los problemas que plantean otras propuestas docentes. El apartado 3 describe el enfoque diacrónico: sus objetivos y la planificación del contenido a impartir. El apartado 4 describe la metodología adecuada para la consecución de estos objetivos.

2. Revisión de las experiencias docentes publicadas

Durante muchos años el paradigma procedimental ha sido el adoptado para el primer acercamiento a la programación. En los últimos veinte años la mayoría de los centros de enseñanza universitaria de Informática ha integrado conceptos de OO en cursos superiores de sus planes de estudio. Pero los alumnos presentan dificultades al cambiar de paradigma, viviendo como una ruptura la exposición al nuevo, lo que se ha llamado el "problema del desplazamiento de paradigma".

Para evitarlo, algunos autores han optado por experimentar el primer acercamiento a la programación directamente desde el paradigma

OO. Este enfoque conlleva la necesidad de exponer muchos y novedosos conceptos al alumno antes de poder ponerlos en práctica y asentarlos, por tanto, supone una sobrecarga de conceptos que puede resultar desbordante.

Algunas de las propuestas docentes que empiezan con OO se basan en la manipulación de universos virtuales, aprovechando el supuesto carácter "intuitivo" del paradigma orientado a objetos. Este tratamiento conlleva una sobresimplificación que genera expectativas irreales.

Otras experiencias de OO desde primero posponen la programación, basando el desarrollo del curso en análisis y diseño. Pero el lenguaje determina el pensamiento, y resulta imposible diseñar sin disponer de un lenguaje subyacente. La OO es intuitiva porque utiliza los mecanismos habituales en la adquisición del conocimiento, pero un diseño eficiente (tanto en el momento del desarrollo, ejecución como en el mantenimiento) no suele ser paralelo a la percepción informal del mundo real y, por tanto, un alumno de primer curso no está mejor preparado para comprender este paradigma que otros.

Otras propuestas de OO en primero empiezan directamente con programación. Usan patrones, frameworks y bibliotecas para fomentar la abstracción, encapsulación, modularización, y reutilización. Los objetivos son buenos, pero sin duda, el empleo de estas herramientas requiere del dominio de muchos conceptos de los que el principiante no dispone.

En las propuestas que se centran en desarrollo de interfaces gráficas de usuario, además del problema que acabamos de mencionar, se maneja un modelo de eventos globalizado -applets- o una arquitectura sin eventos, que no tienen envergadura para resolver problemas reales y posiblemente fomentarán hábitos inadecuados.

Paralelamente, las experiencias presentadas proponen tres soluciones para el lenguaje de soporte de las prácticas. La primera emplea un lenguaje distinto para cada uno de los paradigmas a presentar. Tiene el inconveniente de que es necesario desviar, continuamente, la atención de los conceptos presentados hacia la sintaxis del nuevo lenguaje.

La solución de emplear un lenguaje OO desde el principio implica necesariamente el empleo de conceptos desconocidos por el alumno.

La tercera vía emplea lenguajes híbridos, que en principio, parecen resolver los problemas de las anteriores soluciones, pero generan confusión en el alumno a la hora de ubicarse en uno u otro paradigma, ya que el programa "compila" y "funciona" aunque el diseño no se adecue al paradigma.

3. Enfoque diacrónico

3.1. Justificación y presentación

Partimos de que "las teorías científicas son entidades que se extienden o perduran en el tiempo, que permanecen a través del cambio. Ello supone que el estudio puramente sincrónico que las considera como entidades estáticas, *congeladas*, constituye sólo una primera aproximación que se debe completar con un análisis diacrónico que dé cuenta del carácter persistente de estas entidades" [5].

Esta premisa de la filosofía de la ciencia, en el mundo de la programación, se reescribe en los siguientes términos: "cada paradigma se construye sobre los que lo han precedido, añade algo nuevo al arsenal de herramientas del programador [...] y refleja un planteamiento de diseño" [11]. En particular, no existen dos paradigmas enfrentados, estructurado y objetos; existe un único modelo imperativo cuya evolución genera nuevos enfoques persiguiendo lo mismo: la mejor gestión del software. En palabras de Lewis [9] “El enfoque OO no abandona los conceptos que admiramos en el enfoque procedimental, los aumenta y los fortalece".

La transición entre paradigmas es natural y evolutiva pero algunos alumnos lo encajan como un cambio total. Nuestra experiencia en la docencia de ambos paradigmas demuestra que el problema del desplazamiento no es global. Los alumnos aventajados en la programación procedimental asumen con naturalidad el nuevo paradigma, entendiéndolo como algo lógico y casi esperado. Este hecho queda patente con la frase de Alan Kay, creador de Smalltalk, cuando

evaluó SIMULA-67: "el impacto fue tan grande que fue la última vez que pensé en términos de subrutinas y estructuras" [10].

La conjunción de la dualidad entre la dificultad inicial mayoritaria y la naturalidad del cambio para alumnos aventajados, conduce a un nuevo planteamiento: el problema del desplazamiento afecta a los alumnos no aventajados. Agrupamos en esta denominación, los que superan la asignatura del paradigma procedimental y usan correctamente los constructores del lenguaje, cómo, cuándo y dónde usarlos, pero carecen de una visión más profunda y general de la programación. No han asumido el porqué de los constructores del lenguaje, sus implicaciones y sus carencias. Desde una perspectiva más general, tampoco han asumido que "ciertos conceptos fundamentales son recurrentes a través de toda la disciplina... son ideas significativas, cuestiones, principios y procesos que ayudan a unificar una disciplina académica profundamente" [1]. El informe "Computing curricula'91" ha establecido los siguientes 12 conceptos recurrentes: ordenación en el espacio y tiempo, enlace, niveles de abstracción, completitud, evolución, complejidad, compromisos, seguridad, modelos, reusabilidad y eficiencia.

Tradicionalmente, los primeros cursos de programación se centraban en la exposición de los mecanismos concretos de un lenguaje, no en los conceptos recurrentes de la Informática. "Un concepto recurrente es más fundamental que cualquiera de sus ejemplarizaciones. Se sustenta en sí mismo como fundamental, persistente a lo largo de la historia de la computación, y es muy probable que permanezca en un futuro" [1].

La incorporación de la OO en los planes de estudio evidenció las carencias de estos cursos de introducción a la programación, reflejadas por el problema del desplazamiento del paradigma, al incluir nuevas terminologías y objetivos de diseño: evolución, reusabilidad, jerarquización, acoplamiento, encapsulación, etc. La primera respuesta a este problema fue invertir el orden de exposición, trayendo otras carencias en su lugar: la sobrecarga y, en algunos casos, el desconcierto o la simplificación excesiva de la programación OO en base a su carácter intuitivo, e incluso, la ausencia de la exposición de la programación procedimental.

Entonces, el problema del desplazamiento radica en la ruptura del contenido de la exposición: con el cambio del paradigma, la terminología es completamente nueva (abstracción, encapsulación, modularización, jerarquización...), el lenguaje también es nuevo o inadecuado y los objetivos de ambos paradigmas parecen diferentes. El exponente más dramático de esta situación se recoge en [7]: "algunos profesores comienzan su primera clase con *olviden cualquier cosa que sepan. OO es radicalmente diferente*. Es ilógico". No hay nada que desaprender, la OO no supone una revolución de conceptos, sino una evolución.

Tras detectar esta situación, hemos modificado nuestros cursos de OO, comenzando con un profundo análisis del grado de concreción de los conceptos recurrentes que exhibe el porqué, implicaciones y carencias de los mecanismos del lenguaje procedimental (sistema de tipos, registros, subprogramación, módulos, etc). Para fortalecer el carácter diacrónico desarrollamos un nuevo lenguaje OO, con la misma sintaxis y semántica del lenguaje empleado en primero, para los conceptos asumidos de la programación estructurada, pero que introduce los mínimos cambios necesarios para el nuevo paradigma. El aprovechamiento de los alumnos, sus resultados académicos, y las encuestas de evaluación de la docencia mejoraron notablemente.

En base a esta experiencia, en este trabajo proponemos una tercera vía, un enfoque diacrónico basado en la exposición justificada de cada concepto de la programación a través de su evolución histórica motivada por los conceptos recurrentes que subyacen a los mecanismos particulares de cualquier paradigma de la programación.

Esta manera de exponer la programación en los cursos de introducción mejora la comprensión y mitiga el impacto de la transición al paradigma OO. Los cambios propuestos afectan no tanto al índice del curso como a sus objetivos, a la terminología y al lenguaje y naturaleza de las prácticas. Un ejemplo de esta idea es que el concepto de enlace (estático o dinámico) no surge al presentar la sobrecarga y

el polimorfismo, que es hasta donde se suele postergar su introducción, sino que ya está presente al ligar los nombres de constantes y variables a su valor. En nuestra propuesta, el concepto de enlace, al igual que el resto de conceptos recurrentes, debe ser resaltado desde sus primeras concreciones, y en todas y cada una de ellas. Con esto nos diferenciamos del enfoque procedimental en primero y OO en segundo, ya que presentamos cada constructor como una justificación conjugada de los mismos conceptos recurrentes, independientemente del paradigma al que pertenezca.

3.2. Objetivos

El enfoque diacrónico posibilita el cumplimiento de los siguientes objetivos para mitigar las desventajas de las propuestas anteriores:

- Una exposición escalonada y paulatina de los conceptos de la programación en sintonía con la madurez del aprendiz, evitando la sobrecarga inicial, fluyendo a través de la programación estructurada, modular, basada en objetos, orientada a objetos, etc. "El aprendizaje debería estar basado en el conocimiento previo de los programadores" [4]. "La construcción del conocimiento se forja recursivamente sobre el conocimiento que los estudiantes ya tienen" [2];
- Anulación del impacto del "desplazamiento de paradigma" de programación estructurada a OO gracias a un hilo conductor: los conceptos recurrentes; "la estimación de la penetración de estos conceptos y la facultad para aplicarlos en un contexto apropiado es un indicador de la madurez de los graduados" [1].
- Herramientas adecuadas para la participación en el desarrollo temprano de aplicaciones motivadoras (potentes y con interfaces gráficas de usuario), pero evitando la exposición a conceptos más complejos que sus capacidades.

La figura 1 presenta un esquema de la evolución de los enfoques discutidos.

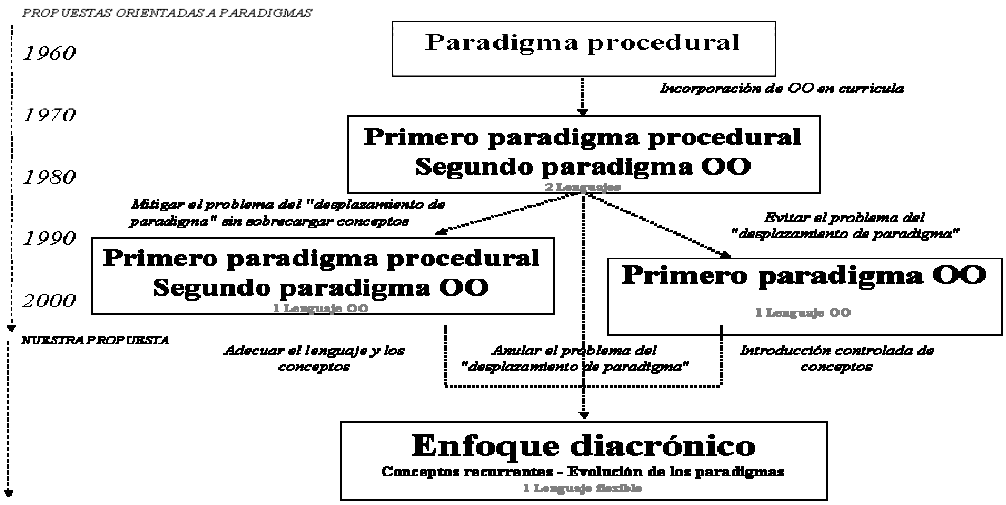


Figura 1 Evolución de los enfoques de la enseñanza de los paradigmas

3.2. Planificación

Los contenidos expuestos deben asegurar la comprensión de los conceptos recurrentes de la Informática y establecer su grado de concreción en cada mecanismo de la programación. "En el diseño de un currículo concreto, estos conceptos recurrentes deben comunicarse de manera efectiva; es importante notar que el uso apropiado de los conceptos recurrentes es un elemento

esencial en la implantación del currículo y cursos" [1].

Proponemos una previa exposición de estos conceptos mediante analogías con el mundo real. Por ejemplo: complejidad y evolución de las normas de tráfico; modelos formales en el "lenguaje de un semáforo"; enlace estático entre un vehículo y su propietario o dinámico con su conductor. A través de toda la exposición de la materia se vuelve a incidir en el papel de los conceptos recurrentes. Un ejemplo es exponer el concepto de seguridad desde el sistema de tipos, pasando por las precondiciones y poscondiciones de las sentencias de control de flujo de ejecución y de los subprogramas, las invariantes de los bucles, hasta llegar al interfaz de los módulos, tipos abstractos de datos y clases.

Como hemos comentado anteriormente, exponer el concepto de enlace estático y dinámico presente entre las constantes, variables o expresiones y sus tipos, valores..., pasando por el enlace entre los tipos genéricos a sus tipos concretos, hasta el enlace en la sobrecarga y el polimorfismo con mensajes a métodos.

O exponer los conceptos de niveles de abstracción, el control de la evolución, la posible reusabilidad y la resolución de compromisos, desde las primeras oportunidades en la solución de los problemas con registros, subprogramación, programación modular, etc.

Por tanto, la terminología del alumno de primer curso debe incorporar términos propios del diseño como patrones, modularidad, encapsulación, acoplamiento, cohesión, abstracción, jerarquización, legibilidad, fiabilidad... y saber evaluarlos en cada mecanismo de la programación.

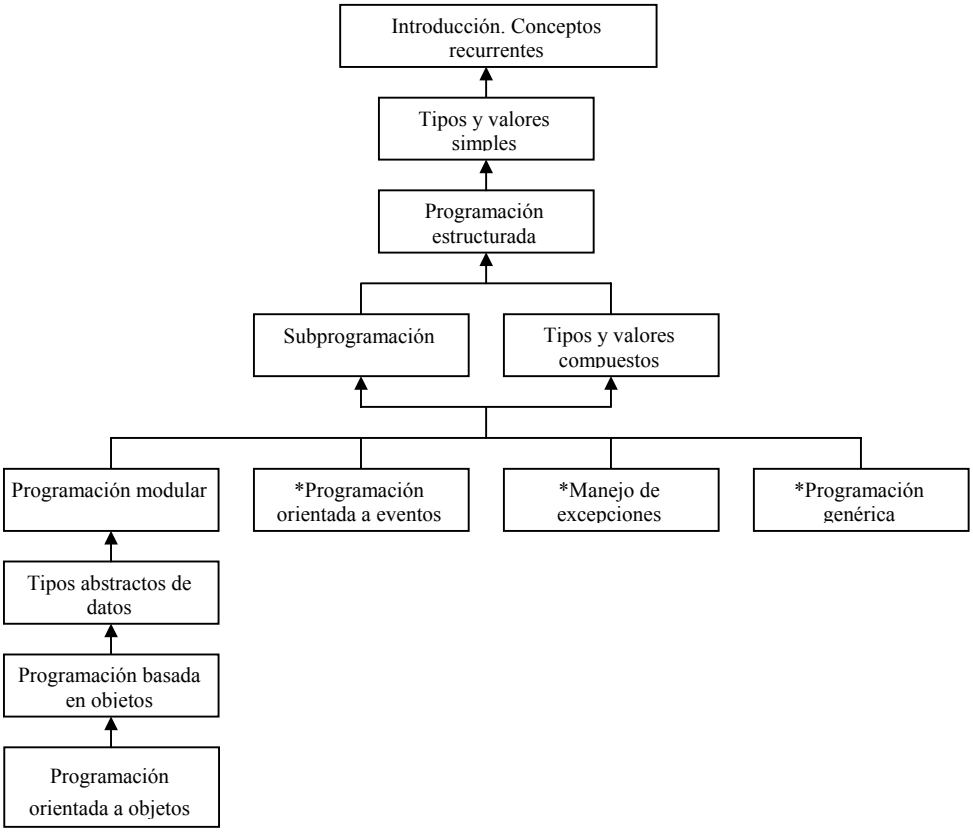


Figura 2.Prerrequisitos entre los núcleos teóricos del enfoque diacrónico.

Acorde al carácter diacrónico basado en la filosofía de la ciencia y el constructivismo de la psicología de la programación, nuestra planificación contiene un conjunto de núcleos cerrados de conceptos en que cada uno viene a resolver los inconvenientes evidenciados y subrayados en las soluciones de programas construidos con los conceptos anteriormente disponibles.

Los 12 núcleos considerados son:

- Introducción: conceptos recurrentes: datos y procesos;
- Tipos y valores simples: constantes y variables; expresiones, precedencia y asociatividad;
- Programación estructurada: sentencias de control de flujo de ejecución;
- Subprogramación: parámetros, Entrada/Salida, recursividad;
- Tipos y valores compuestos: registros y tablas; estructuras de datos dinámicas; ficheros;
- Programación genérica*: plantillas y concreción de tipos;
- Programación modular: interfaz e implantación; acoplamiento y cohesión;
- Manejo de excepciones*: elevación, delegación y captura;
- Programación orientada a eventos*: elevación, delegación y captura;
- Tipos abstractos de datos: múltiple instanciación
- Programación basada en objetos: clases y objetos; sobrecarga.
- Programación orientada a objetos: herencia y polimorfismo.

Esta planificación incluye núcleos opcionales (marcados con'*') que se introducirán en mayor o menor grado, dependiendo de la profundidad deseada en cada curricula. De modo que la figura 2 debe interpretarse como una tabla de prerrequisitos entre núcleos que posibilitan diferentes secuencias de exposición y su distribución, en dos o tres cursos, según convenga en un centro o titulación concreta.

4.Metodología

4.1. Lenguaje flexible

El lenguaje de programación debe ser ejecutable, permitir el desarrollo de aplicaciones motivadoras, pero flexible para incorporar nuevos mecanismos y eliminar otros correspondientes a conceptos subsumidos con el avance del curso. Por ejemplo, este lenguaje debe mantener la tabla de precedencia de operadores, las mismas sentencias de control de flujo de ejecución..., pero debe inhibir los registros y los subprogramas libres al introducir la programación basada en OO, etc.

Por tanto, debe cambiar las palabras reservadas correspondientes a la organización de la estructura de los programas, pero mantener inalterable las reglas sintáctico-semánticas de aquellos conceptos asumidos en los nuevos enfoques: precedencia de operadores, sentencias de control de flujo de ejecución, etc.

4.2. Prácticas

"La teoría dominante de la enseñanza hoy día, llamada constructivismo, afirma que el conocimiento se construye activamente por el estudiante, no se adquiere pasivamente desde los libros de texto o en las aulas" [2]. Así "enfatizar las actividades de lectura y depuración, simultáneamente a las de programación puede iluminar concepciones" [8].

"El aprendizaje cognitivo de la taxonomía de Bloom es útil en la estructuración del principio del currículo informático. Cada nivel de la jerarquía está subsumido por el siguiente nivel, de modo que, funcionalidades más altas exigen habilidad en los niveles inferiores" [3]. Las etapas en dicha taxonomía son:

- Conocimiento: el recuerdo de material previamente aprendido
- Comprensión: la habilidad para extraer el significado del material
- Aplicación: la habilidad para usar el material aprendido en una situación nueva y concreta
- Análisis: la habilidad para descomponer un material en sus componentes y comprender su estructura organizativa
- Síntesis: la habilidad para poner partes juntas formando un todo
- Evaluación: la habilidad de juzgar el valor de un material para un propósito determinado.

La vertiente práctica de nuestra propuesta respeta la aproximación constructivista a través de

los niveles de aprendizaje para cada núcleo teórico de la planificación. Por ejemplo, para la docencia de las estructuras de control de ejecución:

- Conocimiento: sintaxis y semántica de las estructuras
- Comprensión: dados unos pequeños bloques de código, extraer el control de flujo de ejecución y sus efectos sobre el entorno (variables, periféricos, etc.)
- Aplicación: escribir las estructuras que satisfacen unos requisitos específicos
- Análisis: dados unos bloques de código, extraer su organización para comprender los requisitos que satisfacen
- Síntesis: dada una aplicación incompleta, dotarla de la funcionalidad deseada
- Evaluación: contrastar soluciones alternativas, con estructuras y sin estructuras –por ejemplo con sentencia *goto*- y, por otro lado, presentar soluciones "inadecuadas" con 100 líneas de código de estructuras anidadas, evidenciando las bondades y carencias de las estructuras de control de flujo de ejecución en términos de los conceptos recurrentes: complejidad, ordenación en el espacio, evolución, niveles de abstracción, etc.

  Para la consecución de los objetivos consideramos vitales los dos últimos niveles.
- Síntesis: el alumno incorpora código en aplicaciones motivadoras, con interfaces gráficos de usuario, bibliotecas, sistemas de comunicaciones, etc. Todo ello, resaltando la aplicabilidad del material aprendido en el "mundo real", pero, evitando la exposición y manejo de conceptos avanzados
- Evaluación: permite al alumno reconocer, en cada núcleo, el hilo conductor en nuevas concreciones de los conceptos recurrentes y justifica, visualizando las limitaciones, la necesidad de evolucionar a nuevos constructores del lenguaje.

4.3. Herramientas

Las herramientas necesarias para la concreción del enfoque diacrónico responden, paralelamente, a los niveles cognitivos de Bloom:

- Intérpretes para la visualización del software que apoyen los niveles de conocimiento y comprensión: enlace estático de tipos a expresiones, orden de evaluación de operadores en las expresiones; orden de ejecución de las estructuras; comunicación por el paso de parámetros; desencadenamiento de instancias y mensajes en objetos; etc.
- Depuradores para el nivel de aplicación.
- Herramientas CASE e ingeniería inversa para extraer la organización de software en el nivel de análisis: diagramas de jerarquías de estructuras, subprogramas, módulos, clases… adecuados al núcleo que se está impartiendo.
- Entornos de desarrollo con bibliotecas, compilador,… y herramientas que bloqueen la edición del código de una aplicación excepto en los ámbitos a codificar, para la fase de síntesis: codificar el cuerpo de un subprograma, la implantación de un módulo o un TAD, etc.
- Métricas del software para el nivel de evaluación que exhiban la complejidad y consumo de memoria; acoplamiento y cohesión de módulos, TAD's o clases; etc. de las soluciones implementadas.

Todas estas herramientas deben configurarse con un lenguaje evolutivo que incorpore e inhiba reglas sintáctico-semánticas, adecuándose a la secuencia de núcleos expuestos en la planificación.

5. Conclusiones

Con la incorporación de la OO en los planes de estudio han surgido inconvenientes, tanto en su asimilación por parte de los alumnos, como en la localización de recursos adecuados por parte del profesor (disponibilidad de textos, ejemplos que encajen en una sesión docente, incluso elección del lenguaje).

El enfoque, procedimental en primero y OO en segundo, evidenció el problema del "desplazamiento de paradigma". Posteriormente, el enfoque contrapuesto, OO en primero, trajo la sobrecarga inicial de conceptos para los alumnos y, en algunos casos, la inadecuación de los lenguajes a los paradigmas.

Hacemos una propuesta docente, no orientada a paradigmas, que elimina los inconvenientes de las anteriores, utilizando un enfoque diacrónico, que consiste en la exposición justificada de cada constructor de los lenguajes de programación, a través de su evolución histórica, motivada por los

conceptos recurrentes que subyacen a los mecanismos particulares de cualquier paradigma, de modo que cada nuevo concepto a presentar surge para resolver las limitaciones de los anteriores. Reclamamos la necesidad de usar herramientas pedagógicas que apoyen la docencia, basándose en un lenguaje eminentemente evolutivo. Este lenguaje incorpora y elimina, según el avance de la materia, las reglas sintáctico-semánticas de los mecanismos de la programación.

Por tanto, nuestra propuesta modifica:

- objetivos, en concordancia con el enfoque diacrónico de la filosofía de la ciencia, concretados en los conceptos recurrentes de la Informática;
- planificación y lenguaje, acorde al constructivismo de la psicología de la programación, donde la adquisición del conocimiento se construye sobre conocimientos previos;
- prácticas y herramientas, acordes a los niveles cognitivos de la taxonomía de Bloom, que propugna la lectura del código, la depuración, el análisis, la implantación de aplicaciones y la evaluación de soluciones.

Agradecimientos

Este trabajo se ha financiado con el proyecto TIC2000-1413 de la CICYT.

Referencias

[1] ACM/IEEE. Computing curricula 1991. http://www.computer.org/education/cc1991/eab1.html

[2] Ben-Ari, M. *Constructivism in computer science education.* 30 SIGCSE Technical Symposium on Computer Science Education, 1998

[3] Buck, D.; Stucki, D. *Design early considered harmful: Graduate exposure to complexity and structure based on levels of congnitive development.* SIGCSE 2000 3/00

[4] Détienne, F. *Software-design cognitive aspects.* Springer 2002

[5] Díez, J.A.; Moulines, C.U. *Fundamentos de la Filosofía de la Ciencia.* Ariel, 1997

[6] 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.

[7] Fernández, A.; Rossi, G. *An Effective approach to learning Object-Oriented technology* ECOOP 98 xxii+573pp

[8] Fleury, A.E. *Programming in Java: student-constructed rules.* SIGCSE 2000, 3/00

[9] Lewis, J. *Myths about OO and its pedagogy.* SIGCSE 2000, 245-49

[10] Sterwart, M.K. *The natural history of objects* C++. Report nov 1992 (suplemento) pp.12-13

[11] Stroustrup, B. *El lenguaje de programación C++* Addison Wesley, 1998

¿Qué podemos enseñar sobre TI y la Organización en Planes de Estudio de Informática?

Edmundo Tovar
Dept. de Lenguajes y Sistemas Informáticos e Ing. Software
Facultad de Informática
Universidad Politécnica de Madrid
28660 Boadilla del Monte
e-mail: etovar@fi.upm.es

Resumen

La incorporación de materias propias de un currículum de Sistemas de Información (SI) orientadas a la organización, como "Función Informática en la Empresa" o "Auditoría Informática" en una titulación de Ingeniería Informática tiene una especial problemática. La actualización de sus contenidos no puede ser realizada directamente sólo a partir de los modelos curriculares tradicionales, hasta estos momentos, de Informática. Esta ponencia explica cómo se ha llevado a cabo la selección de contenidos para la actualización de estas materias teniendo en cuenta modelos curriculares más específicos, la demanda de profesionales del mercado nacional y el Plan de Estudios vigente en el Centro.

1. Cambios habidos en modelos curriculares de la Disciplina de la Informática

Los estudios universitarios de la titulación de Ingeniería Informática en España se han basado tradicionalmente en los curriculums elaborados conjuntamente por los esfuerzos conjuntos de las asociaciones IEEE y ACM. El modelo referencia durante la última década ha sido el informe "Computing Curricula 1991"[1].
Pero la Informática durante este tiempo ha sufrido cambios drásticos. Continuamente se introducen nuevas tecnologías, y otras, por el contrario, se quedan obsoletas, con la consecuente repercusión en la docencia. Avances técnicos, como el de las aplicaciones para World Wide Web y las tecnologías de redes han incrementado la importancia de muchos temas curriculares.
Pero también ha habido cambios culturales y económicos, como los siguientes:

- La Informática se ha expandido enormemente gracias al desarrollo y las facilidades para el acceso a aplicaciones de software a través de Internet.
- Las Tecnologías de Información (TI en adelante) tienen una creciente importancia en la economía. Han surgido las empresas tecnológicas, y la industria demanda un gran volumen de personal cualificado.
- Hay una mayor aceptación de la Informática como una disciplina académica. La entrada de la TI en los ámbitos culturales y económicos han consolidado el reconocimiento de la disciplina.

El especial dinamismo de la Informática ha provocado también el cambio de los Planes de Estudios en Informática, así como la forma de enseñarlo. En la actualidad La informática abarca unos contenidos tan amplios que resulta difícil establecer los límites, e incluso considerarlo como una única disciplina.
Esta es la razón por la que en la última revisión del "Computing Curricula", (CC2001) [2] el modelo se ha desarrollado en 4 diferentes volúmenes. Sólo uno de ellos ha sido publicado, el de Computer Science, mientras que aún no lo han hecho los otros tres: Computer Engineering, Software Engineering e Information Systems. Según el criterio expresado por ACM e IEEE, todas estas especialidades se complementan perfectamente, por lo que la educación de la Informática debe cubrir este nuevo alcance de la Disciplina.

1.1. Sobre el perfil del Ingeniero en Informática

Un Ingeniero en Informática necesita de muchas habilidades además de las puramente técnicas. Por ejemplo, requiere poseer un sentido de cómo la tecnología se aplica en la práctica. Y el ámbito natural es el de los departamentos de Informática y, más allá, en las organizaciones. Los Ingenieros en Informática deben ser capaces de trabajar con gente de otras disciplinas de manera efectiva. En este sentido, el CC2001 recomienda que los estudiantes se formen junto a la profundización en algún área de aplicación, integrando estudios de casos, o incluyendo la adquisición de experiencia equivalente a un semestre completo en esa disciplina. Las oportunidades son amplias, pues pueden ir desde el campo de la Economía, al de la Psicología, o al de cualquier otra Ingeniería. Pero una de ellas llama poderosamente el interés: el mundo de los negocios. Esta experiencia es sin duda una de las principales características solicitadas por los empleadores, sobre todo en épocas en las que existe una alta demanda de graduados en esta disciplina. Las Universidades se sienten con cierta presión para asegurarse que sus graduados satisfacen las necesidades de las empresas. Pero no se trata de ser un maestro en alguna especialidad que pronto puede quedarse obsoleta. Profundizar en el campo de los negocios puede servir para conocer más en detalle el terreno con el que, con una muy alta probabilidad, tendrá que estar familiarizado. Hay diferentes razones para ello. Por ejemplo:

- ser más competitivo
- tener más a su alcance el dibujo global del entorno en el que tendrá que desarrollarse.
- tener una más fácil relación con la dirección de las empresas a las que pertenezcan
- facilitarles su transición a otros puestos más de gestión que los puramente técnicos que tienen nada más finalizar la carrera.

1.2. Cómo las Universidades responden

La forma habitual en la que se suele permitir que los estudiantes alcancen una visión de la organización más cercana a su futuro papel profesional es permitiéndoles que ellos elijan un número determinado de créditos con respecto de la oferta global de asignaturas optativas que se les proporciona.

Por tanto la selección de materias que componen la oferta de optatividad es un elemento clave. Pero el interés del futuro profesional no es el único factor a tener en cuenta para proponer asignaturas que completen un Plan de Estudios. Hay otros, como:

- El tipo de Universidad de que se trate y las expectativas para su Plan de Estudios. Hay que tener en cuenta el perfil del Ingeniero definido para esa titulación.
- El rango de opciones de estudios de especialidad que el egresado pueda realizar con posterioridad. El profesional relacionado con la TI puede tener diferentes objetivos curriculares que los que estudian para ser Ingenieros de Informática. Las Universidades deben tener en cuenta que la formación que da a sus alumnos tenga una continuidad en el aprendizaje continuo a lo largo de sus carreras.
- La preparación de los estudiantes que ingresan.
- El interés y la experiencia de su plantilla de profesores.

En particular en nuestro Centro, la Facultad de Informática de la UPM, el Plan de Estudios vigente data de 1996 y está dirigido a la obtención del título oficial de Ingeniero en Informática. Está estructurado en un primer ciclo de 2 años y un segundo de tres años. La carga lectiva global es de 381 créditos, 63 de los cuales son de materias optativas. El alumno debe elegir libremente 15 créditos de éstos durante tercer curso, 19,5 durante cuarto curso, y 28,5 durante quinto curso.

1.3. Las necesidades de formación en SI/TI en la FI-UPM

Las necesidades y el perfil de nuestros titulados debiera de aparecer en el Plan de Estudios, o al menos en los trabajos que le dieron lugar. Sin embargo esto no ha sido así, como lo testimonia el reciente proceso de evaluación de la calidad seguido en nuestro Centro [3]. Según las conclusiones recogidas en dicho informe, no se registró ninguna documentación al efecto: ni perfil profesional, ni planificación del proceso. Todo ello se manifiesta en una desorganización y

exceso de oferta de materias optativas. Es por esta razón por la que ha sido necesario recurrir a otras fuentes.

Actualmente existe un Grupo de Trabajo en el centro que está trabajando en la elaboración de un nuevo Plan de Estudios. Uno de sus primeros trabajos consistió en realizar una encuesta a través de la web que sondeara entre las empresas su opinión sobre la utilidad de materias que componen la carrera. No se obtuvo un alto número de respuestas a la encuesta por la que no se ha considerado suficientemente representativo. Además, para el objetivo de este trabajo no sería de gran utilidad puesto que las materias relacionadas con los de Sistemas de Información se encuentran distribuidas en distintas materias como Ingeniería Software y Gestión de Empresas.

Los resultados de las encuestas para el área de Organización de empresas sobre una muestra de 24 encuestas recogidas aparecen en la figura 1.

	Muy Poco Útil	Poco Útil	Útil	Muy Útil	Impres-cindible
Visión general de la empresa	4%	8%	33%	29%	25%
Tipologías de empresas	-	-	-	80%	20%
Comp. básicos (Estrategia, Planif.Organiz. Gestión,Control)	-	20%	40%	40%	-
La función informática	-	20%	40%	20%	20%
Modelos de negocio para empresas en red	Sin datos				

Figura 1. Resultados parciales del cuestionario formulado para la recogida de opiniones de profesionales

Sí se ha contado con un estudio realizado por la "Asociación de Doctores, Licenciados e Ingenieros en Informática" (ALI). Este estudio, que refleja el mercado laboral en Informática durante el año 2000, está basado en una muestra de 4931 ofertas de empleo (del sector público y privado) y que fueron publicadas durante el año 2000 por los diarios ABC, El País, La Vanguardia, Expansión, El Mundo; los semanarios de Informática Computing, ComputerWorld; así como las ofertas recibidas en la Secretaría Técnica de ALI.

En el estudio se han utilizado distintas denominaciones para agrupar diferentes perfiles o puestos de trabajo tal y como aparecen en los anuncios. De la relación global de denominaciones utilizadas sólo se hará referencia a aquellas que tienen una relación más directa con respecto de un conjunto de materias que requieren la visión del entorno organizativo. De ellas se destacan las siguientes:

- Director de Informática: este término agrupa ofertas dirigidas a Directores de SI/TI, Gerentes en TI, Responsables de departamento de Informática...
- Jefe de Proyecto: Incluye ofertas para Director de Sistemas de Procesos, Responsable de Nuevas Tecnologías, Responsable de aplicaciones informáticas, director de Arquitectura de aplicaciones, y Director tecnológico entre otros.
- Informático: especialista en TI Integrador de TI, Técnico en SI, experto en TI...
- Auditoría Informática: auditor de sistemas, técnico auditor de SI...

En la figura 2 aparecen los resultados parciales de este estudio.

Denomi-nación	Total de ofertas	Ofertas para Ing. Informát.	Ofertas para titulado de 2º ciclo
Director Informát.	1,2%	31,5%	47,4%
Jefe de Proyecto	11%	31,6 %	30%
Informát.	6,3%	19,1%	12,9%
Auditoría	0,7%	50%	15,6%

Figura 2. Resultados parciales del estudio de mercado laboral de ALI

Ambos estudios referenciados muestran resultados similares, aún siendo el segundo (el elaborado por ALI) más riguroso y sobre todo más fiable al incluir una muestra mucho más significativa.

- En el primero (Grupo de Trabajo de elaboración del Plan de Estudios), hay un acuerdo generalizado, dentro de la escasa

participación de profesionales, en la utilidad de esta visión para el ejercicio profesional.

- El segundo (estudio de ALI) requiere un análisis más profundo. Primero, entre los perfiles demandados por las empresas cabe destacar que muchos de ellos requieren de un conocimiento de la organización y de la integración de las TI y SI en la empresa. Segundo, de los distintos perfiles de Informática, el colectivo más demandado es el de Ing. en Informática. Pero no es esto lo más importante. Es significativo comprobar que entre los puestos de gestión como el de Director de Informática y Jefe de Proyecto se demandan titulaciones distintas a las de Ing. Informática en la misma o mayor proporción. Esto ocurre también en otras denominaciones de perfiles no incluidas en la figura 2, como los de Ingeniero Software o Técnico de Sistemas. Sólo existe una excepción para el caso de los auditores, donde la demanda mayoritaria corresponde a los Ing. Informáticos.

En resumen, se puede concluir que la titulación no es madura y la competencia con otras de 2º ciclo es muy fuerte, a pesar de que el ámbito de comparación se refiere exclusivamente a ocupaciones informáticas.

Una razón que explica el anterior diagnóstico, sobre todo en los trabajos de gestión, es la falta o deficiente formación específica que reciben nuestros titulados en temas relacionados con el uso de la Informática en el entorno empresarial. No cabe duda que parte de la formación necesaria para ocupar dichos puestos son recibidos en cursos de postgraduados. De hecho, según estudios de la UPM [4] más de un 40% de los titulados por esta Universidad ha considerado necesario realizar cursos de especialización para poder completar su formación, (un 3,6% en temas de Dirección). Pero la cuestión es facilitar la transición a estos estudios de especialización. Y eso sólo se puede conseguir fortaleciendo la formación, en los temas que nos ocupa, en los estudios que dan lugar a Ing. en Informática.

En esta ponencia se explica de qué manera se han seleccionado los contenidos para la actualización de dos materias relacionadas con los SI/TI del Plan de Estudios vigente en la Facultad de Informática de la UPM. Esta actualización aspira facilitar la transición de sus graduados a otros estudios de especialización adecuadas a las demandas del mercado. Para ello se tendrá en cuenta un modelo curricular de SI y se buscará la coherencia con el Plan de Estudios del Centro.

2. Educación en Sistemas de Información

Una característica general que se espera de los Ingenieros en Informática es que posean una perspectiva general del sistema y de la organización. Esta visión debe ir más allá de los detalles de la implementación y desarrollo de los diversos componentes y llegar a los procesos de negocio y la gestión de la organización en donde se instalan.

El tipo de cursos donde se pueden recoger este tipo de conocimientos se corresponde a un modelo multidisciplinar, pues intervienen áreas como el de la gestión, la economía y el negocio.

Los SI automatizados con ayuda del ordenador ha llegado a ser uno de las partes críticas de los productos, servicios y dirección de las organizaciones. El uso eficiente y efectivo de la TI es un elemento importante para lograr la ventaja competitiva entre las organizaciones de negocio y para la excelencia en los servicios de aquellas organizaciones que no tienen ánimo de lucro. Los SI son vitales para la toma de decisión en todos los niveles de la dirección de la empresa: Operacionales, tácticos y estratégicos [5], [6].

Ésta es la razón fundamental por la que se necesitan fuertes lazos entre los profesionales en los SI y TI y los respectivos programas educativos de las Universidades. Pero, además, la TI está presente en todas las funciones de la organización. Este uso permanente incrementa la necesidad de profesionales de SI con experiencia en el desarrollo de y gestión de SI

2.1. El estado actual internacional

Las primeras experiencias académicas comenzaron en los años 60, y han crecido en importancia a medida que se ha extendido el uso de las TI como soporte de la decisión para las empresas. De la misma forma que las Universidades comenzaron a ofertar titulaciones sobre distintas funciones de las organizaciones, como la de gestión de recursos humanos,

financieros o de marketing, en EEUU apareció la titulación de Gestión de recursos de TI, o simplemente SI.

El campo de los SI cubre dos áreas:

- La función de SI: tiene la responsabilidad de desarrollar, implementar y gestionar una infraestructura de Tecnología de Información (ordenadores, comunicaciones), datos y sistemas horizontales de la organización. Se incluye también la labor de seguimiento de la nueva TI y la de asistencia en su incorporación en la planificación, estrategias y prácticas de la organización.
- La función de desarrollo de sistemas para los procesos internos y externos de la organización. Implica el uso creativo de las TI para la adquisición de datos, comunicación, coordinación, análisis y soporte a la decisión. La creación de sistemas de este tipo incluye aspectos de gestión de cambio, innovación, calidad, entre otros.

Hay, pues, una muy fuerte relación entre la Informática y los SI. De hecho suelen compartir muchos cursos en común. Sólo se diferencian en que el contexto del trabajo que se desarrolla, los tipos de problemas que se resuelven los tipos de sistemas que se desarrollan y gestionan y el tipo de tecnología que se emplea. Por lo tanto, independientemente de las áreas de estudio, ambas disciplinas tienen un subconjunto de conocimiento técnico.

Esto se puede constatar si se contempla la última referencia de ACM respecto de un modelo curricular para SI, IS'97 [7]. IS´97 cubre las áreas:

- Fundamentos de SI: se introduce al usuario en el uso de los SI y TI en las organizaciones, y la forma en la que éstas proporcionan valor añadido como parte de productos y servicios mejorados, en el soporte a la toma de decisiones o como elementos en los procesos de la organización.
- Teoría y Práctica de SI: Conceptos y teorías de SI, TI que justifiquen métodos y prácticas en el uso de SI que mejoran el rendimiento de la organización. Se estudia la relación con la planificación y estrategia de la empresa.
- TI: Aspectos técnicos de la disciplina, como arquitecturas, Sistemas Operativos, interconexión a través de telecomunicaciones.
- SI: Se analizan problemas y se diseñan e implementan SI. Incluye reingeniería de procesos.
- Desarrollo: Se construye el diseño físico del SI y se implementa usando herramientas de DBMS.
- Desarrollo de SI y Gestión: Gestión del SI, de proyecto (para asegurar la calidad de sus componentes) e integración de sistemas.

IS01.1 Fundamentos de SI	Introducción a los sistemas y conceptos de desarrollo, TI, SI. Se explica cómo se usa la información y cómo la TI mejora la ventaja competitiva.
IS01.2 Estrategia, Arq. y Diseño de e-Business	Se examina la relación de la estrategia organizativa y métodos electrónicos de entrega de productos, servicios entre organizaciones.
IS01.3 Teoría y Práctica de SI	Se da una visión de los modos organizativos, planificación y cómo se usa la información para tomar decisiones.
IS01.4 Hardware y Software de TI	Se proporciona fundamentos de hardware/software para diversas arquitecturas en red/ordenador que se usan en el diseño, e implementación de SI.
IS01.5 Programación, Datos, Objetos	Se expone desarrollo de algoritmos, programación, diseño y aplicación de datos y estructuras de ficheros.
IS01.6 Redes y Telecomunicación	Este curso se estudian las tecnologías en telecomunicaciones y redes
IS01.7 Análisis y Diseño Lógico	Se examina el proceso de desarrollo y modificación del sistema, haciendo énfasis a la comunicación con usuarios
IS01.8 Diseño e Implementac. con DBMS	Se implementa el diseño lógico usando un software para DBMS.
IS01.9 Diseño e Implementac. en Entornos	Cubre el diseño físico e implementación de aplicaciones de SI en entornos de distribuidos.
IS01.10 Gestión del Proyecto	Aspectos técnicos y de comportamiento en la gestión de sistemas en el nivel de empresas.

Figura 3. Alcance de asignaturas propuestas en el IS´01

Este modelo de currículo, de hace 5 años, está en proceso de actualización. Aún no ha aparecido la

versión definitiva del 2002. Pero las presentaciones previas de los trabajos que se están realizando indican que se mantienen aceptables las áreas presentadas del modelo IS'97 pero añadiendo un único curso que dé soporte al crecimiento del comercio electrónico.
En resumen se proponen 10 cursos que permiten presentar las áreas anteriormente identificadas. En la figura 3 se recogen los alcances de los 10 cursos.

2.3. El estado en la FI-UPM

Hay fundamentalmente dos materias en el Plan de Estudios de la FI-UPM, que aúnan la visión de la organización con la de las TI y SI. Son, "La Función Informática en la Empresa" y "Auditoría Informática". Ambas, de 4,5 créditos, son optativas de 5º curso y se impartieron el curso pasado por primera vez en unas condiciones peculiares que fueron expuestas en las anteriores jenui2001 [8]. Básicamente a partir de la experiencia y criterios personales de los profesores encargados que subsanó provisionalmente la carencia de planificación para la elaboración del Plan de Estudios ya comentada. Tras un curso de experiencia docente en estas materias, éste es el momento de mejorar y actualizar los contenidos propuestos inicialmente para dichas asignaturas pero ahora asegurando la coherencia y coordinando los contenidos con otras asignaturas del Plan de Estudios vigente. Es decir, mientras que en un primer momento los esfuerzos se centraron en proporcionar una docencia coherente y cohesionada desde el punto de vista interno de las asignaturas, ahora, en una segunda fase se pretende asegurar la coherencia externa.

3. La calidad de los programas docentes de asignaturas optativas

Para determinar si el programa docente de una asignatura es adecuado no solamente hay que estudiar los contenidos que incluye. Otros aspectos son la cobertura y el grado de profundidad del Plan, así como las habilidades que tienen que reunir los graduados.
En esta ponencia se trata más en detalle uno de estos aspectos dada la situación explicada para las materias con visión de organización en la FI-UPM: el grado de integración y cobertura con respecto de dos elementos de referencia: el modelo curricular de ACM y el Plan de estudios del centro.
Mientras que el primero es genérico a cualquier contexto y se puede utilizar a modo de estándar, el segundo es específico al caso de que se trate, pues depende de, entre otras cosas, de su propio entorno social e industrial.

3.1. Asignaturas de SI en el Plan de Estudios de la FI-UPM

El objetivo en este punto es mostrar cómo cubre el actual Plan de Estudios de nuestro centro las materias del modelo de currículo de SI descrito en el capítulo anterior, y que se ha considerado necesario para atender más adecuadamente a las necesidades de nuestro mercado laboral, así como ser más competitivos frente otras titulaciones superiores no específicamente de Informática.
Los cursos propuestos del modelo IS01 han sido completados con elementos del Cuerpo de conocimientos de SI. Este conjunto de conocimientos está organizado en tres áreas distintas: TI, Conceptos de gestión y de organización y, por último, Teoría y Desarrollo de Sistemas. Cada una de estas áreas se descompone en tres niveles de subapartados que desarrollan los contenidos concretos por cada una de las áreas, a la vez que se encuentra referencias a los cursos del modelo de currículo en donde se incluyen, así como los niveles de profundidad de su aplicación según la taxonomía de Bloom [9].
Basado en este cuerpo de conocimientos se ha realizado un estudio de cobertura utilizando las materias de nuestro Plan de Estudios. El objetivo ha sido identificar lagunas de contenidos que no son recogidas por materias de nuestro Plan de Estudios. Para este estudio sólo se tendrán en cuenta materias troncales, obligatorias y optativas. No se ha tenido en cuenta si los contenidos del cuerpo de conocimientos en su nivel máximo de detalle aparecen reflejados en los temarios de las asignaturas. Esto no tendría sentido dado el alto grado de actualización de contenidos. Por ello el estudio se limita a comprobar la adecuación de las materias del P96 hasta el segundo nivel de descomposición de las áreas de conocimiento

descritos. A continuación se presentan los resultados por áreas:

- Área de TI (IS´97): Este área cubre los apartados recogidos en la figura 4.

Conocimientos IS´97	Correspondencia con materias P96
Arquitecturas de ordenadores	Estructura de computadores Lab. Estructura de Computadores Arquitectura de Computadores Arquitectura de Multiprocesadores Diseño de sistemas Digitales
Algoritmos y Estructuras de Datos	Metodología de Programación Estructura de datos I Estructura de datos II Desarrollo sistemático programas Informática Teórica IA Conexionista. Redes neuronas
Lenguajes de Programación	Metodología Programación Programación concurrente Modelos desarrollo progamas Entornos de programación
Sistemas Operativos	Sistemas Operativos Diseño de Sistemas Operativos Sistemas Operativos distribuidos
Telecomunicaciones	Redes Computadoras Arq. Redes comunicaciones Criptografía Redes datos de Banda ancha Ing. Protocolo comunicaciones Diseño, planif. Y gestión sists. Comunicaciones de datos Sists. Distribuidos: Arq. Comunics
Bases de Datos	Bases de Datos Protección Información BD deductivas BD distribuidas BD Orientadas a Objetos
Int. Artificial	Int. Artificial Ing. Conocimieto Leng. Natural Aprendizaje Automático Validación SBC Modelos razonamiento IA conexionista

Figura 4. Correspondencia entre los conocimientos del área de Tecnología de Información y el P96

- Área de Teoría organizacional (IS´97): Este área cubre los apartados descritos en la figura 5.

Conocimientos IS´97	Correspondencia con materias P96
Teoría Organizacional	Función Informática en la empresa Admón. y organización empresas
Gestión de SI	Función Informática en la Empresa Evaluación de SI Auditoría Protección de la Información LAGUNAS!!
Teoría de la decisión	Sistemas de ayuda a la decisión
Comportamiento organizacional	LAGUNAS!!
Gestión proceso de cambio	LAGUNAS!!
Aspectos legales y éticos	Auditoría Protección de la información
Profesionalismo	LAGUNAS!!
Habilidades personales, Inter-personal	LAGUNAS!!

Figura 5. Correspondencia entre los conocimientos del área de Teoría organizacional y el P96

- Área de Teoría y Desarrollo de Sistemas (IS´97): Este área cubre los apartados de la figura 6.

Es decir, de las tres áreas estudiadas, en una de ellas (la de Teoría Organizacional) se han encontrado determinados bloques de contenidos que no se tratan actualmente en ninguna de las materias del actual Plan de Estudios. Estas lagunas de conocimiento, por su naturaleza, son candidatas a ser incorporadas en la próxima actualización de las asignaturas "Auditoría Informática" y "Función Informática en la Empresa".

7. Conclusiones

La experiencia que aquí se presenta ha permitido la detección de una carencia de unos determinados bloques de contenidos en los programas de dos asignaturas que forman parte de la disciplina de Sistemas de Información. El proceso que se ha seguido está particularizado al contexto proporcionado por el Plan de Estudios vigente en

el centro, su documentación y planificación, y por último, por el conjunto de asignaturas que han querido ser actualizadas mediante esta aproximación.

Conocimientos IS´97	Correspondencia con materias P96
Conceptos de Información y Sistemas	Función Informática en la empresa Ingeniería Software II
Aprox. al desarrollo de Sistemas	Modos de desarrollo de programas Ingeniería Software I Ingeniería Software II Proy. práctico construcción softw. Profundización en Ing. Softw.
Conceptos y metodologías de desarrollo	Modelos desarrollo de programas Ingeniería Software II Proy. práctico construcción softw.
Herramientas y Técnicas	Entornos de programación Proy. práctico construcción softw.
Planificación de aplicación	Sistemas Informáticos Ingeniería Software I Profundización en Ing. Software
Gestión de riesgos	Auditoría Informática Ingeniería Software I
Gestión de Proyectos	Ingeniería Software I Profundización en Ing. Softw.
Análisis de información y negocio	Función Informática en la Empresa
Diseño de SI	Modos de desarrollo de programas Ingeniería Software II Proy. práctico construcción softw. Profundización en Ing. Softw.
Implementación y estrategias de Testing	Evaluación de SI Ingeniería Software II
Operación y mantenimiento de sistemas	Profundización en Ing. Software
Desarrollo de tipos específicos de SI	Sistemas Informáticos

Figura 6. Correspondencia entre los conocimientos del área de Teoría y Desarrollo de Sistemas y el P96

Sin embargo, se pueden derivar distintas lecciones:

- En las titulaciones de Ingeniero en Informática en España hay una generalizada carencia de formación en muchos aspectos de Sistemas de Información.
- Esta formación específica es importante dada la demanda del mercado, no sólo en ámbito internacional.
- Las actualizaciones de programas de asignaturas deben ser coherentes con el Plan de estudios de la titulación.

Referencias

[1] The Joint Task Force on Computing Curricula. *Computing Curricula'91,* ACM and the Computer Society of the IEEE, 1991.

[2] The Joint Task Force on Computing Curricula. *Computing Curricula 2001, Final Report.* ACM and the Computer Society of IEEE, December 15, 2001.

[3] Comité Evaluación Interna, *Informe de Evaluación Interna de la FI-UPM*, Informe Técnico FIM/113/Decanato/2001, Noviembre de 2001

[4] Gabinete de Estudios Sociológicos y Estadística de la UPM. *Estudio sobre el empleo de los graduados de la UPM*, Rectorado de la Universidad Politécnica de Madrid, 2000.

[5] Mawhinney, Morrel and Morris. *The IS Curriculum: Closing the gap*, ISECON´94 Proceedings, 1994.

[6] Trauth, Frawell and Lee. *The IS expectation Gap: Industry expectations versus academic preparation* MIS Quaterly, 1993.

[7] ACM, AIS, AITP, *IS'97 Model Curriculum Guidelines for Undergraduate Degree Programs in Information Systems*, Association of Information Technology Professionals 1997.

[8] Tovar, E. y otros. La Función Informática en la Empresa, VII Jornadas de Enseñanza Universitaria de la Informática jenui 2001, Palma de Mallorca, 2001.

[9] Bloom, B. *The taxonomy of educational objectives: Classification of educational goals. Handbook1: the cognitive domain*. New York: McKay Press, 1956.

Perfil profesional y académico de la informática en España

Gloria Martínez, Germán Fabregat
Dpto. de Ingeniería y Ciencia de los Computadores
Universidad Jaume I
12071 Castellón
{martine,fabregat}@icc.uji.es

Resumen

La situación actual del mercado laboral para los titulados informáticos es halagüeña, como consecuencia del contexto socio–económico. Hay una gran demanda laboral en todo el mundo en el sector de las Tecnologías de la Información, TI, también conocido en España como "nuevas tecnologías".

Esta gran demanda puede llegar a afectar a la definición del perfil académico. En este punto, históricamente, la visión académica siempre difiere de la empresarial, y viceversa. En esta discusión debería incorporarse, de forma positiva, otro factor: tanto en EEUU como en Europa, se está promoviendo un amplio debate sobre cuáles son los perfiles y habilidades necesarias para cada uno de los profesionales de las TI.

En esta definición de los perfiles profesionales, la universidad no debería permanecer al margen. Además de servir como un acercamiento a la visión empresarial, puede ser un motivo de reflexión sobre nuestra troncalidad intelectual, sobre la idoneidad de las titulaciones actuales y nuestro propio perfil científico, actualmente vertebrado en tres áreas de conocimiento.

1. Contexto socio–económico

Durante el último lustro las Tecnologías de la Información y de las Comunicaciones, TIC, se han convertido en una de las áreas económicas de mayor crecimiento; de hecho, diversos estudios de 2000 sugieren que el rápido crecimiento de la economía estadounidense de 1995 a 2000 ha sido liderado por los avances en el sector de la Tecnología de la Información, TI, y la aplicación de estas tecnologías a otras áreas de la economía. Los primeros datos sobre el año 2001 apuntan que sigue este crecimiento, si bien menos acelerado, en parte por la situación económica (que ha propiciado la quiebra de empresas tecnológicas surgidas en plena expansión), en parte por la situación especial que vive la economía norteamericana tras el atentado del 11 de Septiembre de 2001 ([4] [2]).

El uso de las tecnologías está asociado a nuevos patrones de creación y pérdida de trabajos: las TIC reemplazan viejas tareas, mediante la automatización, pero crean nuevos puestos para ocupaciones con demandas crecientes, como la de programación *software*. De hecho, en Europa y en EEUU, la demanda de profesionales ha superado a la oferta y se ha tenido que suplir la carencia con la incorpo-

ración de profesionales extranjeros. Es famoso el caso de la oferta alemana de incorporar a 200.000 profesionales informáticos de élite extranjeros [7].

La situación es similar (o peor) en España. Si bien en los últimos años la economía española se ha acercado a la media europea, y el sector de las TI ha conocido un crecimiento sostenido en los últimos 6 años de más de un 80 %, los datos indican que el mercado español de las TI aún está por debajo de la media europea y que, además, en nuestro mercado, las importaciones superan a las exportaciones [15].

La demanda de profesionales informáticos es muy alta; se estima [8] que el déficit de expertos en TI será de 100.000 personas en 2003, de acuerdo a la evolución de la demanda. Aun cuando los datos de estimación puedan ser demasiado optimistas, la situación real es que, en el estado, no existen suficientes titulados para hacer frente a la demanda; y, lo que es peor, dada la pirámide de población y el previsible descenso de alumnos matriculados en la universidad para los próximos cursos, no parece que ésta se pueda cubrir.

Esta situación ha provocado la toma de medidas, algunas detalladas en el plan *InfoXXI* del Ministerio de Ciencia y Tecnología [13], como las tendentes a promover la "alfabetización informática" de la población y la formación de profesionales (por cada titulado superior, se estima que se necesitan tres personas con conocimientos básicos de tecnologías de la información) y a impulsar el uso de telecomunicaciones y nuevas tecnologías en las *pymes*.

Lo que, en principio, puede ser una situación laboral beneficiosa para nuestros estudiantes, tiene una vertiente peligrosa que ha de preocuparnos como docentes: la creciente demanda por las empresas de titulados informáticos, puede provocar una tendencia a rehacer los *curricula* universitarios en función de la demanda empresarial, en perjuicio de otros factores.

2. El mercado laboral y la visión empresarial

En [11] se puede encontrar un amplio estudio del mercado laboral en España. De este estudio se desprende que éste se concentra, principalmente, en Madrid y Cataluña, y el desglose por sectores muestra que Finanzas, Administración Pública e Industria son, por este orden, los que cuentan con mayor peso relativo.

En cuanto a la evolución anual del crecimiento de empleo, tras un bajón en los años 92–95, sufrió una espectacular aceleración en el 98 (crecimiento de un 25 %), disminuyó ligeramente en el 99 y las previsiones son que se mantendrá cercano a un 10 % [1] (según los últimos estudios de la SEDISI[1], [16], la evolución interanual de Octubre de 2000 a Septiembre de 2001, ha sido de 8,8 %. Además, los últimos informes del Centro de Predicciones, [2], estiman que el crecimiento para 2002, sobre la cifra de este año, será de aproximadamente un 1 %). Los datos sobre el período 98/99/00, también indican que el 83,67 % de los trabajadores en empresas informáticas tienen un contrato fijo y que la mayoría de las ofertas de trabajo fijo suelen dirigirse a titulados. Sólo un 32 % de la ofertas no exigían titulación y se correspondían mayoritariamente con contratos temporales, de menor salario y poca responsabilidad.

También en [11] se puede encontrar un análisis de ofertas laborales en un diario de ámbito nacional[2]. Primero, se observa que hay una gran demanda de profesionales con perfiles específicos (el 70 %). De entre estos perfiles, por solicitudes, destacan bases de datos

[1] SEDISI, Asociación española de empresas de tecnologías de la información

[2] El período de estudio comprende desde Diciembre de 2000 a Abril de 2001.

(25 %), desarrollo de software (20 %), aplicaciones web (14,4 %), seguridad/sistemas operativos (13,2 %) y redes (12,1 %). Del análisis también se desprende que, de las ofertas que exigían una titulación, el 25 % era para titulados superiores en informática, el 18 % para titulados técnicos en informática (aunque con un fuerte solape con la titulación superior) y sólo el 3 % para titulados en Formación Profesional. Según el propio diario, en 2000 el número de ofertas publicadas para Ingenieros en Informática fue de 9.668 y para Ingenieros Técnicos en Informática, de 7.293.

Estos datos, junto con la alta demanda que se deriva de la exposición de la situación económica, muestran un futuro laboral prometedor para nuestros titulados. Cuestión aparte es cómo desean las empresas que formemos a los alumnos y cuáles son las destrezas que buscan: es el dilema perfil profesional/perfil académico que, históricamente, enfrenta al empresario con el docente. Y este enfrentamiento se puede ver agravado por la necesidad acuciante de profesionales y podría llegar a influir, de forma perniciosa, en los planes de estudio de las universidades.

La situación socio–económica en el ámbito de las TI, se traduce en una falta de personal cualificado, lo que provoca que aumenten los salarios, lo que a su vez desemboca en que muchas *pymes* no puedan competir lo que, a la larga, perjudica al desarrollo económico. Esto está llevando a que, cada vez más, se oigan propuestas en el sentido de que las universidades evolucionen en una especie de centros de formación profesional, con mayor capacidad de evolución. Esta visión, que difícilmente será compartida por la universidad pública, sí puede ser utilizada por la privada y aumentar así las diferencias entre ambas. Si esto se combina en el marco de la nueva nueva ley de ordenación de las universidades, las consecuencias pueden ser nefastas para la educación universitaria pública.

Es de desear que esta posibilidad no se dé. Hay varios factores que, combinados, pueden permitir el desarrollo armónico de los intereses universitarios, en cuanto a garantizar profesionales verdaderamente cualificados, y los empresariales. En primer lugar, el acercamiento universidad-empresa. La universidad debe reconocer la importancia de las prácticas dentro del desarrollo curricular de los estudios tecnológicos, pero también debe hacer partícipe a la empresa de las ventajas prácticas que suponen, a la larga, la contratación de profesionales con una amplia base fundamental y con capacidad de adaptación, en un entorno en que los avances tecnológicos son tan rápidos.

Pero, quizás, la discusión que es preciso incorporar de forma prioritaria en este acercamiento universidad-empresa es la definición de las habilidades propias de un profesional en informática, en sus distintos niveles de cualificación: no es eficiente formar un ingeniero si después el desconocimiento de cuáles son sus habilidades reales, se traduce en una contratación para un cargo que no requiere ese nivel de formación. El desconocimiento sobre nuestra titulación hace que, muchas veces, se confunda un técnico con un ingeniero. Utilizando un símil, nadie acude a un ingeniero industrial especializado en máquinas y motores térmicos cuando su coche se estropea; va al mecánico. Pero, en cambio, muchas veces se recurre a un ingeniero informático para cambiar la configuración de una impresora. Hay que definir bien la profesión para que el empresario sepa cómo y en qué campos actúan los diferentes profesionales de la informática.

3. ¿Quiénes somos?

Este es el título de un artículo de Peter Denning [6] (quien, entre otros cargos, ostenta el de Presidente del Consejo de Educación de ACM). En él, se hace un análisis crítico de las características que hacen de una actividad

una profesión, además de considerar los factores que justifican la "profesión en tecnología de la información". También, hace una clasificación de las actividades relacionadas con el ejercicio en el campo de las TI y las TIC, ya que no todos los puestos de trabajo están directamente relacionados con la informática, o deben ser desempeñados por informáticos. El mismo autor, en una entrevista publicada en la revista *Ubiquity* de ACM [5], expone las ideas básicas que han dado lugar a que exista esa inquietud por definir convenientemente dicho campo profesional.

Destaca, entre otras consideraciones, la constatación de que existen muchos grupos profesionales propios dentro de las TI, y que la informática se ha convertido en uno más de esos grupos. Si cada grupo tiende a seguir su propio camino y no cooperar, puede haber lugar a roces y enfrentamientos. De ahí, la importancia de una buena definición. Para conseguir esto, por ejemplo, ACM ha puesto en marcha la iniciativa ITP, *Information Technology Profession*, cuya finalidad es acabar estableciendo la Tecnología de la Información como una profesión. Hay ya tres proyectos en marcha, el proyecto *Ubiquity* nacido como un foro de debate en el que se pretende involucrar al mayor número posible de profesionales en la definición de la profesión, y la promoción del ICDL, *International Computer Driving License*, como acreditación internacional similar al ECDL, *European Computer Driving License*. El tercero es el más ambicioso, el *proyecto de identidad ITP*, cuya misión es definir la estructura del campo de las TI, incluyendo troncalidad profesional e intelectual, estándares de cualificación, instituciones propias y grupos profesionales.

En Europa también se están realizando actividades en este sentido. En 1999 se formó el ICT (*Information and Communication Society*) Consortium, para fomentar las actividades que la falta de profesionales de las TI podría estar frenando. Este consorcio está formado por distintas empresas y entre sus actividades, hasta el momento, destaca la recomendación de acciones específicas, orientadas a promover el uso de las nuevas tecnologías en el ámbito diario, profesional y educativo. Destaca también la serie de perfiles profesionales que ha elaborado, a los que incorpora no sólo habilidades técnicas, sino también de comportamiento. Además ha promovido *Career-space* [12], un portal de Internet que proporciona los perfiles profesionales más demandados y permite que los candidatos conecten con trabajadores en ejercicio que puedan asesorarles desde su experiencia. Otro organismo que está desarrollando una amplia labor en la definición de la profesión, es el CEPIS, *Council of European Professional Informatics Societies*, que es el promotor del ya mencionado ECDL, que ha tenido un gran éxito en Europa. Se trata de una acreditación de habilidades laborales básicas en ofimática, que incluye manejo de procesador de textos, hojas de cálculo, bases de datos e Internet. Es de destacar que esta acreditación no está dirigida a profesionales de las TI, sino a los que usan las TI en su trabajo.

En España, uno de los proyectos más importantes en este sentido ha estado promovido por el ANIEL (Consejo de la Asociación Nacional de Industrias Electrónicas y de Telecomunicaciones), el Colegio Oficial de Ingenieros de Telecomunicación y la Universidad Politécnica de Madrid. Es el Proyecto PAFET, Propuesta de Acciones para la Formación de Profesionales de Electrónica, Informática y Telecomunicaciones [14]. A partir del diagnóstico de la situación actual, el informe identifica perfiles profesionales y establece un conjunto de recomendaciones, a todos los implicados, para el período 2001–2003. Además, los promotores han decidido la creación de un Observatorio Permanente sobre las necesidades del sector.

4. La importancia del perfil profesional

En este contexto internacional de definición de la profesión en tecnologías de la información, en España se está produciendo la creación de colegios profesionales en diferentes comunidades autónomas, en la creencia de que ello ayudará a la definición y consolidación de la profesión. Se ha aprobado la creación de colegios en Murcia (28 de abril 1998), Valencia (19 de Mayo 2000), País Vasco (29 de Junio 2000), Cataluña (9 de Abril 2001) y Asturias (21 de Mayo 2001). En Aragón el proyecto de creación fue rechazado por las cortes aragonesas en Mayo de 2001. El proceso legislativo se encuentra en marcha en Galicia y Castilla-León. Hay que recordar, también, que el Partido Popular intentó tramitar una ley nacional cuya tramitación fue interrumpida por la convocatoria a elecciones legislativas de Marzo de 2000 [9].

Las repercusiones que la creación de los colegios pueden tener en el progreso de la profesión en España, no están muy claras. En principio, se puede calificar de positiva cualquier iniciativa que contribuya a que los profesionales informáticos se agrupen para mejorar su profesión, su formación, su código ético y su influencia social; pero, al analizar cada una de las leyes de creación y los estatutos que las desarrollan se ve que dos cuestiones básicas son tratadas de diferente forma por los diferentes colegios: los más estrictos, los de Murcia y Valencia, exigen estar colegiado para poder ejercer y sólo permiten la colegiación a profesionales de la titulación (en esta última cuestión, sólo el colegio de Cataluña contempla la posibilidad de permitir la colegiación a profesionales informáticos provenientes de otras titulaciones).

¿Puede mejorar la situación profesional la aplicación estricta de las leyes colegiales? No parece claro; incluso, algunos expertos señalan que tales normas podrían ser anticonstitucionales. Lo que parece un contrasentido es que, cuando el principal problema del sector es la falta de profesionales cualificados, se intente legislar de forma que se corra el riesgo de provocar inflexibilidad, falta de reciclaje o, incluso, paralizar un sector profesional (el intrusismo, que ya no es un problema tan grave, es consecuencia en gran medida, no lo olvidemos, del hecho de estar hablando de una titulación muy reciente), yendo, además, en contra del libre movimiento de los trabajadores europeos y de las recomendaciones de los organismos europeos antes mencionados (una de cuya acciones recomendadas es propiciar la movilidad de trabajadores cualificados entre estados, miembros o no de la UE, para favorecer el aprovechamiento de los recursos profesionales existentes).

Se da, además, una paradoja que vuelve a incidir en la importancia de trabajar prioritariamente en la definición del perfil profesional [10]: una sentencia del Tribunal Constitucional, de 11 de Mayo de 1989, advierte de que "*el legislador al hacer uso de la habilitación [..] deberá hacerlo de forma tal que restrinja lo menos posible, y de modo justificado [..] y que al decidir, en cada caso concreto, la creación de un Colegio Profesional, en cuanto tal, haya de tener en cuenta que, al afectar la existencia de éste a [..] derechos fundamentales [..] sólo será constitucionalmente lícita cuando esté justificada por la necesidad de servir un interés público*".

En el caso de los Colegios Profesionales de Informática, se requerirá, por tanto, demostrar el interés público al que sirve su creación; los fines específicos de interés público quedan determinados por la propia *profesión* titulada cuya regulación se pretende. Es decir, la delimitación de la profesión informática ha de ser previa al intento de demostrar el interés público. Y aún estamos discutiendo si la Informática es un ciencia, una ingeniería, una técnica o

un sector económico.

El colectivo universitario debe implicarse en este proceso de definición. Como docentes, para procurar un buen ajuste entre el perfil académico y el perfil profesional; y, sobre todo, como científicos. La definición de la troncalidad intelectual no nos debe ser ajena y, por supuesto, no se debe dejar exclusivamente en manos de las empresas, ya que, a la larga, se corre el riesgo de crear profesionales "a la carta" en lugar de buenos profesionales, con una sólida base de conocimientos.

Y es en esta situación en la que también deberíamos ser críticos con nosotros mismos: ¿responde la oferta de titulaciones actual a las necesidades actuales del mercado profesional? ¿responde la ordenación académica a la evolución del mercado profesional y a la evolución científica de la informática?

5. La Enseñanza Universitaria de Informática en España

En el Decreto del Ministerio de Educación y Ciencia 327/1976, de fecha 26 de Febrero de 1976, se estableció el desarrollo, a través de la educación universitaria y de la formación profesional, de las enseñanzas de informática en España, así como la especificación de los diferentes estudios requeridos para cada tipo de trabajo:

- Codificador de Datos, título de Técnico Auxiliar de Informática.
- Operador, título de Técnico Especialista en Informática.
- Analista de Aplicaciones, título de Diplomado en Informática (actualmente Ingeniero Técnico en Informática).
- Técnico de Sistemas, Licenciado en Informática (actualmente Ingeniero en Informática).

La adopción de las denominaciones de *Licenciatura* y *Diplomatura* fue impuesta por las circunstancias políticas. Con la reforma de las directrices para la elaboración de los Planes de Estudio de 1987, se instauró la ordenación de las materias en troncales, obligatorias de universidad, optativas y de libre configuración. También se estableció la nueva organización de los planes de estudio de acuerdo a créditos, de los que cada asignatura tiene asignada una cantidad. La reforma de las titulaciones de 1990 convierte las de Licenciado y Diplomado en Informática en, respectivamente, Ingeniero en Informática e Ingeniero Técnico en Informática de Gestión o de Sistemas.

Pero esa reforma supuso, casi exclusivamente, una reforma de la ordenación académica y no ha propiciado una redefinición de los perfiles laborales de los titulados ni una mayor adecuación a las circunstancias actuales. Los avances en el terreno de las telecomunicaciones y las redes de ordenadores en los últimos años deben propiciar un amplio debate sobre los contenidos de los estudios de informática, replanteando tanto su troncalidad como la posibilidad de que aparezcan nuevas carreras que contemplen especializaciones hasta ahora no consideradas y que se adapten mejor a las demandas del mercado de trabajo. Si en 1976 los estudios de informática quedaban muy limitados con la creación de un único título superior y dos diplomaturas, en la actualidad esta situación, que permanece idéntica, se hace cada vez más injustificable. Es preciso un debate en profundidad sobre la situación profesional de la informática, a fin de obtener un perfil más ajustado de los *curricula* educativos.

Y, en este debate, el colectivo universitario tal vez debería implicarse también en su propia definición intelectual. En el año 1984, con la publicación de la Ley de Reforma Universitaria (LRU) se organiza la docencia de nivel universitario en áreas de conocimiento. Según define el Real Decreto 1888/1984, por el que se

regulan los concursos para la provisión de plazas de los cuerpos docentes universitarios, en su artículo segundo "*se entenderá por área de conocimiento aquellos campos del saber caracterizados por la homogeneidad de su objeto de conocimiento, una común tradición histórica y la existencia de comunidades de investigadores, nacionales o internacionales*".

De acuerdo con esta definición, se constituyeron tres áreas de conocimiento en Informática: el área de Arquitectura y Tecnología de Computadores (ATC), el área de Lenguajes y Sistemas Informáticos (LSI) y el área de Ciencias de la Computación e Inteligencia Artificial (CCIA).

Los contenidos temáticos del área de ATC son relativamente amplios, abarcando todo lo relacionado con el estudio del soporte físico, el *hardware*, su programación y control. En el caso de las áreas de LSI y CCIA, los contenidos son igualmente amplios; ambas están relacionadas con el estudio y el desarrollo de los sistemas lógicos. Pero, no está muy claro qué atribuciones corresponden a cada área: basta con mirar las directrices generales de los planes de estudio de las titulaciones en informática, para observar que el número de materias que pueden asignarse indistintamente a ambas áreas es muy elevado. Es decir, la frontera entre ellas, académicamente hablando, es mínima. Tómese como ejemplo, el colectivo de científicos que desarrollan su trabajo en relación con las Bases de Datos: dependiendo de la universidad en la que desarrollen su trabajo, estarán adscritos al área de CCIA o al área de LSI.

La pregunta es ¿esta situación es deseable? ¿son las áreas únicamente el instrumento administrativo que permite concursar a una plaza o las áreas deberían vertebrar el proceso educativo? Si al discutir sobre las titulaciones se observa una falta de especialización en los titulados, más acorde con el desarrollo actual de la Informática, es posible que no baste con tres áreas de conocimiento de la docencia y la investigación para cubrir esas especializaciones.

Es de prever que, en el futuro, las áreas de informática deberían redefinirse. El desarrollo natural de las áreas debería conducirnos a distinguir, dentro del desarrollo de los sistemas físicos, entre el Estudio y Desarrollo de las Redes, de los Sistemas Operativos, además de la Arquitectura de Computadores y de la Tecnología Electrónica. Y en el desarrollo de los sistemas lógicos, sería más coherente distinguir entre el desarrollo de la Ingeniería del Software, la Gestión de la Información, los Sistemas Inteligentes, la Informática Gráfica, los Fundamentos de la Computación y la Tecnología de la Programación, entre otras áreas.

Si queremos implicarnos en la definición del perfil profesional de nuestros titulados, tal vez deberíamos comenzar debatiendo esta situación, identificando los distintos campos de desarrollo y su ámbito de estudio.

Referencias

1. Centro de Predicción Económica, Informe del Año 2000.
2. Centro de Predicción Económica, Informes Sectoriales.

   <http://www.ceprede.com>
3. "El Dilema Universitario ¿Cómo enseñar Tecnología?", *Ciberpais Mensual*, n. 17. Diciembre 2001.
4. "En tiempos de crisis, compras a precios de saldo". Cinco Días, 8 de Diciembre de 2001.
5. P. J. Denning. "El Futuro de la Profesión de Tecnología de la Información". Entrevista en *Ubiquity* de ACM, 21 de Marzo de 2000. Reproducido por *Novática*, n. 147. Sep/Oct 2000.

6. P. J. Denning. "Who are we?". *Communications of the ACM*, Febrero 2001. Reproducido por *Novática*, n. 152. Jul/Ago 2001.

7. "Un problema de cien mil empleos", El País, sección Economía, 5 de Junio de 2000.

8. "La otra cara de Internet", El País, suplemento Negocios, 7 de Enero de 2001.

9. Rafael Fernández Calvo. "El Déficit de Informáticos y la Regulación Legal del Ejercicio de la Profesión Informática en España", *Novática*, n. 152. Jul/Ago. 2001.

10. Gonzalo Gavín González. "Los Colegios Profesionales de Informáticos: Análisis del Marco Legal", *Novática*, n. 152. Jul/Ago. 2001.

11. José Hernández Orallo. "Contexo Socio-Económico", Capítulo 6, Proyecto Docente "*Bases de Datos*". Escuela Universitaria de Informática, Universidad Politécnica de Valencia. Mayo 2001.

12. Information and Communication Society Consortium,

    <http://www.career-space.com>

13. Ministerio de Ciencia y Tecnología, Plan InfoXXI,

    <http://www.infoxxi.es>

14. Proyecto PAFET,

    <http://www.aniel.es/aniel/proyecto%20pafet.htm>

15. SEDISI, Asociación Española de Empresas de Tecnologías de la Información. Resumen Ejecutivo del Informe "Las Tecnologías de la Información en España, 2000".

16. SEDISI, Asociación Española de Empresas de Tecnologías de la Información. Estadísticas del Mercado Informático 2001,

    <http://www.sedisi.es>

Programación, algoritmos y estructuras de datos

Análisis de las propuestas de la enseñanza de la programación orientada a objetos en los primeros cursos

L. Fernández Muñoz (a), R. Peña(b), F. Nava (c), Á. Velázquez Iturbide(c)
(a) Dept. LPSI. Universidad Politécnica. 28071 Madrid e-mail: setillo@eui.upm.es
(b) Facultad de Documentación. Universidad de Alcalá. 28871 Madrid e-mail: rpr@uah.es
(c) Escuela Superior de Ciencias Experimentales. Universidad Rey Juan Carlos. 28933 Madrid
e-mail: {f.j.nava,a.velazquez}@escet.urjc.es

Resumen

Existe un fuerte debate sobre qué paradigma conviene impartir en el primer curso de los planes de estudio de Informática: orientación a objetos (OO) o procedimental, así como sobre el lenguaje de soporte de las sesiones prácticas. Muchos artículos presentan sus experiencias o proponen nuevas metodologías de forma aislada, pero no encontramos un estudio sistemático de los distintos enfoques. En este trabajo se presenta una visión crítica de las propuestas de diferentes autores, se evalúan sus ventajas e inconvenientes. Se concluye la necesidad de reconducir el debate de la planificación de los primeros cursos de programación, hoy día centrado en el orden en que se enseñan los paradigmas y el lenguaje para implementarlo, hacia la búsqueda de los conceptos fundamentales y herramientas pedagógicas que permitan una exposición gradual de los conceptos de la programación.

1. Introducción

En la década de los noventa, la mayoría de los centros de enseñanza superior de Informática ha ido integrando conceptos de OO en sus planes de estudio, como se refleja en los datos aportados por [11,18,27,37,42,49]. La última propuesta curricular del grupo conjunto de ACM e IEEE [2] incluye la OO como una de las materias claves en las titulaciones universitarias de Informática.

Inicialmente, la programación OO se ubicó en segundo curso o superiores. Actualmente hay un debate abierto sobre cuál es el primer paradigma al que debe enfrentarse al alumno. Unos abogan por presentar antes un paradigma procedimental [4-6,9,10,15,23,26,36,39,45,53,56], basándose en la necesidad de no aumentar el gran número de conceptos que es necesario presentar al alumno que se acerca por primera vez a la programación; mientras otros abogan por la OO como primer paradigma, basándose en su carácter intuitivo, potencia del lenguaje, y la fuerte motivación que provoca en los alumnos [3,7,16,18,21,25,28,33-37,47,48,51,55,58,59,61-63].

Esta discusión es relevante también desde el punto de vista administrativo, ya que la disparidad de criterios en la planificación de los primeros cursos provoca dificultades en la movilidad de los alumnos [11], que se pretende fomentar en Europa [17], resultando importante homogeneizar los enfoques adoptados por los centros.

Este artículo presenta una panorámica de las diferentes propuestas, las clasifica y sistematiza, resaltando sus ventajas e inconvenientes. La figura 1 sintetiza esta clasificación.

La estructura del artículo es la siguiente: el apartado 2 resume las recomendaciones curriculares de ACM/IEEE para la docencia de la programación de los primeros cursos. En los apartados 3 y 4 estudiamos, respectivamente, las propuestas que presentan el paradigma procedimental en primero y el OO en segundo, y las que abordan el paradigma OO desde el principio.

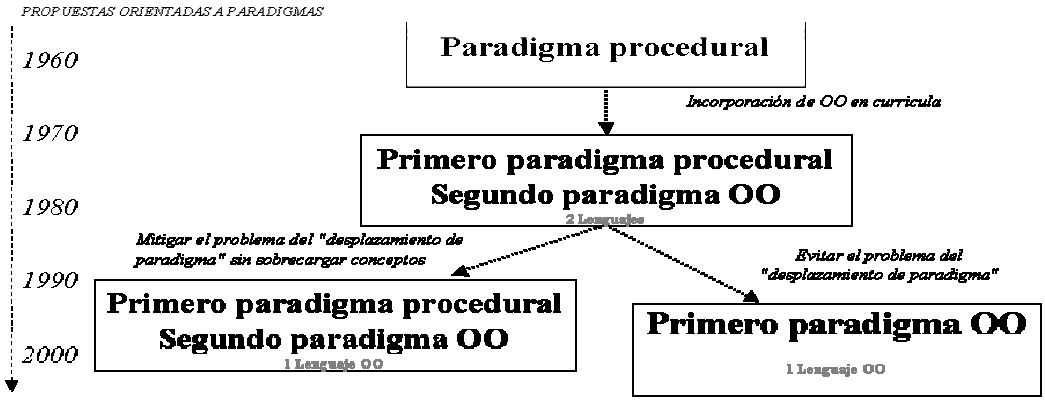


Figura 1 Evolución de los enfoques de la enseñanza de los paradigmas

2. Recomendaciones ACM/IEEE2001

La última propuesta curricular ACM/IEEE avala la seriedad de la polémica acerca del momento adecuado de introducir al alumno en la OO. Reconoce el debate y lo amplía proponiendo, no dos, sino seis diseños alternativos para los cursos de introducción a la programación:

- comenzando con el paradigma procedimental: Incluso usando un lenguaje OO, el primer curso se centra sobre los aspectos imperativos del lenguaje.
- comenzando con el paradigma OO: Empieza directamente con nociones de objetos y herencia.
- comenzando con el paradigma funcional: Se caracteriza por usar un lenguaje funcional sencillo.
- presentación en amplitud: El primer curso, además de atender la programación, algoritmos y estructuras de datos integra otras subdisciplinas, como las matemáticas.
- enfoque algorítmico: Los conceptos básicos se introducen usando pseudocódigo en vez de un lenguaje ejecutable.
- enfoque arquitectónico: Comienza con circuitos, con los que construye una máquina de von Neumann simple. Establecidos los fundamentos del *hardware*, presenta un lenguaje de alto nivel.

Las dos primeras propuestas son las que afectan a nuestra discusión. Las experiencias publicadas que tratan la polémica se pueden clasificar como: procedimental en primero y OO en segundo versus OO en primero.

3. Paradigma procedimental en primero y OO en segundo

Esta línea aborda primero la programación procedimental y, una vez asentados los conceptos de control de flujo de ejecución y de tipos abstractos de datos, se introducen los conceptos de OO.

La OO viene a resolver la complejidad de la gestión del software. No es adecuada para los ejercicios correspondientes a un primer acercamiento al paradigma procedimental. La OO no es una novedad, se emplea desde el año 1967 [16,18,37,40] en áreas de simulación y gráficos. En los sistemas de gestión, comunicaciones, etc., se introdujo 20 años después, como una vía para aumentar la eficiencia en la gestión del software, cuando éstos elevaron su complejidad, antes no tenía justificación.

La enseñanza de OO en segundo curso permite abordar problemas más complejos, en los que se pueda tomar mayor ventaja, y por tanto presentar mejor sus principales aportaciones: (reutilización de código, jerarquización, encapsulación y facilidad de la gestión del *software*).

Las ventajas de este enfoque son la exposición escalonada y paulatina de los conceptos de la programación. Pero muchas

experiencias relatan la dificultad que presentan los alumnos al cambiar de paradigma, lo que ha venido denominándose el "problema del desplazamiento" [18,40]. Otros, consideran como desventaja de esta ordenación temporal el retraso en el desarrollo de aplicaciones motivadoras.

Este enfoque se articula de dos formas distintas, que evaluamos a continuación.

3.1. Dos lenguajes para dos paradigmas

Estas propuestas defienden la exposición tradicional [15,23,26]: primero programación estructurada con un lenguaje y luego programación OO con otro lenguaje.

La ventaja es que los conceptos se pueden introducir de forma gradual e independiente, pero el problema del "desplazamiento de paradigma" se muestra en toda su amplitud.

El cambio sintáctico/semántico entre ambos lenguajes, distrae al profesor en la exposición y al alumno en el aprendizaje de nuevos conceptos que se apoyan en otros asumidos, pero presentados con una nueva cara. Hay que parar constantemente para explicar unos nuevos tipos primitivos, una nueva tabla de precedencias de operadores, parecidas estructuras de control de flujo de ejecución, etc.

3.2. Un lenguaje OO para los dos paradigmas

Para mitigar el problema del desplazamiento, esta línea comienza con el paradigma convencional, empleando un lenguaje OO [39,48,53].

La ventaja de usar el mismo lenguaje para las dos partes es mantener una uniformidad y evitar la exposición de los aspectos léxicos, sintácticos y semánticos de dos lenguajes diferentes.

Sin embargo, presenta graves inconvenientes: el tratamiento de tipos, variables, expresiones y sentencias tiene que ubicarse en algún sitio; sin duda en una clase. Es imprescindible empezar con ejemplos sencillos (quizás la gestión de un contador, o dos alternativas anidadas para determinar el tipo de triángulo formado por tres vértices fijados) ¿cómo se llaman las clases?, ¿dónde están los objetos?, ¿quién lanza los mensajes? La resolución de este tipo de ejercicios desvirtúa conceptos muy complejos que no tienen cabida en los ejemplos iniciales, necesariamente sencillos.

Emplear un lenguaje que no está orientado al paradigma explicado genera confusión y hábitos inadecuados en el alumno. Algunos autores advierten a sus alumnos que "esto, realmente, no se resuelve como lo estamos haciendo" [48]. Nos parece una pauta pedagógica inadecuada.

El otro grave inconveniente radica en cuestiones psicológicas y sociológicas. El cambio de las palabras reservadas que organizan los módulos del software no perjudica, sino que facilita la transición de la programación estructurada a la OO. Si se desea cambiar la percepción social, por ejemplo, sobre el oficio de "barrendero", es adecuado cambiar su denominación, por ejemplo a "especialista en sanidad urbana".

Merecen una atención especial las propuestas docentes que emplean lenguajes híbridos (Ada/Ada9x, C/C++ o Pascal/Delphi) que podrían mitigar el impacto en la transición [3,39,59]. Nuestra experiencia demuestra que generan confusión en el alumno a la hora de ubicarse en uno u otro paradigma ya que el programa "compila" y "funciona" aunque no se respete el diseño adecuado al paradigma. En [26,29,31,34] se concluye que "el uso de un lenguaje OO puro para enseñar los conceptos OO es mejor e involucra menos conceptos que el empleo de un lenguaje híbrido". En lo tocante a procedimientos, declaración de variables y estructuras de control no hay inconveniente; por el contrario, beneficia, mantener la sintaxis.

En resumen, consideramos adecuado abordar los conceptos de OO cuando están consolidados los correspondientes al paradigma procedimental y abordar la docencia de cada paradigma con un lenguaje que se adecue a los conceptos que se están trasmitiendo.

4. Objetos en primero

La motivación general de estas propuestas viene de la mano del carácter "intuitivo" [49] y la facilidad del alumno para descubrir las clases del dominio [47].

Sorprendente la importancia dada al carácter intuitivo de la OO. Esto no es nuevo. Cada

lenguaje surge para hacer más intuitiva la expresión de determinados mecanismos mentales.

Negroponte afirma que "los sistemas informáticos son el mayor reto de complejidad al que se ha enfrentado el hombre en la historia" [46]. Esta aseveración resulta paradójica con la intuitiva sencillez prometida.

La falacia de la paradoja es sencilla: hablan de cosas diferentes. Una cosa es aprender a manejar un mundo de objetos simulados y otra es diseñar y programar una aplicación de calidad a través del enfoque OO. Son tareas de muy diferente índole, a realizar por un usuario y un ingeniero, respectivamente.

Meyer apunta [43]: "hay que huir de dar mucha importancia a la idea de que los sistemas OO se deducen directamente del *mundo real*... Aunque una vez que se entienden pueden parecer tan reales como cualquier otro, para un recién llegado pueden parecer menos *naturales* que los conceptos utilizados en las soluciones orientadas a procesos".

El programa OO resultante no es tan intuitivo como se pretende. Pensemos en la clase *Fecha* con su método *esNavidad*, o *Tablero* con el método *ponerFicha* en una *Coordenada*... ¿desde cuando una fecha informa sí es el día de Navidad?, ¿es "intuitivo" un tablero con manos que coloca fichas donde le dice un objeto coordenada?, ¿coordenada es un objeto? En "el mundo real" las acciones (*esNavidad* o *ponerFicha*), las controla el hombre, no los objetos (de la clase *Fecha* o *Tablero*).

Cuando se hace referencia carácter "cercano al pensamiento cotidiano" de la OO, se está hablando de localizar entes y asociarlos con sus operaciones plausibles, advirtiendo, por ejemplo que no se botan vasos, ni se beben pelotas. De forma similar, en OO se determina qué operaciones tienen cabida para cierto conjunto de datos y, para organizar el software, se usan los mecanismos habituales para estructurar el conocimiento: los modelos, la abstracción, los enlaces, la ordenación en el tiempo/espacio... Solo en este sentido es intuitiva.

En la programación OO hay que encontrar un modelo eficiente, tanto en ejecución e implementación, como en la gestión y posterior mantenimiento del software, que puede no corresponder a la visión informal del mundo.

"Es útil remarcar que los primeros libros sobre OO enfatizaban lo fácil que era identificar objetos mientras que los últimos, a menudo de los mismos autores, enfatizan su dificultad" [19]. "Experiencias con principiantes en OO muestran que tienen dificultades en crear clases adecuadas" [14,19].

En conclusión, un principiante no está en mejor disposición para aprender el paradigma OO que otro paradigma [60].

Existen diversas concreciones para el enfoque de OO en primero, que hemos clasificado como: posponiendo o anteponiendo la programación.

4.1 Posponiendo la programación

Suelen implementarse mediante entornos gráficos de universos simulados de objetos, o bien mediante análisis, diseño y, posteriormente, programación OO.

Las del primer grupo proponen entornos interactivos (la charca virtual, la cesta de la compra...) repletos de objetos susceptibles de recibir mensajes [30,49,61,62], en los que el alumno describirá la secuencia de mensajes lanzados sobre objetos para conseguir cierto objetivo propuesto, aprovechando el supuesto carácter "intuitivo" del paradigma. Ya hemos rebatido la validez de esta hipótesis.

El empleo de herramientas interactivas para la presentación motiva al alumno, pero conlleva una sobresimplificación que genera expectativas irreales. Puede que le enseñe a comprender sus propios mecanismos de adquisición de conocimiento, pero no a programar mejor. No es un enfoque válido para abordar la docencia de primer curso.

Las propuestas basadas en análisis y diseño, mantienen que lo fundamental es entender el análisis y diseño de la parcela del mundo real a programar, no siendo necesario, en principio, recurrir a un lenguaje concreto. Las propuestas giran en torno al uso de UML o parecidos [32,47]. Algunos abordan este enfoque para la enseñanza de la OO, aunque la ubican en segundo curso [36,45].

La ventaja es la aproximación paulatina a los conceptos. Pero el argumento en contra es rotundo: el lenguaje determina el pensamiento. Si no se conoce el lenguaje no se puede ni diseñar ni analizar. Hay citas categóricas contrarias al enfoque de presentar la filosofía de un paradigma de forma disociada del desarrollo de código: "enseñar a programar a través de buenos diseños,

enseñar a diseñar a través de buenos patrones y, por último, enseñar a analizar" [T.Love]; "para dominar el análisis OO, es preciso haber aprendido los conceptos fundamentales desde el nivel de la implantación [...] Únicamente después de un encuentro, con las *manos en la masa*, con el uso operativo del método, estará uno preparado para comprender los conceptos del análisis OO" [43].

El informe de ACM/IEEE2001 previene sobre los efectos de trabajar sin un lenguaje de programación y llama la atención sobre el mayor esfuerzo requerido por parte del profesor.

Por tanto, pensamos que no se puede empezar con modelos, ni con análisis primero, para evitar la complejidad de la programación, sino, que conviene introducirla escalonadamente, reforzando cada concepto teórico con su correspondiente implantación concreta sobre un lenguaje ejecutable.

4.2 Anteponiendo la programación

Este enfoque se apoya en la siguiente tesis: ¿conoce usted algo "mejor" que la OO?, ¿no?, ¿por qué enseña algo que le parece "peor"? [16,18,35,40,54].

Habría otras alternativas, por ejemplo empezar con programación concurrente OO con interfaces gráficos, o usar lenguajes multiparadigmáticos como Leda [13] (lógico/funcionales/procedimentales orientados a procesos y a objetos) y resolver cada parte del dominio del problema con el paradigma más adecuado. Se podría empezar con XML y, desechar definitivamente el lastre de los datos formateados. Pero, ¿Tengo que enseñar "el mejor" paradigma?... ¿cuál será el mejor mañana? El objetivo de la docencia es prepararles para que puedan evolucionar en consonancia con la evolución del software y según las necesidades de su propio ejercicio profesional.

Algunos de estos trabajos plantean el aprendizaje a partir del estudio y adaptación de buenos programas diseñados por otros, que servirán como *patrone*s o modelos de nuevos programas [48,58]. En el entorno OO, muchos patrones interesantes conllevan complejas relaciones entre clases, que producen una sobrecarga, no sólo de conceptos, sino de diseño.

Otras propuestas conjugan la exposición de los conceptos OO con el uso de bibliotecas (*API* y *frameworks*) [41], que fomentan la abstracción, encapsulación, modularización, y jerarquización. No parece viable utilizar las clases de la biblioteca sin conocer el paso de parámetros por valor o referencia de los mensajes, sin conocer las estructuras de control de flujo de ejecución que ordenen los mensajes en el tiempo y aprender todo a la vez, es en una sobrecarga excesiva para el alumno.

Otras propuestas que anteponen la programación presentan un acercamiento a la OO desarrollando aplicaciones con interfaces gráficos de usuario [26,33,50,51,55,63,64], o basando el curso en la creación de interfaces coloridas y juegos. La solución se articula a través de *applets* o con *frameworks* simplificados e irreales.

Sus desventajas son la suma de las de otras propuestas. Respecto al temario, es preciso introducir simultáneamente: tipos, variables, expresiones, sentencias de control añadiendo clase, herencia, método y mensaje. Si es duro aprender programación estructurada en un curso, la introducción de estos nuevos conceptos produce una sobrecarga inabordable [26].

Respecto a mecanismos de diseño: también sobrecargan con relaciones de herencia, composición y asociación, impuestas desde el primer momento, para terminar construyendo un *applet,* que es el peor ejemplo de clase, del estilo *Conversor de Celsius a Fahrenheit* o similares; ¿no será la clase *GradoTemperatura* quién asume la responsabilidad de la conversión, donde el interfaz no calcula, sino que requiere y presenta los datos?

Respecto a la estructura de la interfaz, manejan un modelo de eventos globalizado o una arquitectura sin eventos, con comunicaciones por cadenas de caracteres entre todos los componentes del modelo/vista/controlador. Estos modelos no sirven como antesala de las bibliotecas industriales y no tienen envergadura para resolver problemas reales.

Por tanto, compartimos que "adoptar el paradigma OO en primero implica (al menos, tal como se ha llevado a cabo en muchos libros) exponer a los estudiantes a conceptos muy complejos y, a menudo delicados, antes de que dispongan de conocimientos adecuados para comprenderlos" [12]. Los, sin duda, loables objetivos de fomentar la reusabilidad y de desarrollar potentes aplicaciones desde el

principio no son justificación para empezar con OO; ni la reusabilidad, ni las aplicaciones motivadoras, con o sin interfaz gráfico de usuario, son patrimonio de la OO.

Además ACM/IEEE-2001 no considera la interfaz gráfica y las librerías, ni como objetivos, ni como unidades temáticas a considerar en los cursos de introducción en ninguno de sus 4 diseños curriculares (primero con o sin objetos, distribuido en 2 o 3 cursos). Ubicando, específicamente, estos contenidos en cursos intermedios de desarrollo del *software* o en computación gráfica. La *figura 2* presenta la dedicación porcentual en la propuesta curricular para API y GUI en los cursos introductorios y en los intermedios específicos de la materia.

		Horas Totales	APIs %	GUIs %
Primero Imperativa	2 cursos	80	0	0
	3cursos	120	2	2
Primero Objetos	2 cursos	80	5	2
	3 cursos	120	2	1
CS260T. Software Development	INTERMEDIOS	40	8	53
CS262. Software Development and Professional Para.		40	8	23
CS261S. Software Development and Systems Prog.		40	8	28
CS255. Computer Graphics and Multimedia		40	8	80

Figura 2.Ubicación de la docencia de API y GUI en la propuesta de ACM/IEEE.

5. Conclusiones

Hemos presentado una clasificación de los enfoques de la enseñanza de la OO propuestas por diferentes autores. Atendiendo a la viabilidad pedagógica, además de a la coherencia técnica, todas las propuestas estudiadas presentan algún inconveniente. Desafortunadamente, se han desarrollado pocos experimentos para medir la eficiencia pedagógica de los distintos enfoques.

Entendemos que la discusión sobre la planificación de los primeros cursos de programación no debe centrarse en el paradigma a presentar, ni en el lenguaje más adecuado para la implementación de los ejercicios, sino en los objetivos perseguidos y el modo de conseguirlos.

En concreto, el "problema del desplazamiento" surge como consecuencia del modo en que se han impartido los conceptos del paradigma procedimental, no por haber comprendido el concepto de *TAD* antes de abordar el de *clase*.

Urge disponer de una propuesta que resuelva los problemas planteados por las actuales, que permita la unificación de los planes de estudio, facilitando la movilidad de los estudiantes.

Por otro lado, nosotros compartimos que: "las teorías científicas son entidades que se extienden y perduran en el tiempo" [20]; "la OO no supone una revolución, sino, una evolución" [8,57] y que "la OO potencia y fortalece los conceptos que admiramos en la programación procedimental" [37]. De hecho, ACM/IEEE91 [1] detecta una serie de conceptos recurrentes "que ayudan a unificar una disciplina, se presentan reiteradamente en diversos contextos, son independientes de la tecnología concreta y se extienden a través de la historia".

En base a estas consideraciones hemos desarrollado una alternativa en la que los conceptos recurrentes son la clave que conduce la transición a través de los conceptos de la programación, "independientemente del paradigma concreto y a través de su evolución histórica", evitando la sobrecarga inicial que conlleva un enfoque OO en primero y eliminando el problema del desplazamiento de paradigma que aparece en las propuestas que comienzan por procedimental, al aportar la necesaria continuidad.

La descripción de dicha propuesta excede los objetivos del artículo y se desarrolla en otra ponencia de estas Jornadas [22].

Agradecimientos

Este trabajo se ha financiado con el proyecto TIC2000-1413 de la CICYT.

Referencias

[1]ACM/IEEE. *Computing curricula 1991*. http://www.computer.org/education/cc1991/eab1.html

[2]ACM/IEEE. *Computing curricula 2001*. http://www.acm.org/sigcse/cc2001/15-12-2001.html

[3]Adams, J.C. *Object centrered design a five-phase introduction to OO programming.* SIGCSE 1996, 78-82

[4]Albert, E. et al. *La enseñanza de Java en los estudios de informática.* JENUI 2000, 557-62

[5]Angster E. *A simple OO system pattern to introduce OOP with design*. OOPSALA'96 Workshop 1996

[6]Angster, E. *Our OO teaching concepts.* ECOOP 98

[7]Barr, M. et al. *An Exploration of novice programming errors in an OO environment.* SIGCSE 1999, 31(4) 42-6

[8]Bishop J. *Java gentile for engineers and scientists.* Addison-Wesley 2000

[9]Bisshop-Clark C.; Kiper J. *An undergraduate course in Object-Oriented software design.* 28 thFIE'98. Conference proceedings. IEEE, Piscataway.NJ,USA, (1) 1998pp 38-42

[10]Böszörményi, L. *Why Java is not my favorite first-course language software*. Concepts & Tools 1998, 19 141-145pp

[11]Brilliant, S.S.; Wiseman, T.R. *The first programming paradigm and language dilemma.* SIGCSE 1996, 338-42

[12]Buck, D.; Stucki, D. *Design early considered harmful: graduate exposure to complexity and structure based on levels of congnitive development.* SIGCSE 2000 3/00

[13]Budd, T. *Multiparadigm programming in Leda.* Addison-Wesley 1994

[14]Carter, J. et al. *Object Oriented students?* ITiCSE 1998, 271

[15]Casanova A. et al. *La enseñanza de la programación en los estudios de informática de la UPV* . JENUI 1998, 411-15

[16]Deckerr, R.; Hishfield, S. *The top 10 reasons why OO programming can't be taught in CS1*. SIGCSE94 (27) 51-4

[17]Declaración conjunta de los Ministros Europeos de Educación. Bolonia 19 de junio de 1999. www.universia.es/contenidos/universidades/documentos/universidades_docum_bolonia.htm

[18]DeClue, T. *OO and the principles of learning theory a new look at problems and benefits.* SIGCSE 1996, 232-6

[19]Détienne, F. Détienne, F. *Software-desing: cognitive aspects*. Springer 2002

[20]Díez, J.A.; Moulines, C.U. *Fundamentos de la filosofía de la ciencia.* Ariel, 1997

[21]Esteban A. et al. *La programación basada en Component Pascal: de la programación OO a los componentes del software.* JENUI 1998, 423-26

[22]Fernández Muñoz L., Peña R., Nava F., Velázquez Iturbide A. *Enfoque diacrónico para la enseñanza de los paradigmas de la programación imperativa.* JENUI 2002.

[23]Fernández, A.; Rossi, G. *An effective approach to learning object-oriented technology.* ECOOP 98.

[24]Franch, X. et al. *Transición de Modula2 a Java en un curso de iniciación a la programación.* JENUI 1998, 319-326

[25]Franch, X.; et al. *An introductory programming pilot course using Java* EATCS 1999. pp 69

[26]García Molina, J. *¿Es conveniente la OO en primer curso de programación.* Novática 154 (11/12) 2001, 64-8

[27]Hardgrave, B.C.; Douglas, D.E. *Trends in information systems curricula: object-oriented topics.* Journal of Computer Information System vol 39, nº1, 1-6.

[28]Hitz, M.; Hudec, M. *Modula2 versus C++ as a first programming language. Some empirical results*. SIGCSE 1995, 317-21

[29]Hosch F. *Java as a First language: an evaluation.* SIGCSE Bulleting 28(3) 1996 45-50

[30]Houle, M.E. *Ethics, programming, and virtual environments.* ITiCSE 1997, 91-3

[31]Joyner, I. Objects u*nencapsulated: Java, Eiffel and C++*. Prentice Hall, 1999

[32]Kelemen B. *A Newcomer's Thoughts about responsibility distribution.* ECOOP'98.

Lecture Notes in Computer Science, 1543 Springer-Verlag 1999.

[33]Koffman E. Wolz U. *A simple Java package for GUI-like interactivity*. SIGCSE 2001, 11-15

[34]Kölling, M. et al. *Requirements for a first year OO teaching language.* SIGCSE 1995, 173-77

[35]Kölling, M. *The problem of teaching object-oriented programming. Part I: Languages* Journal of Object-Oriented Programming 1999, 11(8) 8-15pp

[36]Kornecki A. J. *Teaching object-oriented simulation in a software engineering framework. Simulation*. (abril 2001) 233-239

[37]Lewis, J. *Myths about OO and its pedagogy.* SIGCSE 2000, 245-49

[38]Lidtke, D.K.; Zhou, H.H. *A New approach to an introduction to computer science.* ASEE/IEEE 29th FIE'99 4-23, 1999

[39]Llopis, F.; Pérez, E. *C++-OO como lenguaje introductorio a la programación*. JENUI 2001, 299-304

[40]Lucker *There's more to OOP than syntax!.* SIGCSE Bulletin 26 (1), p 56-60 (1994)

[41]McCracken, D.D. *An inductive approach to teaching object-oriented design.* SIGCSE 1999 3/24-28

[42]McDonald, C. *1st year programming languages in australian and New Zealand universities*. http:// www.cs.uwa.edu.au/~chris /java-in-cs1/anzacs.html

[43]Meyer, B. *Object-Oriented software construction, 2ªed.* Prentice-Hall, 2000 ISBN0136291554

[44]Meyer, B. *Towards an OO curriculum. Journal OO Programming* 1994, (3)76-81

[45]Moreira A. *Teaching objects: the case for modelling*. ECOOP'98 (1998), Lecture Notes in Computer Science, 1543 pp350-354 Springer-Verlag 1999

[46]Negroponte N. *Ser digital* (1995) Atlantida Publishing; ISBN: 9500814730

[47]Oliveira, C.A. *Aspects on teaching/learning with OO programming for entry level Courses of engineering.* ICEE 1998, (8)17-20

[48]Ortega, A. et al. *Programación multilenguaje con Component Pascal y Java en un primer curso de programación.* JENUI 1999, 343-48

[49]Osborne M., Johnson J. *An only undergraduate course in object-oriented technology.* SIGCSE Bulletin. 25(1) 101-106 (1993).

[50]Rasala, R. et al. *Java Power Tools: Model software for teaching OO design*. SIGCSE 2001, 297-301

[51]Rasala, R. *Toolkits in first year computer science: a pedagogical imperative.* SIGCSE 2000, 185-91

[52]Rayside, D.; Campbell, G. *Aristotle and OO programming: why modern students need traditional logic.* SIGCSE 2000, 237-44

[53]Reges, S. *Conservatively radical Java in CS1*. SIGCSE 2000 3/00

[54]Roberts, E. *An Overview of MiniJava.* SIGCSE 2001 2/01

[55]Schaub, S. *Teaching Java Graphics in CS1*. SIGCSE 2000 32(2)71-3

[56]Schoenefeld, D.A. *OO Design and Programming: an Eiffel, C++, and Java Course for C Programmers*. SIGCSE 1997, 135-9

[57]Stroustrup B. *The Design and Evolution of C++*. Addison-Wesley, 1994. ISBN 021543303

[58]Wallingford, E. *Toward a first course based on OO patterns.* SIGCSE 1996, 27-32

[59]Wick, M.R. *On Using C++ and OO in CS1: the Message is still more important than the Medium* SIGCSE 1995, 322-6

[60]Wiedenbeck, S.; Ramallingam, V. *Novice Comprhension of Small Programs Writen in the Procedural and OO Styles.* International Journal of Human Computer Studies (1999) 51 71-87

[61]Woodman, H.; Robinson, H.; Griffiths, R.; Holland, S. *An Object Oriented Approach to Computing* OOPSLA 98

[62]Woodman, M. et al. *The Objects Shop-Using CD-ROM Multimedia to Introduce Object Concepts*. SIGCSE 1997, 345-9

[63]Woodman, M.; Holland, S. *From software user to software author: an initial pedagogy for introductory OO computing.* ITiCSE 1996, 60-2

[64]Woodworth P.;Dann W. *Integrating Console and event-driven Models in CS1*. SIGCSE 1999 (3)132-135

Una propuesta para organizar la enseñanza de la Orientación a Objetos

Jesús García Molina, Marcos Menárguez Tortosa y Begoña Moros Valle

Depto. de Informática y Sistemas
Universidad de Murcia
Campus de Espinardo-30071 Murcia
e-mail: (jmolina|marcos|bmoros)@um.es

Resumen

En este trabajo se presenta una propuesta sobre cómo organizar en los planes de estudio de las titulaciones de informática los temas relacionados con la orientación a objetos. Se describen dos asignaturas cuatrimestrales obligatorias de seis créditos: una primera de *Introducción a la Programación Orientada a Objetos* y una segunda sobre *Análisis y Diseño Orientado a Objetos*. Proponemos que la primera sea obligatoria en las tres titulaciones y que la segunda sea obligatoria en Ingeniero en Informática y optativa en las dos titulaciones técnicas. Ambas conforman un curso que proporcionará al alumno una formación básica en el desarrollo de software orientado a objetos. Esta propuesta está en la línea expuesta en el *Computing Curricula 2001* de ACM/IEEE.

1. Introducción

En la década pasada la programación orientada a objetos (OO) se ha consolidado como el paradigma de programación más apropiado para construir software de calidad, sobre todo porque favorece la escritura de código reutilizable y extensible, además de reducir el salto semántico entre los modelos del análisis del problema y el código de la aplicación: los objetos del dominio del problema son objetos de la solución software.

La propuesta curricular *Computing Curricula 2001* de ACM/IEEE [2] reconoce la importancia adquirida por la programación OO desde la publicación de las recomendaciones curriculares de 1991 y añade este área al núcleo de conocimientos básicos de la disciplina, estableciendo que cualquier estudiante de informática debería tener una formación en Programación OO (unidad *PL6* de Lenguajes de Programación), Análisis y Diseño OO y Patrones de Diseño (unidad *SE1* de Ingeniería del Software).

Las directrices generales propias de las titulaciones de informática (B.O.E. de 20 de Noviembre de 1990) no incluyen descriptores relacionados con la OO en ninguna de las materias troncales, de modo que la forma de incorporar a los planes de estudio la enseñanza obligatoria de la OO, es a través de asignaturas obligatorias, tal y como se está haciendo en la mayoría de los planes de estudio aprobados recientemente, como es el caso de los planes de la Universidad Politécnica de Valencia y de la Universidad de Alicante, entre otros. También se puede aprovechar las asignaturas que corresponden a la materia troncal *Ingeniería del Software* para incluir contenidos sobre construcción de software OO.

En nuestra facultad se imparte una asignatura sobre programación OO desde el curso 1991-92, en el que se implantaron los estudios de segundo ciclo. Se trataba de una asignatura de cuarto curso, cuyos descriptores correspondían a la materia troncal *Ingeniería del Software*. En la actualidad, nuestros planes de estudio incluyen la asignatura *Programación Orientada a Objetos* como optativa en las dos titulaciones técnicas (planes de 1994) y como obligatoria en el primer ciclo de Ingeniero en Informática (plan de 1996, con otro nombre). Esta titulación también incluye una asignatura troncal en cuarto curso sobre análisis y diseño OO.

En este trabajo analizamos cómo organizar la enseñanza del paradigma OO en los planes de estudio de informática. Presentamos una propuesta para la titulación Ingeniero en Informática, indicando como trasladarla a las titulaciones técnicas. Esta propuesta es fruto de nuestra experiencia al impartir las asignaturas relacionadas con la OO en nuestra facultad.

El trabajo se organiza del siguiente modo. En la siguiente sección presentamos nuestra propuesta para Ingeniero en Informática. Luego describimos las dos asignaturas que conforman la enseñanza obligatoria sobre OO que proponemos y que actualmente impartimos en nuestra facultad. A continuación comentamos cuál es la situación en el caso de las titulaciones técnicas y en los futuros planes. Finalmente presentamos las conclusiones.

2. La propuesta

Creemos que un ingeniero en informática debe recibir una formación básica en orientación a objetos a través de dos cursos: uno de *introducción a la programación OO* (nos referiremos a él como IPOO) y otro de *análisis y diseño OO* (nos referiremos a él como ADOO). En IPOO se introducen los conceptos básicos que caracterizan al paradigma de programación OO y en ADOO el alumno conoce y aprende a aplicar un proceso software OO y los patrones de diseño básicos. IPOO se desarrollaría como una asignatura de tercero de seis créditos, una vez que el alumno conoce la programación procedural y modular, mientras que ADOO se cursaría como una asignatura de cuarto curso también de seis créditos. Es recomendable que ambos cursos no se impartan en el mismo año ya que es bueno que el alumno tenga tiempo para madurar esa nueva visión de la programación que le muestra IPOO.

Desde hace cinco años, nuestro departamento sigue este enfoque en los estudios de Ingeniero en Informática, a través de la asignatura obligatoria *Diseño de Programas* en tercero (corresponde a IPOO) y de la troncal *Arquitectura del Software* de cuarto (corresponde a ADOO). En este trabajo utilizaremos los nombres IPOO y ADOO, en vez de los nombres de estas asignaturas, para insistir en el carácter general de nuestra propuesta.

La organización de IPOO presupone un enfoque *procedural-primero* para la introducción a la programación del primer año, por los motivos que se exponen en [5]. En la mencionada propuesta curricular de ACM/IEEE [2] se señala que es posible idear planes en los que la introducción a la programación puede ser abordada tanto a través del paradigma imperativo como del paradigma OO o incluso del funcional, siendo una cuestión a debatir en cada facultad. En [5] se defiende comenzar con el paradigma imperativo, aunque con un enfoque en el que los conceptos de secuencia e inducción juegan un papel central, así como el de tipo abstracto de datos.

Proponemos pues seguir un enfoque que podemos llamar *evolutivo*, en el sentido que pasamos de la programación procedural a la modular y luego aterrizamos en la programación orientada a objetos. En primer año una introducción a la programación tal y como la descrita en [5]; en segundo curso el alumno aplicaría la programación modular dentro una asignatura anual de algoritmos y estructuras de datos, antes de cursar las dos asignaturas mencionadas arriba IPOO y ADOO que en su conjunto constituyen un curso completo sobre construcción de software OO que incluirían todos los contenidos sobre OO considerados como básicos en la recomendación curricular de ACM/IEEE [2].

B. Meyer critica este enfoque evolutivo al considerarlo fruto de una visión equivocada del profesor que cree necesario enseñar la programación siguiendo el orden en que él ha conocido los paradigmas, y se pregunta por qué no empezar con los objetos desde el primer año, dado que todos estamos de acuerdo en que la OO es el medio adecuado para construir software. Aunque reconocemos el papel de B. Meyer en el desarrollo de la OO y sus grandes aportaciones, después de reflexionar sobre esta cuestión y tras una experiencia de más de diez años creemos que nuestro enfoque *procedural-modular-OO* es el camino adecuado para abordar la enseñanza de la programación, ya que el alumno puede valorar mucho mejor las ventajas de la OO y además es más natural el paso *procedural-OO* que al revés. De hecho, en la primera parte del libro [7], B. Meyer justifica las ventajas de la OO partiendo de la experiencia del lector en programación

procedural, en el manejo de rutinas y módulos, y en el conocimiento del diseño descendente.

Lógicamente, el curso IDOO es un prerrequisito para el curso ADOO. En estos dos cursos, el alumno recibe una formación que le capacita para estudiar otras temas relacionados con la tecnología del software OO como es la construcción de aplicaciones basadas en objetos distribuidos (RMI y CORBA) y el desarrollo basado en componentes. Nuestro departamento oferta en quinto curso dos asignaturas relacionadas con estas dos tecnologías.

Conviene señalar que en las titulaciones técnicas se sigue el enfoque evolutivo comentado, pero no en la titulación superior ya que dos de las cuatro asignaturas de programación que hay entre primero y segundo no están adscritas a nuestro departamento y en ellas se utiliza Java para la enseñanza de los tipos abstractos de datos y estructuras de datos. En el siguiente apartado comentaremos este hecho. A continuación describiremos los cursos IDOO y ADOO que proponemos.

3. Introducción a la Programación Orientada a Objetos

El objetivo de esta asignatura es introducir al alumno en el paradigma de programación OO, siendo sus objetivos: i) describir los conceptos que lo caracterizan: clase, objeto, herencia, polimorfismo y ligadura dinámica, ii) contrastar cómo esos conceptos se reflejan en los cuatro lenguajes OO más extendidos: C++, Java, Smalltalk y Eiffel, iii) enseñar un lenguaje OO y un entorno de programación, iv) introducir algunas técnicas y heurísticas básicas de programación OO y v) valorar en qué medida las técnicas OO mejoran la calidad del software.

La asignatura se plantea como dos cursos que se imparten en paralelo desde la primera semana: una parte teórica de treinta horas y una parte práctica con quince sesiones de hora y media en el laboratorio. En las clases prácticas el alumno recibe un curso de programación OO en Java. Este curso se imparte de forma simultánea a las clases de teoría, ya que no es viable llevar las clases prácticas al ritmo de las clases teóricas. Esto no crea confusión en el alumno, ya que las clases teóricas sirven para proporcionar una visión más general de los conceptos que previamente ha visto en las clases prácticas en el contexto de Java. Al final del curso, las dos partes han debido servir para conseguir los objetivos y que el alumno tenga unos buenos conocimientos sobre programación OO, dos partes separadas que forman una unidad coherente.

En las clases teóricas describimos con detalle todos los conceptos que caracterizan al paradigma OO. Primero se estudian todos los conceptos relacionados con clases y objetos: ocultación de información, mensajes, semántica referencia, creación y copia de objetos, igualdad e identidad, y genericidad. Después se estudia un capítulo sobre el *diseño por contrato* que sirve para contrastar los mecanismos de asertos y excepciones de Java y Eiffel. A continuación se estudian todos los aspectos que tienen que ver con la herencia: principio de sustitución, polimorfismo y ligadura dinámica, intento de asignación, genericidad en Java, clases abstractas, código genérico, herencia y ocultación de información, herencia múltiple, herencia repetida, diferentes tipos de herencia, e implementación de la ligadura dinámica. En el estudio de la herencia también se incluye la descripción de algunos patrones básicos como son: *Template Method* (para ver el papel de las clases parcialmente diferidas para escribir código genérico), *Iterador* (distinguiendo entre iteradores externos e internos, analizando cómo resolver con herencia la imposibilidad de pasar funciones como argumentos de un método), *Observer* (para ver el papel que pueden jugar las interfaces Java) y *Composite* (como ejemplo de herencia múltiple). Finalmente se valora cómo la OO ayuda a mejorar la calidad del software y se introducen heurísticas básicas para la creación de software OO: algunas de las heurísticas de Riel [8] y los patrones *Controlador* y *Experto* [6].

En un curso de estas características es importante que el alumno no adquiera la visión de la OO ofrecida por un lenguaje particular, sino que comprenda los conceptos que subyacen a la OO de una forma independiente del lenguaje. Para conseguir este objetivo, en la parte teórica primero se explican los conceptos sin recurrir a un lenguaje y luego se muestra el contraste en la forma en que son reflejados en los cuatro lenguajes OO que consideramos. Con este enfoque no se encorseta la visión del alumno a la ofrecida por un lenguaje concreto, sino que

adquiere una compresión de los conceptos de más alto nivel. No cabe duda que el contraste enriquece la visión del alumno, ya que se favorece su actitud crítica y se le prepara para poder programar, sin grandes dificultades, en cada uno de los cuatro lenguajes, ya que conoce los aspectos clave relacionados con la OO en cada uno de ellos. Además, está capacitado para valorar cómo se introducen los conceptos OO en otros lenguajes existentes o que puedan definirse en un futuro (por ejemplo, podría dominar sin dificultad C#).

Para ilustrar la idea sirva el ejemplo de la *ocultación de información*. Al alumno se le muestra su significado y se le justifican sus beneficios, entonces se revisan y se contraponen las variaciones que encontramos en los diferentes lenguajes: atributos públicos en C++ y Java, clases amigas en C++, visibilidad *friendly* en Java, clases anidadas en Java y C++, exportación selectiva en Eiffel, todos los métodos son públicos en Smalltalk ocultación a descendientes en Java y C++, …

En la parte práctica nos hemos tenido que enfrentar al problema de la elección de un lenguaje OO. El lenguaje ideal para la docencia sería aquel que tuviese buenas propiedades, existiesen buenos entornos y abundante documentación, y que además su uso estuviese muy extendido entre las empresas de desarrollo. Desde nuestro punto de vista, que un lenguaje tenga buenas propiedades supone que debería: i) ser OO puro para obligar al alumno a programar dentro del paradigma, ii) reflejar con claridad los conceptos OO, por ejemplo, no parece conveniente poder declarar que un atributo se exporta en modo escritura o tener que declarar los métodos en los que se aplicará ligadura dinámica o no poder declarar métodos privados, iii) ser legible, iv) ser pequeño y con gran potencia expresiva, v) ser tipado estáticamente, para favorecer la legibilidad y fiabilidad. Además, sería conveniente que permitiese escribir asertos para posibilitar la programación por contrato [7] y que incluyese un mecanismo de genericidad. En cuanto a la exigencia de un buen entorno entendemos la existencia de entornos robustos, gratuitos, fáciles de usar y que no necesitasen muchos recursos.

Como es lógico, no existe un lenguaje que cumpla todos los requisitos anteriores. Creemos que Eiffel o Eiffel# podría ser la mejor elección considerando sus buenas propiedades (OO puro, tipado, legible, incluye genericidad, mecanismo de asertos y un entorno robusto), sin embargo está muy poco extendido y no hay buenos entornos gratuitos. Por ello hemos elegido Java que podemos considerar *casi OO puro* ya que el alumno no puede salirse del paradigma OO al programar (aunque no es OO puro ya que hay una distinción entre tipos básicos y clases, un método *main* en las clases y otras cuestiones), es tipado estáticamente, refleja con sencillez los conceptos OO si lo comparamos con C++, existe abundante documentación y sobre todo está muy extendido en la industria, habiendo surgido alrededor de él una importante tecnología para el desarrollo de aplicaciones OO: Servlet, RMI, Beans, EJB, JINI,.. Con Java tenemos un lenguaje que no tiene tan buenas propiedades como Eiffel (por ejemplo, no tiene genericidad, ni herencia múltiple) pero a cambio se trata de un lenguaje muy extendido que es el lenguaje más utilizado en el desarrollo de aplicaciones web. Además, el alumno seguirá trabajando con Java en asignaturas posteriores que traten de tecnología del software, por ejemplo las relacionadas con componentes y objetos distribuidos.

Hasta hace dos años, Smalltalk fue el lenguaje elegido por tratarse de un lenguaje OO puro, muy simple pero de gran potencia expresiva y con buenos entornos de libre distribución que consumen pocos recursos. Nunca nos planteamos C++ por tratarse de un lenguaje híbrido, muy complicado y poco legible. Al aparecer Java, y a pesar de su rápido éxito, no apostamos por él ya que no era un lenguaje estable, no se disponían de buenos entornos y por la resistencia al cambio. Entendíamos que Smalltalk permitía cumplir los objetivos y teníamos mucha experiencia y documentación. Sin embargo, ahora no dudamos que Java es la mejor elección.

Como trabajo práctico, el alumno debe realizar un boletín de ejercicios, de complejidad creciente, en los que debe demostrar su conocimiento de aspectos propios de la programación en Java, (entrada/salida, excepciones, concurrencia, serialización e interfaces gráficos), así como de los conceptos de la OO y el manejo de las clases básicas de la jerarquía como son las colecciones. Estos ejercicios obligan al alumno a diseñar e

implementar sus primeras clases para resolver problemas concretos, a ver los programas como una colección estructurada de clases. Finalmente debe desarrollar un pequeño proyecto de programación que abarque de una forma global todos los conceptos estudiados en los ejercicios, en el que se puedan aplicar algunos patrones y heurísticas como la *separación modelo-vista*, el patrón *Observer* y los patrones de reparto de responsabilidades *Controlador* y *Experto* [6]. En los dos últimos cursos hemos encontrado adecuado como proyecto de programación la implementación de un juego en el que varios personajes interactuan en un escenario que impone unas reglas. La implementación exige diseñar las clases que representan el escenario y una jerarquía interesante para representar los personajes del juego (con clases abstractas, código genérico, interfaces,...), y utilizar unos patrones sencillos de colaboración entre objetos.

Conviene señalar que en este curso se cubren todos los temas incluidos en PL6 (*Object-Oriented Programming*) dentro del *Computing Curricula 2001*, así como los objetivos de aprendizaje que allí se plantean.

Las encuestas y comentarios de los alumnos muestran su rechazo a la utilización de Java en dos asignaturas de primer y segundo curso, dedicadas fundamentalmente a trabajar con estructuras de datos y el concepto de tipo abstracto de datos. Los alumnos se quejan de que deben utilizar los conceptos OO sin conocerlos, actuando a base de recetas, "aquí hay que poner *Object*", "aquí debes hacer una conversión de tipos", "las interfaces son un tipo de clases". Reconocen que el conocimiento que tienen de Java no les ha sido de mucha ayuda a la hora de cursar IPOO, ya que tenían pocos conceptos claros. Estas opiniones refuerzan nuestra idea de empezar con el paradigma procedural en el primer curso.

4. Análisis y Diseño OO

Este curso es una continuación del anterior y se dedica a proporcionar al alumno una formación que le permita abordar de una forma sistemática el desarrollo de aplicaciones OO, aplicando un proceso software. Se supone una parte teórica y otra práctica de tres créditos cada una. En las clases teóricas, primero se describe el *Lenguaje Unificado de Modelado*, *UML* como notación estándar para el modelado OO, luego se describe un proceso basado en UML y finalmente se presentan y discuten los patrones de diseño de Gamma et al. [3].

No cabe duda que actualmente el modelado OO está centrado en UML que se ha convertido en un estándar "de facto". En este curso se realiza una descripción detallada del lenguaje UML, analizando con detalle el modelado de casos de uso, modelado estructural, modelado del comportamiento y modelado dinámico.

Tras la presentación de UML, se describe un proceso para UML orientado al desarrollo de aplicaciones de gestión (*business applications*). El proceso presentado es simple y está destinado a aplicaciones cuyo desarrollo no involucra equipos de programadores ni tiempos de desarrollo muy grandes. Se trata de un proceso iterativo, incremental y guiado por los casos de uso, que es resultado de añadir una etapa de modelado de negocio al proceso descrito por Craig Larman [6]. El modelado de casos de usos y el modelado conceptual se realiza a partir de los diagramas de actividad del modelado del negocio siguiendo las técnicas descritas en [4] para identificar casos de uso, vocabulario del sistema y reglas de negocio.

Dentro de una formación básica en orientación a objetos es obligado incluir los *patrones de diseño* o soluciones a problemas recurrentes en el desarrollo de software OO. Los patrones han adquirido bastante importancia en la comunidad software, siendo fundamentales para obtener una arquitectura software de calidad. En la actualidad, existe un consenso al considerar que los patrones de diseño descritos en el libro *Design Patterns* [3] son parte del núcleo básico de conocimientos de informática que un estudiante debe conocer, como se señala en el *Computing Curricula 2001*. Casi la mitad de la parte teórica de la asignatura se dedicará al estudio detallado de estos patrones, planteándose ejemplos que implican identificar qué patrones conviene utilizar.

Dentro de esta parte teórica también se introduce el problema de la persistencia en aplicaciones OO, analizándose con detalle el almacenamiento de objetos en bases de datos relacionales y presentándose el diseño de un nivel de persistencia [1,6]. Este diseño también sirve para ilustrar el uso de patrones.

Las clases prácticas se dedican al principio a resolver problemas de modelado de casos de uso, modelado estructural y el diseño de colaboraciones entre objetos (un total de 5 sesiones de una hora). Es bien sabido que la parte más difícil del modelado OO es diseñar cómo deben colaborar un grupo de objetos para realizar una determinada actividad [3, 6], por lo que se le presta especial atención. Más adelante se describe como aplicar el proceso software estudiado en clase sobre un ejemplo concreto; por ejemplo, se parte de la especificación de la práctica del curso anterior (tres sesiones de una hora). Luego se describe la herramienta que el alumno utilizará para el modelado, *Rational Rose* en nuestro caso (esto conlleva tres sesiones de hora y media). Finalmente se le plantea al alumno una especificación de un problema que debe modelar hasta llegar a una implementación en Java.

El trabajo práctico ideal para esta asignatura sería un proyecto de desarrollo de una aplicación para una empresa, desde el modelado del negocio hasta la implementación. El profesor entregaría al alumno una especificación de un sistema y el alumno debería primero hacer el modelado del negocio y luego aplicar el proceso descrito en [6]. Sin embargo, esto no tiene cabida en una asignatura cuya carga práctica son tres créditos por lo que la implementación se limita a unos pocos casos de uso y no se presta importancia a la interfaz gráfica y a la persistencia en bases de datos relacionales.

Los alumnos forman grupos de dos y el profesor mantiene dos reuniones con los alumnos para seguir el proyecto, una al acabar el modelado de casos de uso y modelado conceptual, y otra al modelar una serie de colaboraciones que son elegidas por el profesor. El objetivo de la primera reunión es asegurarnos que el alumno ha comprendido bien los requisitos, ha identificado una colección de casos de uso apropiada y los ha descrito bien, haciendo uso de la plantilla utilizada en [6]. El objetivo de la segunda entrevista es comprobar que los alumnos han comprendido bien cómo diseñar colaboraciones entre objetos. Esta actividad es la que les resulta más complicada y a la que dedican mayores esfuerzos.

El modelado del negocio es opcional para los alumnos, ya que involucra un tiempo considerable y preferimos llegar a la implementación de alguna parte del sistema, aunque son bastantes los grupos que lo desarrollan ya que lo encuentran útil para identificar los casos de uso. El alumno entrega una documentación que refleja cada etapa de aplicación del proceso y cada grupo mantiene una entrevista final con el profesor en la que se evalúa el trabajo práctico en su conjunto.

Conviene señalar que este curso cubre los temas y objetivos de aprendizaje incluidos en SE1 (*Software Design*) del *Computing Curricula 2001* que tienen que ver con el análisis y diseño OO y patrones de diseño: “seleccionar y aplicar patrones de diseño apropiados en la construcción de una aplicación”, “crear y especificar el diseño software para un producto software de tamaño medio, usando un método de desarrollo de software y notación apropiada.

En nuestro caso, este curso también permite que los alumnos puedan contrastar la aplicación de un proceso software OO con el análisis y diseño estructurado que ya conocen de la asignatura *Fundamentos de Ingeniería del Software* de tercer curso. Un aspecto que los alumnos valoran en gran medida es la trazabilidad que ofrecen los casos de uso dentro de los procesos OO: se diseñan colaboraciones que satisfacen un objetivo que corresponde a un caso de uso y se implementan clases teniendo en cuenta las colaboraciones, de modo que es fácil llegar del código al requisito que implementa. Los alumnos también valoran positivamente la continuidad entre las diferentes etapas del proceso, además notan como todo el trabajo es de utilidad y está encaminado a escribir código que es lo que hay que acabar haciendo.

5. La propuesta en las titulaciones técnicas y los futuros planes

En las titulaciones técnicas creemos necesaria una asignatura obligatoria que corresponda a IPOO en el primer cuatrimestre del tercer curso. Se podría incluir una optativa en el segundo cuatrimestre de este tercer año con los contenidos de ADOO, aunque nos parece que el proyecto de la parte práctica y el estudio de los patrones no puede ser tan exigente como el caso de la asignatura de cuarto curso, ya que el alumno tiene menos tiempo para madurar los conceptos.

Actualmente, en las titulaciones de Sistemas y Gestión de nuestra facultad, nuestro departamento

oferta una asignatura optativa denominada *Programación Orientada a Objetos* (6 créditos) que coincide con IPOO, aunque con algunas diferencias en el planteamiento de las prácticas. Se exige un menor número de ejercicios individuales y se propone como proyecto de programación una aplicación de gestión, guiándoles en la implementación, aunque sin aplicar un proceso. Este enfoque es motivado por el hecho que la asignatura no tiene continuidad en otra de la naturaleza de ADOO. Siempre se elige una especificación que al menos exija la definición de una jerarquía de clases y la existencia de clases parcialmente diferidas con código genérico (patrón *Template Method*). Se introducen los diagramas de clase de UML y se insiste en la aplicación de heurísticas básicas de diseño OO como no incluir código de las clases del modelo en las interfaces y el reparto de responsabilidades entre clases.

En los futuros planes, que se implantarán el próximo curso, en Ingeniero en Informática se mantiene la organización propuesta en este trabajo, con dos asignaturas que corresponden a IPOO y ADOO en tercer y cuarto curso, respectivamente; además se mantienen las optativas de quinto curso sobre objetos distribuidos y componentes. En cuanto a las titulaciones técnicas, la asignatura *Programación Orientada a Objetos* será obligatoria en tercer curso de Sistemas y Gestión. Además, se ha incluido en la oferta de optativas una asignatura que corresponde a la descripción de ADOO. La organización refleja la propuesta presentada por los autores al departamento sobre la enseñanza de las materias relacionadas con la OO.

6. Conclusión

Hemos presentado una propuesta de organización de la enseñanza de las materias relacionadas con la OO: una asignatura obligatoria de seis créditos en tercer curso (titulaciones técnicas e Ingeniero en Informática), destinada a ofrecer una introducción a los conceptos que configuran el paradigma OO y una asignatura, también de seis créditos en cuarto curso (como optativa de tercero en las técnicas) para describir un proceso basado en UML y los patrones de diseño básicos. Ambas asignaturas conforman un curso de doce créditos sobre OO, con el que los alumnos conseguirán una buena formación en el desarrollo de software OO y que coincide con la propuesta curricular de *Computing Curricula 2001*. Además, estos dos cursos preparan al alumno para estudiar otros aspectos de la tecnología del software como son las aplicaciones basadas en objetos distribuidos y el desarrollo basado en componentes.

Este enfoque de la enseñanza de la OO, es fruto de la colaboración, en los últimos cuatro años, de los autores de este trabajo al encargarse de la enseñanza de las asignaturas relacionadas con IPOO y ADOO, partiendo de la experiencia del primer autor que hace once años comenzó a impartir la docencia relacionada con la OO en nuestra facultad.

Referencias

[1] S. Ambler, *Mapping Objects to Relational Databases,* http://www.ambysoft.com/mappingObjects.html

[2] *Computing Curricula 2001*, Final Report, December 2001, ACM e IEEE, http://www.computer.org/education/cc2001/final/index.htm

[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 International Conference on Conceptual Modeling, Utah, USA, 2000. Versión en castellano *Un proceso basado en UML para aplicaciones de gestión* en http://dis.um.es/~jmolina/as.html

[5] J. García Molina, *¿Es conveniente la orientación a objetos en un primer curso de programación?*, VII Jornadas de Enseñanza Universitaria de la Informática (JENUI'2001), Palma de Mallorca, 16-18 de Julio, 2001. Puede descargarse en http://dis.um.es/~jmolina/publi.html

[6] C. Larman, *Applying UML and patterns*, 2ª edición,Prentice-Hall, 2002

[7] B. Meyer, *Construcción de Software Orientado a Objetos,* 2ª edición,Prentice-Hall, 1998.

[8] A. Riel, *Object-Oriented Design Heuristics,* Addison-Wesley, 1996.

Metodología basada en descomposición funcional y orientación a objetos en la introducción a la programación

Mercedes Gómez Albarrán
Dpto. de Sistemas Informáticos y Programación
Universidad Complutense de Madrid
28040 Madrid
e-mail: albarran@sip.ucm.es

Resumen

Este trabajo realiza una propuesta docente para un amplio cuerpo de materia de iniciación a la programación en el que tienen cabida la Programación Orientada a Objetos (POO) y una metodología de programación más tradicional como es la basada en descomposición funcional. Se muestra cómo podría quedar reflejada dicha propuesta en el plan de estudios de la Ingeniería en Informática de la Universidad Complutense de Madrid.

1. Introducción

A comienzos de los años noventa, en línea con las recomendaciones de *Computing Curricula 1991*, las asignaturas introductorias a la programación y la presentación de la POO ocupaban posiciones muy distintas en el currículo: las primeras presentaban una metodología de diseño procedimental usando un lenguaje de programación imperativo, con frecuencia Pascal, mientras que la POO aparecía en asignaturas de cursos avanzados.

En nuestros días vivimos lo que podríamos denominar un período de *(r)evolución* en lo que respecta a la enseñanza de la materia de introducción a la programación. Unos apuestan por mantener el enfoque "tradicional"; otros apuestan por incluir la orientación a objetos (OO) en el currículo introductorio. No existe un consenso en la comunidad docente en lo que respecta a la forma de abordar la introducción a la programación, incluso el propio *Computing Curricula 2001* [1] no se decanta por una estrategia concreta.

La tendencia a incluir la OO en el currículo introductorio es cada vez mayor. La cuestión es, ¿cómo se incluye? Podemos distinguir dos enfoques: el enfoque "objetos más tarde" (*objects-late*), en el que se presenta la OO tras la metodología basada en descomposición funcional, y el enfoque "objetos primero" (*objects-first*), que comienza presentando directamente los fundamentos de la POO[1].

La propuesta que aquí se plantea se enmarca dentro del enfoque "objetos más tarde" y se articula en torno a las siguientes ideas:

- Es innegable el papel fundamental que tiene la OO en el desarrollo de software actual y parece claro que será el modelo de programación dominante en un futuro próximo. De ahí que apostemos por no relegarlo a una enseñanza de "segundo nivel" –entendiendo por "segundo nivel" una presentación alejada del primer o segundo año de estudios–, sino por que forme parte de la formación básica en programación de los alumnos.
- La metodología basada en descomposición funcional ha sido y sigue siendo una metodología válida, que permite a los alumnos enfrentarse al desarrollo de algoritmos aislados y pequeñas aplicaciones. En consecuencia, tampoco consideramos que deba ser excluida. En este sentido, un enfoque "objetos primero" discrimina a la metodología basada en descomposición funcional, tal y como se indica en [15].

[1] Existen diversas tendencias dentro del enfoque "objetos primero" según se dé más importancia al uso de clases, a la creación de clases, a la construcción de jerarquías, etc. [3][7].

- La descomposición funcional y la descomposición conducida por los datos (en la que se apoya el diseño OO) no son metodologías disjuntas y mutuamente excluyentes. La Ingeniería del software nos aconseja que el diseño conducido por los datos (la abstracción de datos) domine el desarrollo de aplicaciones de cierta envergadura. Pero cuando hablamos de abstracción de datos no hablamos sólo de los valores sino también de operaciones asociadas. Las operaciones requieren algoritmos. La descomposición funcional es aplicada para diseñar esos algoritmos, y también puede serlo en la coordinación de las interacciones entre datos que representa el código conductor de la aplicación.

Con estas ideas subyacentes, en el bloque de materia de iniciación a la programación que se propone, se presentarían, en este orden:

- Descomposición funcional para el desarrollo de algoritmos concretos y aplicaciones a pequeña escala.
- Descomposición conducida por los datos, para el diseño de aplicaciones no triviales. En este punto la abstracción procedimiental no es eliminada sino subyugada a la abstracción de datos que debe ocurrir antes en el proceso de diseño.
- Orientación a objetos, una técnica basada en el diseño conducido por los datos que incluye mecanismos que facilitan la creación de software extensible y reutilizable.

A continuación, pasamos a describir los objetivos de una propuesta como la nuestra. Seguidamente se presentan los contenidos de la propuesta, reflejando cómo puede ser aplicada en un plan de estudios concreto que es el de la Ingeniería en Informática de la Universidad Complutense de Madrid. Así mismo, en relación con la propuesta, abordaremos la cuestión del/de los lenguaje(s) de programación a utilizar.

2. Objetivos de la propuesta

Consideramos que el objetivo general de un bloque de materia de iniciación a la programación, como integrante del grueso de asignaturas dedicadas a la enseñanza de la programación con las que los alumnos se enfrentan a lo largo de la carrera, es contribuir en la capacitación de los alumnos para desarrollar programas de calidad. Entendemos por programas de calidad aquellos que cuentan con las siguientes características: claridad, corrección, extensibilidad, reusabilidad y eficiencia.

A nivel más concreto, la propuesta que aquí se presenta pretende alcanzar los siguientes subobjetivos:

- Dejar claro que la programación es un acto de resolución de problemas. Saber programar no es aprender uno o más lenguajes de programación concretos sino conocer y saber utilizar métodos que permitan la construcción sistemática de algoritmos.
- Presentar los conceptos de algoritmo y programa.
- Reconocer la necesidad de manipular datos cuando programamos y exponer el concepto de tipo de datos.
- Resaltar el papel fundamental que juega la abstracción en todo proceso de resolución de problemas y, en particular, en el proceso de desarrollo de software. Se deben exponer procesos de abstracción aplicados a diferentes entidades, presentando a lo largo de la materia la tendencia a incluir mecanismos de abstracción cada vez de más alto nivel, tendencia que acompaña a la evolución de la Programación como disciplina.
- Aportar elementos de programación básicos y metodologías de programación para la resolución de problemas.

  Se propone la metodología de descomposición funcional para el desarrollo tanto de algoritmos aislados como de aplicaciones completas de pequeño tamaño. Al plantearse la necesidad de tipos de datos más complejos se presentan la noción de Tipo Abstracto de Datos (TAD) y herramientas para su definición.

  La programación con TADs se introduce en el marco de la programación modular como una metodología de diseño que permite organizar los programas en torno a los datos que manipulan y no en base a las operaciones a realizar. Este diseño conducido por los datos se plantea como la técnica preferente a usar en el diseño de aplicaciones de mayor envergadura. Como ya se ha señalado, la metodología basada en descomposición

funcional y la conducida por los datos no son técnicas mutuamente excluyentes y en la presentación de los contenidos a los alumnos se debe hacer hincapié en ello.

- Introducir a los alumnos en la POO, presentándola como un paradigma que se apoya en la abstracción de datos e incluye mecanismos que permiten construir software extensible y reutilizable.
- Examinar los conceptos fundamentales de la OO.
- Dar a conocer los pasos esenciales que conlleva proporcionar una solución orientada a objetos a un cierto problema.
- Aplicar los conceptos presentados para solucionar problemas usando una aproximación orientada a objetos.
- Adquirir conocimiento acerca de (al menos) un lenguaje de programación. Dicho lenguaje debe contar con los mecanismos necesarios para dar soporte a las metodologías presentadas. Otra característica preferible, aunque no primordial en un primer lenguaje de programación, es que sea de proyección profesional.

3. La organización de los contenidos de la propuesta

La tabla 1 muestra los temas que componen el contenido de nuestra propuesta. Con la organización propuesta se espera realizar una transición relativamente "suave" de unas cuestiones a otras.

El primer tema presenta nociones básicas sobre Informática y programación, tales como la diferencia entre el hardware y el software, el concepto de algoritmo y el concepto de programa.

En el segundo tema hacemos que los alumnos caigan en la cuenta de que los algoritmos trabajan sobre datos y que es necesario representarlos y manipularlos. Se presentan el concepto de tipo de datos, las nociones de variable y constante, la instrucción de asignación. Se trata también la comunicación vía consola con el usuario.

El objetivo del tercer tema es la presentación de las construcciones alternativas e iterativas. Esas instrucciones se deben presentar como un paso en el camino hacia la abstracción, comparándolas con lo que ocurre en los lenguajes máquina. Se

1. Introducción a las computadoras y a la programación
2. Instrucciones y tipos elementales
3. Estructuras de control
4. Abstracción funcional
5. Introducción a la recursividad
6. Abstracción de datos: una introducción a los TADs
7. Colecciones
8. El camino hacia la orientación a objetos
9. Más sobre clases y objetos
10. Herencia
11. Polimorfismo y vinculación dinámica
12. Introducción al diseño orientado a objetos: aplicación de conceptos

Tabla 1. Temario de la propuesta.

esquematizan algoritmos iterativos de uso frecuente como son los recorridos y las búsquedas.

En la tendencia a incluir mecanismos de abstracción cada vez de más alto nivel, se presenta en el tema 4 la abstracción funcional. Se aborda el diseño de algoritmos aplicando diseño descendente.

La recursión se presenta como una técnica de resolución de problemas en el tema 5.

Llegados a este punto de la materia, se plantea a los alumnos la necesidad, a medida que aumenta la complejidad de los problemas abordados, de que el programador cree sus propios tipos de datos. Se presentan así en el tema 6 las ideas de abstracción de datos, TAD y las clases como la herramienta para implementar TADs –al permitir aquellas la encapsulación de datos y operaciones, hacer privados la representación de los datos y la implementación de las operaciones y proteger de accesos incontrolados. En este mismo tema se hará hincapié en que la metodología de descomposición conducida por los datos es la predominante en el diseño de aplicaciones de envergadura y que la metodología basada en descomposición funcional vista sigue siendo aplicable al desarrollo de las operaciones de los

nuevos tipos creados. Estas consideraciones son muy importantes para que los alumnos no tengan la sensación de que lo que han visto hasta ese momento deja de ser útil.

Muchos problemas requieren manejar cierto número de entidades de un determinado tipo. Si el número es considerable no se puede recurrir al uso de variables "independientes". La solución viene de la mano del uso de colecciones de entidades homogéneas. El tema 7 se dedica a estas cuestiones.

Tras haber presentado los conceptos de abstracción y encapsulación de datos y las clases como una herramienta para implementar los TADs, en el tema 8 se presenta la POO como una programación con TADs junto una serie de mecanismos que permiten alcanzar mejoras en la calidad del software. En este tema se ofrece una visión preliminar de esos mecanismos. Básicamente, se trata de transmitir a los alumnos de manera intuitiva las "bondades" de la OO.

El tema 9 aborda las cuestiones de creación, inicialización y destrucción de objetos. Para que los alumnos puedan llegar a apreciar mejor las características incorporadas en los diferentes lenguajes que soportan la OO, deberían exponerse algunos temas relacionados tales como la gestión de asignación de memoria y la diferencia existente entre las copias superficiales y las copias profundas.

Aprender a "programar orientado a objetos" implica comprender uno de los pilares de la filosofía que hay detrás: la realización de una tarea se materializa en la interacción de objetos, cada uno de los cuales tiene un estado y proporciona una serie de servicios (comportamiento). Llegados a este punto, los alumnos cuentan ya con los conocimientos necesarios para definir clases, crear objetos y conseguir que éstos se comuniquen entre sí mediante pasos de mensajes. El siguiente paso es establecer relaciones entre clases, siendo la herencia y la composición las dos típicas utilizadas en la organización de las clases. El tema 10 se dedica a la herencia.

Desde el punto de vista más pragmático, la herencia se presentará a los alumnos como un mecanismo que permite reutilizar código. Efectivamente, permite compartir código: las características comunes a un conjunto de clases se extraen y definen en una clase más general a la que luego especializan. Las descripciones de las clases resultado de esa especialización se complementan con la descripción de la superclase. De esta forma, disminuyen el tiempo y el esfuerzo necesarios para desarrollar las nuevas clases. Y el hecho de que el código de las clases esté organizado en una jerarquía permite darle una estructura al conjunto de módulos –haciéndolo más comprensible– así como facilitar los cambios, pues una modificación realizada en una clase se propaga automáticamente a sus subclases.

Aún con todo, estas ventajas no son las principales. Lo verdaderamente relevante aquí, y hay que hacérselo ver a los alumnos, es que la herencia, complementada con la vinculación dinámica y el polimorfismo (aspectos introducidos brevemente en el tema 8 y que serán tratados en detalle en el siguiente), permite construir descripciones a distintos niveles de abstracción.

En el tema 11 se revisa el concepto de polimorfismo brevemente descrito en el tema 8. Conviene relacionar, siempre que sea posible, lo que se vaya viendo con aspectos y ejemplos que ya hayan surgido. En este caso se hace ver a los alumnos que ya hemos encontrado entidades polimórficas (por ejemplo, los nombres de los métodos polimórficos debido a sobrecarga y redefinición). Aquí nos centramos, sin embargo, en las variables polimórficas.

Es interesante que los alumnos distingan entre el tipo estático y el dinámico de una variable y que se analice cómo influye la forma de reservar espacio de memoria en la existencia de variables polimórficas.

Asimismo, hay que resaltar el polimorfismo y las clases abstractas como aspectos clave en la reutilización.

Para finalizar el tema, se analizan las implicaciones que conlleva la existencia de variables polimórficas, presentando la distinción entre vinculación estática y dinámica.

El último tema del grueso de materia propuesto lo consideramos indispensable. En él se lleva a cabo una introducción al diseño OO: se presentan algunas consideraciones generales que conducen a buenos diseños –la alta cohesión, el bajo acoplamiento, la trazabilidad, etc.– y un esquema simple con las tareas básicas de todo diseño OO –detección de clases, determinación de relaciones entre clases, diseño de la coordinación de los objetos. Los conceptos expuestos a lo largo

del bloque de materia se aplican al desarrollo de soluciones orientadas a objetos a ciertos proyectos de programación. Consideramos que cualquier asignatura donde se introduzca la OO necesita de un tema como éste, en el que los alumnos pongan a prueba su comprensión del significado y la potencia de la OO.

Aparte de las correspondientes clases en las que abordar todos estos temas, sería altamente recomendable que los alumnos pudiesen hacer prácticas donde poner a prueba todos los conocimientos adquiridos.

3.1. La propuesta reflejada en el plan de estudios de la Ingeniería en Informática de la Universidad Complutense de Madrid

Esta sección refleja cómo una propuesta como la descrita se puede llevar al plan de estudios de la Ingeniería en Informática de nuestra universidad.

Las asignaturas directamente relacionadas con la programación en los dos primeros años de dicha titulación son:

- Primer curso: *Introducción a la Programación* y *Laboratorio de Programación I* son las asignaturas de toma de contacto con la programación. La primera es anual (9 créditos, 6 teóricos y 3 prácticos); la segunda (4,5 créditos, todos prácticos) se imparte en el segundo cuatrimestre como laboratorio de soporte a la primera.
- Segundo curso: *Estructura de Datos y de la Información* es una asignatura anual (15 créditos, 10 teóricos y 5 prácticos); *Programación Orientada a Objetos* (4,5 créditos, 3 teóricos y 1,5 prácticos), que se imparte en el primer cuatrimestre; *Laboratorio de Programación II* es anual (9 créditos, todos prácticos).

Las asignaturas que escogeríamos para impartir los contenidos presentados son *Introducción a la Programación* y *Programación Orientada a Objetos*. Grosso modo, el reparto de contenidos entre dichas asignaturas queda como sigue:

- Presentación de la metodología basada en descomposición funcional en el primer cuatrimestre de *Introducción a la Programación* como colofón a la presentación de la abstracción funcional. El segundo cuatrimestre de *Introducción a la Programación* se dedica a la metodología basada en descomposición conducida por los datos. Podríamos pues decir que en esta asignatura se seguiría un enfoque "basado en objetos" (*object-based*): programación con objetos (entidades o datos) como bloques principales de construcción, pero sin herencia. En definitiva, los temas del 1 al 7 quedarían recogidos en esta asignatura.
- Tomando como punto de partida los conocimientos adquiridos en la anterior asignatura, la asignatura *Programación Orientada a Objetos* se dedicaría a profundizar en la OO a lo largo de los temas 8 al 12. La descomposición funcional se seguirá presentando como técnica candidata a utilizar en el diseño del comportamiento de los objetos de una clase.

La tabla 2 recoge la estimación de la asignación de tiempo (medida en horas) para cada uno de los temas del programa propuesto.

Por lo que respecta a las asignaturas *Laboratorio de Programación I* y *Laboratorio de Programación II* pueden emplearse para la realización de prácticas en las que los alumnos utilicen los conocimientos adquiridos en *Introducción a la Programación* y *Programación Orientada a Objetos*.

4. La elección del lenguaje de programación

Dentro del planteamiento general de la propuesta que aquí se realiza queda una cuestión más que abordar: el/los lenguajes de programación a utilizar.

Hay que indicar que consideramos conveniente presentar los diferentes conceptos de la forma más desligada posible de un lenguaje de programación específico. Pero, a la vez, la ejemplificación con un lenguaje de programación concreto resulta interesante de cara a que los alumnos puedan más fácilmente poner en práctica sus conocimientos, ya sea en su propia casa o en el laboratorio (bien sea en horas regladas de las asignaturas de laboratorio que sirvan de complemento o en horas libres).

A la hora de elegir lenguaje, debemos tener en cuenta las necesidades de cada parte del grueso de materia. Para los siete primeros temas necesitamos

	Teoría	Práctica	Total
Introducción a la Programación			
1. Introducción a las computadoras y a la programación	3	–	3
2. Instrucciones y tipos elementales	5	3	8
3. Estructuras de control	10	6	16
4. Abstracción funcional	12	6	18
5. Introducción a la recursividad	4	3	7
6. Abstracción de datos: una introducción a los TADs	12	7	19
7. Colecciones	14	5	19
Programación Orientada a Objetos			
8. El camino hacia la orientación a objetos	2	–	2
9. Más sobre clases y objetos	6	2	8
10. Herencia	8	4	12
11. Polimorfismo y vinculación dinámica	5	2	7
12. Introducción al diseño orientado a objetos: aplicación de conceptos	9	7	16

Tabla 2. Asignación temporal (en horas) para los temas.

un lenguaje que soporte la abstracción funcional y la de datos. Para los cinco temas restantes necesitamos un lenguaje que soporte la orientación a objetos.

Podríamos optar por utilizar dos o más lenguajes. Sin embargo, hemos decidido utilizar un único lenguaje con el fin de evitar cambios de sintaxis dentro del bloque de materia. Actualmente en el primer curso de programación en la Facultad de Informática de la Universidad Complutense de Madrid se emplea Pascal. En segundo curso, en los últimos años se ha intentado unificar el lenguaje tanto para la asignatura *Programación Orientada a Objetos* como para *Laboratorio de Programación II*, siendo C++ el lenguaje utilizado. Los profesores de segundo curso somos conscientes de los problemas iniciales que tienen no pocos alumnos de segundo curso al cambiar de sintaxis.

En esta propuesta hemos elegido el lenguaje multiparadigma C++. Somos conscientes de que C++ es un lenguaje muy amplio y permisivo. Ante esta situación, estamos de cuerdo con lo que se indica en [1]:

> "*... many of the languages used for object-oriented programming in industry –particularly C++, but to a certain extent Java as well– involve significantly more detail complexity than classical languages. Unless instructors take special care to introduce the material in a way that limits this complexity, such details can easily overwhelm introductory students.*"

Conscientes de su amplitud, hay que indicar que no pretendemos en absoluto presentar el lenguaje en su totalidad. Para los cinco primeros temas utilizaremos un subconjunto de la parte procedimental de C++. Esta propuesta va en la línea de [9]. Hay que decir que, a este respecto, nos sentimos totalmente respaldados por los libros seleccionados como bibliografía básica. Estos libros son [5] y [16] para los siete primeros temas y [6] para los cinco restantes.

Por lo que respecta a la permisividad, que en ocasiones conduce a la obtención de código excesivamente críptico, estamos de acuerdo con lo que se indica en [5]: la solución viene de la mano de un estilo de programación directo, disciplinado

	Curso 1995-96	Curso 1996-97	Curso 1997-98	Curso 1998-99	Curso 1999-00
Pascal	36%	23%	6%	2%	5%
Ada	12%	18%	19%	7%	6%
C	17%	14%	11%	20%	19%
Scheme	2%	4%	4%	1%	---
C++	32%	39%	47%	50%	54%
Java	---	---	9%	22%	22%

Tabla 3. Primer lenguaje de programación usado en centros estadounidenses [10] [11] [12] [12] [14].

y libre de características intrincadas del lenguaje, de forma que los alumnos aprendan a utilizar C++ para producir código claro y legible.

El lenguaje de programación Java se va abriendo camino como primer lenguaje de programación (especialmente en centros estadounidenses está alcanzando niveles de uso nada despreciables). No hay más que echar un vistazo a los estudios que desde el curso 1995-96 son realizados acerca de la situación en departamentos que ofertan programas oficiales de Informática en Estados Unidos [10] [11] [12] [13] [14]–programas acreditados por CSAC/CSAB (*Computer Science Accreditation Commission of the Computing Sciences Accreditation Board*). La tabla 3 muestra los resultados que, relativos al primer lenguaje de programación usado, reflejan los citados estudios. Los resultados aparecen en forma de porcentaje de departamentos encuestados que utilizan cada uno de los lenguajes indicados.

Aunque hoy por hoy C++ es el lenguaje de programación más utilizado en universidades y *colleges* de Estados Unidos, vemos que Java que ha experimentado en tres años un crecimiento considerable[2].

Java es, indudablemente, un lenguaje muy interesante. Sin embargo no lo consideramos apropiado en nuestro marco por ciertos motivos. Por un lado, Java no es especialmente adecuado para presentar una metodología basada únicamente en descomposición funcional. Podemos usar Java siguiendo un estilo básicamente procedimental (usando métodos estáticos en una única clase). Pero los programas resultantes en Java serían, a nuestro juicio, un poco artificiosos. Además Java limita las posibilidades del flujo de datos a través de la interfaz de un bloque funcional cuando se trabaja exclusivamente con tipos primitivos (y no con objetos), que sería la situación en la que nos encontraríamos en los cinco primeros temas. Por otro lado, cuestiones nada triviales como son las excepciones aparecen desde el primer momento relacionadas con la entrada de datos. Finalmente, están las revisiones que probablemente aún sufran el lenguaje y sus bibliotecas, fruto de un lenguaje que aún no ha madurado lo suficiente.

El hecho de que Java sea un lenguaje reciente también se acusa en la bibliografía. No hemos conseguido encontrar un número significativo de libros de Java especialmente dedicados a un primer curso de programación. Por último, señalar un aspecto que nos ha llamado la atención en la bibliografía de Java. Entre los textos que hemos analizado existe una gran diferencia en la presentación de un tema que inevitablemente aparece desde las primeras clases y de forma recurrente: la entrada de datos. Entre los libros manejados nos hemos encontrado con los que utilizan las clases proporcionadas por Java [2], otros que "recubren" dichas clases con clases propias [8] y otros que utilizan interfaces gráficas desde el primer momento [4]. Esto hace que la resolución de un mismo ejemplo según el texto considerado tenga aspectos muy diferentes, algo

[2] Una visión más global del panorama actual en relación con el primer lenguaje de programación se puede obtener a partir de los informes Reid (accesibles desde la página personal del encargado de mantenimiento de los mismos, Frances L. Van Scoy, http://www.csee.wvu.edu/~vanscoy). Dichos informes recogen la situación de centros de todo el mundo.

que consideramos puede crear confusión en alumnos que (al menos teóricamente) se enfrentan por primera vez con la programación.

5. Conclusiones

Hemos presentado una propuesta de contenidos para un bloque de materia de iniciación a la programación que aúna la metodología procedimental junto con la orientación a objetos. La propuesta se ejemplifica llevándola a un plan de estudios concreto, el de la Ingeniería en Informática de la Universidad Complutense de Madrid.

Entre los diferentes aspectos tenidos en cuenta cabría destacar, por un lado, la conveniencia de desligar (en la medida de las posibilidades) la presentación de los conceptos de un lenguaje de programación concreto, y, por otro, la conveniencia de que se realice una transición "suave" entre contenidos, intentando que los alumnos vean las "bondades" de cada una de las metodologías y cómo se construyen unas sobre otras.

Agradecimientos: Este trabajo ha sido financiado parcialmente por el proyecto TIC 2000-0737-C03-01.

Referencias

[1] ACM/IEEE-CS Joint Task Force, *Computing Curricula 2001*, final draft (diciembre 2001).

[2] Arnow, D., Weiss, G., *Introducción a la programación con Java*, Addison-Wesley, 2001.

[3] Culwin, F., "Objects Imperatives!", *Procs. of the SIGCSE Technical Symposium on Computer Science Education*, 1999.

[4] Dale, N., Weems, C., Headington, M., *Introduction to Java and Software Design*, Jones and Bartlett, 2001.

[5] Dale, N., Weems, C., Headington, M., *Programming and Problem Solving with C++, 2nd Edition*, Jones and Bartlett, 2000.

[6] Lafore, R., *Object-Oriented Programming in C++*, SAMS Publishing, 1999.

[7] Lewis, J., "Myths about Object-Orientation and its Pedagogy", *Procs. of the SIGCSE Technical Symposium on Computer Science Education*, 2000.

[8] Liang, Y.D., *Introduction to Java*, Prentice Hall, 2001.

[9] Llopis Pascual, F., Pérez López, E., "C++-OO como lenguaje introductorio a la programación", *Actas de las VII Jornadas de Enseñanza Universitaria de la Informática*, 2001.

[10] McCauley, R.A., Manaris, B.Z., *Comprehensive Report on the 1999 Survey of Departments Offering CSAC/CSAB-Accredited Degree Programs*, Technical Report CoC/CS TR# 2000-9-1, Department of Computer Science, College of Charleston, octubre 2000.

[11] McCauley, R.A., Manaris, B.Z., *Comprehensive Report on the 1995 Survey of Departments Offering CSAC/CSAB-Accredited Degree Programs*, Technical Report TR-96-9-1, Center for Advanced Computer Studies, University of SouthWestern Louisiana, octubre 1996.

[12] McCauley, R.A., Manaris, B.Z., *Comprehensive Report on the 1996 Survey of Departments Offering CSAC/CSAB-Accredited Degree Programs*, Technical Report TR-97-9-1, Center for Advanced Computer Studies, University of SouthWestern Louisiana, octubre 1997.

[13] McCauley, R.A., Manaris, B.Z., Comprehensive Report on the 1997 Survey of Departments Offering CSAC/CSAB-Accredited Degree Programs, Technical Report TR-98-9-1, Center for Advanced Computer Studies, University of SouthWestern Louisiana, octubre 1998.

[14] McCauley, R.A., Manaris, B.Z., *Comprehensive Report on the 1998 Survey of Departments Offering CSAC/CSAB-Accredited Degree Programs*, Technical Report TR-99-9-1, Center for Advanced Computer Studies, University of SouthWestern Louisiana, octubre 1999.

[15] Marion, W., "CS1: What Should We Be Teaching?", *SIGCSE Bulletin*, 31, (4), 1999.

[16] Savitch, W., *Resolución de problemas con C++. El objetivo de la programación*, Prentice Hall, 2000.

Programación en Internet: La enseñanza de una nueva filosofía de desarrollo de aplicaciones informáticas

Sergio Luján-Mora, Jaume Aragonés Ferrero
Departamento de Lenguajes y Sistemas Informáticos
Universidad de Alicante
E-03080 Alicante
e-mail: {slujan,jaume}@dlsi.ua.es

Resumen

En este artículo presentamos los objetivos, contenidos y sistema de evaluación de *Programación en Internet*, asignatura optativa perteneciente a las carreras de Informática de la Universidad de Alicante. Esta asignatura, de reciente creación en el nuevo plan de estudios de Informática que se está implantando en Alicante, tiene como objetivo principal preparar ingenieros que sean capaces de analizar, diseñar e implementar aplicaciones en entornos Internet (Internet, intranet y extranet). Tras el primer año de docencia, exponemos los resultados y conclusiones, así como el planteamiento con el que hemos querido orientar la materia.

1. Introducción

La asignatura *Programación en Internet* (PI), pertenece al plan de estudios 2001 de las titulaciones de Ingeniería en Informática, Ingeniería Técnica en Informática de Sistemas e Ingeniería Técnica en Informática de Gestión de la Universidad de Alicante [3]. El nuevo plan de estudios 2001 comenzó a implantarse en el curso 2001-2002 y sustituye al anterior plan de estudios del año 1992.

En la Universidad de Alicante, esta asignatura la imparte el Departamento de Lenguajes y Sistemas Informáticos, adscrito a la Escuela Politécnica Superior de dicha Universidad.

Los profesores que hemos impartido esta asignatura durante su primer año de vida, hemos adquirido una experiencia en la materia gracias al trabajo desarrollado en diversos proyectos de programación de aplicaciones en entornos Internet/intranet en el Laboratorio Multimedia (mmlab) de la Universidad de Alicante y en la empresa privada. En esta asignatura hemos intentado transmitir las habilidades y conocimientos necesarios para poder llevar a cabo satisfactoriamente desarrollos informáticos en estos entornos. Hemos intentado transmitir no sólo los conocimientos o habilidades técnicas, sino también el "día a día" en el desarrollo de este tipo de proyectos.

Desde nuestro punto de vista, uno de los objetivos actuales de las carreras de Informática debería ser preparar ingenieros informáticos que puedan responder a la gran demanda actual (y futura) de profesionales especializados en temas relacionados con Internet. Como prueba de la gran demanda existente, queremos destacar dos datos. Por un lado, según la consultora International Data Corporation (IDC) [2], en España las empresas del sector tecnológico demandan un gran número de profesionales que hasta la fecha no ha sido posible satisfacer: en el año 2000 el déficit anual fue de unas 60.000 personas, en el año 2001 de 71.000 profesionales y se espera que pueda rebasar las 100.000 personas en el año 2003.

Por otro lado, el informe *e-España 2001* de la Fundación Retevisión [1], revela que sólo una de cada tres empresas cuenta con su propia página web, si bien rozan el 70% en el caso de las de alta tecnología y alcanza el 55% cuando se trata de compañías de servicios. Por tanto, existen muchas empresas que aún no tienen presencia en Internet y que demandarán profesionales cualificados en un futuro cercano.

2. Contexto del plan de estudios

El plan de estudios de Informática del año 2001 en la Universidad de Alicante consta de 364,5 créditos, de los cuales 72 créditos son optativos. La asignatura PI es optativa y se imparte en un cua-

trimestre, con una carga docente de 6 créditos, repartidos entre 3 de teoría y 3 de prácticas. La asignatura está vinculada al área de conocimiento de Lenguajes y Sistemas Informáticos La descripción oficial de la asignatura es (página 35.680 de [3]):

- Desarrollo y programación de sistemas de acceso a bases de datos de Internet.
- Planificación, diseño y administración de sitios Web.
- Migración de aplicaciones a entornos en Internet.
- Herramientas de desarrollo.
- Diseño y programación de elementos multimedia en Internet.

La asignatura PI no posee prerrequisitos, pero sí las siguientes recomendaciones oficiales (no es necesario haber aprobado las siguientes asignaturas antes de cursar PI, pero sí recomendable):

- Fundamentos de Programación I (1er curso).
- Fundamentos de Programación II (1er curso).
- Bases de Datos I (2º curso).

Además de las anteriores recomendaciones que figuran en el plan de estudios, los profesores de PI también recomendamos a los alumnos haber cursado antes alguna de las siguiente asignaturas obligatorias de la carrera de Ingeniería en Informática (indicamos porqué las consideramos necesarias):

- Programación orientada a objetos (2º curso). Porque la mayoría de las tecnologías empleadas en PI (ASP, JavaScript, Java, etc.) se basan en la orientación a objetos.
- Bases de datos II (3er curso). Porque profundiza en aspectos prácticos de las bases de datos (seguridad, bases de datos distribuidas, etc.) muy útiles en PI.
- Análisis y especificación de sistemas de información (4º curso). Tanto esta asignatura como la siguiente, desarrollan las habilidades necesarias para abordar correctamente el desarrollo de proyectos informáticos.
- Ingeniería del Software I (4º curso).

3. Asignaturas similares en otras universidades

Ante la gran demanda de profesionales con conocimientos sobre la programación en Internet, existen multitud de universidades españolas que ofertan cursos de programación en Internet o programación de páginas web mediante diferentes modalidades (curso de especialista, máster, título propio, etc.). Sólo como ejemplo incluimos las siguientes referencias:

- Construcción de portales de Internet en entorno dinámico con PHP, MySQL, ASP y FrontPage 2000 [7].
- Programación en Internet con HTML y Java [10].
- Especialista Universitario en Programación de Internet e Intranet [9].
- Programación Internet en Java [5].

Sin embargo, son pocas las universidades que posean en el plan de estudios oficial de alguna de las carreras de Informática una asignatura que trate la programación en Internet. En concreto, nosotros sólo conocemos tres universidades españolas: Granada, Oviedo y Sevilla.

En la Universidad de Granada, el Departamento de Lenguajes y Sistemas Informáticos oferta en su programa de tercer ciclo la asignatura *Programación de Aplicaciones para Intranet-Internet* [6], de 3 créditos. El temario de esta asignatura está dividido en cinco puntos. Los dos puntos principales tratan la programación del cliente (HTML, XML, CSS, etc.) y la programación del servidor (CGI). Sin embargo, no parece que se utilice alguna de las tecnologías actuales de programación de servidores (ASP, JSP, etc.).

En la Universidad de Oviedo, en el plan de estudios de Ingeniero Técnico en Informática de Sistemas en el campus de Gijón, existe la asignatura *Programación Internet/Intranet* [8]. La única información que poseemos sobre esta asignatura es la disponible en el BOE número 276 de 17 de noviembre de 2000:

- Créditos: 3 teóricos, 3 prácticos.
- Descripción: Lenguajes, técnicas y estándares de programación. Páginas web

dinámicas: componentes servidor y cliente.
- Vinculada al área de conocimiento: Ciencias de la Computación e Inteligencia Artificial.

Finalmente, en la Universidad de Sevilla, el Departamento de Lenguajes y Sistemas Informáticos oferta la asignatura *Programación en Internet* [11] como créditos de libre configuración para las carreras de Informática. Aunque nuestra asignatura y la de la Universidad de Sevilla posean el mismo nombre, sus contenidos son muy diferentes. Mientras que nosotros hemos intentado proporcionarle un contenido general, que se plasma en el uso de unas tecnologías particulares, la asignatura de Sevilla se centra exclusivamente en el uso del lenguaje Java (applets, servlets, CORBA, JDBC, etc.).

4. Descripción de la asignatura

A continuación se describen los objetivos, contenidos teóricos y prácticos, sistema de evaluación, recursos didácticos y bibliografía de la asignatura.

4.1. Objetivos

Desde nuestro punto de vista, uno de los objetivos actuales de las carreras de Informática debería ser formar ingenieros informáticos que puedan satisfacer la gran demanda actual (y futura) de profesionales especializados en temas relacionados con Internet (incluyendo intranet y extranet). El perfil de estos profesionales debería incluir conocimientos de:

- Diseño, instalación y administración de una intranet.
- Instalación, configuración y administración de servicios de Internet (web, correo electrónico, etc.).
- Instalación, configuración y administración de servicios de red (seguridad, cortafuegos, enrutadores, conmutadores, etc.).
- Evaluación de configuraciones informáticas orientadas a Internet.
- Análisis, diseño e implementación de aplicaciones web.

Los profesores de PI hemos planteado la asignatura de modo que satisfaga el objetivo de "análisis, diseño e implementación de aplicaciones web". En concreto, los objetivos de PI son:

- Que el alumno conozca las características principales de las tecnologías empleadas en el desarrollo de aplicaciones web.
- Que el alumno conozca la estructura y funcionamiento de una aplicación web.
- Que el alumno adquiera los conocimientos y habilidades necesarios para programar aplicaciones destinadas a ser usadas en entornos Internet.
- Que el alumno conozca los recursos específicos (hardware y software) necesarios para poner en producción aplicaciones web.

No sólo se trata de aprender habilidades técnicas, sino de dotar de conocimientos de fondo para formar profesionales flexibles capaces de trabajar con cualquier tecnología con una curva de aprendizaje mínima. Las tecnologías que se emplean en Internet están en continua evolución, por lo que no tiene sentido especializarse en una única tecnología, sino ofrecer una visión más amplia del estado actual de las tecnologías de desarrollo en Internet.

4.2. Contenidos teóricos

La parte teórica de PI se ha estructurado en tres módulos. En el primer módulo, se presenta al alumno la materia que compone el curso y se explican una serie de conceptos y conocimientos necesarios para trabajar en el entorno Internet, ya que programar en Internet no sólo requiere conocer un lenguaje de programación sino tener conocimientos de redes de computadores, arquitectura cliente/servidor, gráficos, bases de datos, etc. En el segundo módulo, se explican las técnicas de programación empleadas en la parte cliente de las aplicaciones web. Finalmente, en el tercer módulo se explican algunas de las tecnologías de programación de servidor más aceptadas en el mercado actual.

Las sesiones de teoría (2 horas cada una) de PI se basan en el siguiente temario:

<u>Módulo I: Introducción</u>

1. Introducción a la asignatura. (1 sesión)
2. Modelo cliente/servidor: dos y tres capas. (1 sesión)
3. Internet, intranet y extranet. (1 sesión)
    a. Introducción a Internet.
    b. Historia de Internet.
    c. Servicios básicos (web, correo electrónico, FTP, IRC, grupos de noticias, videoconferencia, audio y vídeo).
    d. ¿Qué es una aplicación web?
    e. Estructura de un sitio web: estructura física y lógica.

Módulo II: Programación de cliente

4. HTML. (2 sesiones)
    a. Estructura de una página, etiquetas, diseño de una página, consejos de estilo.
    b. Adquisición y acondicionamiento de imágenes digitales para la web.
    c. Técnicas de diseño: teoría del color, composición, tratamiento de imágenes.
5. Programación en el cliente: JavaScript. (2 sesiones)
    a. Sintaxis del lenguaje.
    b. Validación de formularios.
    c. Uso del modelo de objetos del navegador, interactividad.

Módulo III: Programación de servidor

6. Programación básica de servidor. (1 sesión)
    a. Common Gateway Interface (CGI).
    b. Server Side Includes (SSI).
    c. Internet Database Connector (IDC).
7. Active Server Pages (ASP). (3 sesiones)
    a. VBScript: sintaxis del lenguaje, objetos intrínsecos del lenguaje.
    b. Sintaxis y modelo de objetos, esquema básico de funcionamiento.
    c. Acceso a bases de datos a través de ODBC.
    d. Uso de variables de sesión y de aplicación. Global.asa.
8. Java Server Pages (JSP). (3 sesiones)
    a. Introducción al lenguaje Java.
    b. Sintaxis y modelo de objetos, esquema básico de funcionamiento.
    c. Acceso a bases de datos a través de JDBC y puente JDBC-ODBC.
9. Repaso. (1 sesión)

4.3. Contenidos prácticos

Al alumno se le plantea como trabajo final de la asignatura el desarrollo completo de una aplicación web. Además, se plantea una serie de prácticas a desarrollar durante el curso. El objetivo de estas prácticas es doble: por un lado, su realización coincide con las sesiones de teoría en las que se explican los conocimientos y habilidades necesarias para llevarlas a cabo, logrando un enfoque práctico de lo explicado en clase; por otro lado, estas prácticas se plantean como pasos intermedios en la realización del trabajo final, lo cual facilita el correcto desarrollo del mismo. Los alumnos pueden realizar las prácticas individualmente o por parejas.

Las seis prácticas planteadas a lo largo del curso son:

1. Configuración del servicio web Microsoft Personal Web Server. (1 sesión)
2. Creación de un conjunto de páginas HTML estáticas, mediante el uso exclusivo de etiquetas HTML. (3 sesiones)
3. Validación de formularios HTML mediante JavaScript. (3 sesiones)
4. Desarrollo y puesta en funcionamiento de un programa CGI programado en C. (2 sesiones)
5. Programación de una página sencilla con acceso a una base de datos con tecnología ASP. (2 sesiones)
6. Programación de una página sencilla con acceso a una base de datos con tecnología JSP. (2 sesiones)

A estas 13 sesiones, se añaden 2 sesiones más, hasta completar las 30 horas que corresponden a los 3 créditos prácticos. Estas dos sesiones se destinan al seguimiento por parte de los profesores de los trabajos finales de los alumnos. El objetivo

de este seguimiento es verificar la adecuación del trabajo realizado a los requisitos planteados en el enunciado.

En este primer año de la asignatura, como trabajo final obligatorio se propuso la programación de un sistema de "tienda virtual" dividida en tres zonas:

- Zona pública: catálogo de artículos agrupados por categorías y solicitud de más información sobre artículos de la tienda.
- Zona privada de los clientes: en ella cada cliente puede consultar el estado de los pedidos que ha realizado. Para acceder a esta zona el cliente se tiene que validar.
- Zona privada del administrador de la tienda: en ella el administrador puede gestionar el sistema de información por medio de mantenimientos de artículos, pedidos, clientes, etc. Para acceder a esta zona el administrador se tiene que validar.

Además, para aquellos alumnos que quisieran optar a una mayor calificación, se propuso una parte optativa que consistía en añadir los siguientes módulos:

- Zona pública: posibilidad de comprar desde la tienda ("carrito de la compra") y buscador de artículos en el catálogo.
- Zona privada del administrador: gestión de categorías y gestión de avisos y novedades de la tienda.

4.4. Evaluación

El método de evaluación de los alumnos abarca tanto los conocimientos teóricos como prácticos. Debido al carácter optativo de la asignatura y a que la mejor forma de asimilar los contenidos estudiados es llevándolos a la práctica, no influyen con el mismo peso en la nota final de la asignatura las partes de teoría y de prácticas:

- La parte teórica supone un 30% de la nota final. Se evalúa mediante un test de conocimientos.
- La parte práctica supone un 70% de la nota final. Los ejercicios prácticos desarrollados a lo largo del curso contarán un 30%, mientras que el trabajo final un 40%.

Además, los alumnos tienen la posibilidad de realizar trabajos optativos para aumentar su nota.

4.5. Recursos didácticos

Como no podía ser menos en una asignatura que se llama *Programación en Internet*, la página web de la asignatura es un medio de comunicación esencial. A través de ella, los alumnos han tenido a su disposición el temario completo de la asignatura antes de matricularse.

La página web se ha empleado durante el curso como medio de publicación de todos los materiales de la asignatura: transparencias, enunciados de prácticas, ejemplos, etc. También ha servido para publicar anuncios y avisos sobre la asignatura. En el futuro pensamos emplear el Campus Virtual de la Universidad de Alicante, ya que proporciona una serie de herramientas (tutorías, foros de discusión, alojamiento de materiales, etc.) que creemos que puede mejorar la calidad de nuestra docencia.

Para las clases de teoría, empleamos transparencias realizadas con Microsoft PowerPoint y ejemplos de código.

Por último, para realizar las prácticas, cada alumno o pareja de alumnos dispone de un ordenador con el software necesario para realizar las prácticas.

4.6. Bibliografía

Debido a lo novedoso de la materia tratada en esta asignatura y a que no existen asignaturas con contenidos similares en planes de estudio anteriores o actuales, no disponemos de libros docentes relacionados con la asignatura. Además, tampoco existen libros que traten la programación de aplicaciones web desde un punto de vista teórico. Por ello, todos los recursos bibliográficos recomendados son libros técnicos o manuales de referencia. A continuación citamos los que consideramos más interesantes (en la página web [4] de la asignatura existe una relación más extensa):

- Harvey M. Deitel, Paul J. Deitel, T. R. Nieto. *Internet & World Wide Web How to Program, 1/e*. Prentice Hall, 2000.
- Jesús Bobadilla Sancho. *Superutilidades para Webmasters*. Osborne McGraw-Hill, 1999.
- Jorge Serrano Pérez. *Programación con ASP 3*. Anaya Multimedia, 2001.
- Marty Hall. *Core Servlets and JavaServer Pages (JSP)*. Prentice Hall PTR/Sun Microsystems Press, 2000.

Además, como complemento a los anteriores libros, los profesores de PI hemos redactado los siguientes libros:

- Sergio Luján Mora. *Programación en Internet: clientes web*. Editorial Club Universitario, 2001
- Sergio Luján Mora. *Programación de servidores web con CGI, SSI e IDC*. Editorial Club Universitario, 2001.

5. Resultados

El nuevo plan de estudios de Informática ha comenzado a implantarse en la Universidad de Alicante durante el curso 2001-2002. En este primer año hemos tenido 218 alumnos matriculados en PI: 29 la han elegido como optativa y 189 como libre configuración. El número de matriculados como libre configuración es tan alto porque se ha ofertado como créditos de libre configuración a los alumnos de Informática del plan antiguo (plan de 1992).

De los 218 alumnos, 152 se apuntaron a prácticas por parejas, 48 de forma individual y 18 no se apuntaron (no han seguido la asignatura).

Respecto las calificaciones de los alumnos, al examen de la convocatoria de febrero se presentaron 180 personas. Los resultados obtenidos fueron 71 personas suspensas (39,4%), 47 aprobados (26,1%), 55 notables (30,6%) y 7 sobresalientes (3,9%). De las 71 personas suspendidas, 30 lo fueron porque se presentaron al examen sin haber presentado las prácticas, y por tanto ya sabían que estarían suspendidas (si se aprueba el examen, se conserva la nota obtenida para la convocatoria de septiembre). Si eliminamos ese grupo de gente, los porcentajes que se obtienen son: suspensos 27,3%, aprobados 31,3%, notables 36,7% y sobresalientes 4,7%.

En cuanto a la opinión de los alumnos sobre la asignatura, pese a no disponer de datos cuantitativos, los comentarios que nos han hecho llegar valoran muy positivamente los contenidos de la asignatura. Respecto al método de evaluación, la mayoría hubiera preferido que no hubiese examen de la parte teórica.

6. Conclusiones

Hemos expuesto el enfoque con que impartimos la asignatura *Programación en Internet* en la Universidad de Alicante. La asignatura sólo tiene un año de vida, así que es prematuro proporcionar conclusiones "definitivas".

Hemos comentado los objetivos, los contenidos teóricos y prácticos, la bibliografía, así como los resultados académicos logrados.

Los profesores de la asignatura creemos que la mejor forma de aprender esta asignatura es a través de la práctica. Esta idea coincide con lo que los alumnos esperan encontrar cuando se matriculan en PI.

Hemos intentado que esta asignatura no se convierta en el típico cursillo de creación de páginas web. Hemos pretendido transmitir una filosofía de programación nueva y dotar a los alumnos de habilidades y conocimientos de fondo para solucionar problemas en entornos reales. De este modo, el alumno será capaz de trabajar con cualquier tecnología con una curva mínima de aprendizaje, ya que dispone de una base sólida de conocimientos del entorno con el que se ha de enfrentar.

Una aspecto que creemos que se tiene que potenciar no sólo en esta asignatura, sino a lo largo de toda la carrera de Informática, es el desarrollo de la polivalencia y la flexibilidad de los alumnos, para que sean capaces de adaptarse a contextos tecnológicos en cambio permanente.

Evidentemente, aún queda mucho por desarrollar y validar. Estamos muy interesados en conocer experiencias similares de otras universidades españolas, con el fin de mejorar nuestro planteamiento de la asignatura. Por otro lado, el continuo cambio de las tecnologías empleadas en Internet, nos obligará a modificar año tras año el planteamiento de la asignatura.

Referencias

[1] Fundación Retevisión. *e-España 2001*. 2001. Internet: http://www.fundacionretevision.es/publi/publi_ee01.htm

[2] IBM. *IBM, nuevos empleos y nuevas tecnologías*. Nota de prensa, enero 2001. Internet: http://www5.ibm.com/es/press/notas/2001/marzo/cumbre.html

[3] Resolución de 5 de septiembre de 2001, de la Universidad de Alicante, relativa al plan de estudios conducente a la obtención del título de Ingeniero en Informática. BOE número 230 (25/09/2001).

[4] Universidad de Alicante. *Programación en Internet*. Internet: http://www.dlsi.ua.es/asignaturas/pi/

[5] Universidad Carlos III. *Programación Internet en Java*. Internet: http://www.it.uc3m.es/curs/java97.html

[6] Universidad de Granada. *Programación de Aplicaciones para Intranet-Internet*. Internet: http://www-lsi.ugr.es/doctorado/jguirao.shtml

[7] Universidad de Murcia. *Construcción de portales de Internet en entorno dinámico con PHP, MySQL, ASP y FrontPage 2000*. Internet: http://www.um.es/siu/congre/php/formacion.html

[8] Universidad de Oviedo. Programación Internet/Intranet. Internet: http://www.uniovi.es/Vicerrectorados/Estudiantes/Estudios/Carreras/INGENIEROTECNICOENINFORMATICADESISTEMAS_Gijon.html

[9] Universidad Politécnica de Valencia. *Especialista Universitario en Programación de Internet e Intranet*. Internet: http://www.cfp.upv.es

[10] Universidad Rey Juan Carlos. *Programación en Internet con HTML y Java*. Internet: http://www.urjc.es/eventos/cursos/ProgramacionInternet.html

[11] Universidad de Sevilla. *Programación en Internet*. Internet: http://www.lsi.us.es/docencia/asignaturas/pro_inte.html

Utilización de prácticas con gráficos 3D animados en la enseñanza de la programación orientada a objetos

Javier Macías, José María Gutiérrez, José Ramón Hilera, José Antonio Gutiérrez
Dpto. Ciencias de la Computación
Universidad de Alcalá
28871 Alcalá de Henares
e-mail: {javier.macias, josem.gutierrez, jose.hilera, jantonio.gutierrez}@uah.es

Resumen

Se presenta una experiencia real basada en la utilización de prácticas con gráficos animados en tres dimensiones en la enseñanza de la programación orientada a objetos, indicando los resultados obtenidos tanto en relación al aprendizaje como en la motivación de los alumnos.

1. Introducción

Es evidente la importancia que la programación orientada a objetos ha adquirido tanto en la educación universitaria como en la propia industria [1]. Sin embargo, los conceptos clave de la programación y el diseño de este paradigma, como clase, objeto, atributo, método, encapsulación, herencia, polimorfismo, etc. [4] no son fáciles de entender recurriendo únicamente a la explicación teórica.

Por otro lado, los alumnos de programación sufren problemas de falta de motivación [7]. Una de las razones de esta falta de motivación es achacable a las propias prácticas de programación utilizadas en su enseñanza. Éste es el caso de las prácticas basadas en ejemplos poco realistas o muy forzados, o prácticas demasiado complejas, o aquellas que consisten en una entrada simple seguida de un proceso tras el cual se obtiene un triste número en la pantalla, o, por citar un último caso, prácticas cerradas de difícil ampliación por parte del alumno.

Es curioso observar que cuando se utilizan lenguajes como Pascal o Modula en las prácticas, los alumnos culpan al lenguaje de defectos atribuibles a las propias prácticas, tachándolos de obsoletos e incluso de inútiles en lo relativo a su aprendizaje para la vida real, lo cual además crea en ellos un círculo vicioso desmotivador.

2. Objetivos

Dada la importancia de las prácticas en la enseñanza de la programación en general y de la programación orientada a objetos en particular, y considerando a su vez la necesidad de motivar a los alumnos en su aprendizaje, se buscaron unas prácticas que:

- Cubrieran una parte significativa de los conceptos básicos del diseño y la programación orientada a objetos.
- Fueran suficientemente cortas de forma que pudieran ser realizadas sin requerir grandes esfuerzos por parte del alumno, pero que a su vez resultaran vistosas y estimulantes.
- Fuera posible su ampliación de una forma fácil, si el alumno así lo deseaba.
- Se pudieran realizar utilizando el lenguaje conocido por los alumnos (en nuestro caso concreto, Turbo Pascal®) sin recurrir a artificios, aunque deberían ser fácilmente adaptables a cualquier otro lenguaje que soporte la orientación a objetos, como C++ o Java.

Una vez analizadas varias posibilidades de forma que se cubrieran los objetivos perseguidos, nos decantamos por diseñar prácticas fundamentadas en la utilización de gráficos en tres dimensiones y, en concreto, de figuras poliédricas, tal y como se desarrolla a continuación.

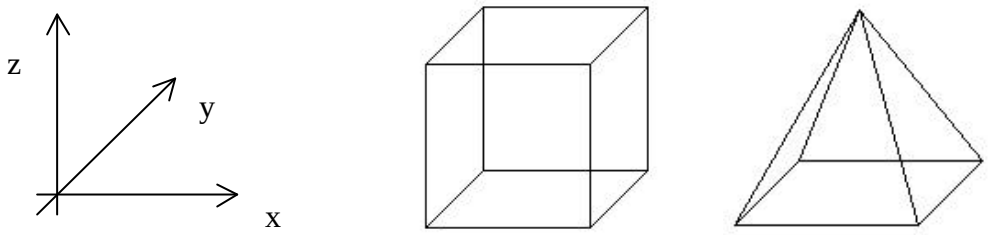


Figura 1. Ejes en perspectiva caballera y apariencia del cubo y pirámide solicitados en la primera práctica

3. Las prácticas

Las prácticas construidas al efecto han sido dos. A continuación se resumen ambas brevemente, así como se indican algunas cuestiones generales que se tuvieron en cuenta en las mismas.

La primera práctica se denomina "Figuras geométricas tridimensionales" [6]. Los alumnos deben crear un cubo y una pirámide cuadrangular (Figura 1) que giren en la pantalla sobre el eje z.

Antes de explicar a los alumnos el diseño recomendado, se les pidió que pensaran sobre cómo desarrollarían ellos la práctica utilizando programación estructurada. En general, ven mucha dificultad, lo cual hace que luego les resulte más impresionante la facilidad con la que se consigue utilizando un buen diseño y programación orientados a objetos.

El tiempo asignado, incluyendo tanto el destinado para su explicación como el utilizado por los alumnos para su desarrollo, fue de dos sesiones de dos horas de duración cada una.

La segunda práctica se denomina "Sistema solar de figuras geométricas" [8]. Los alumnos deben colocar una figura geométrica en el centro de coordenadas y distribuir otras figuras a su alrededor. Seguidamente, deben animar el sistema creado de forma parecida a la manera en que giran los planetas alrededor del Sol (Figura 2).

El tiempo total destinado a esta práctica fue de una sesión de dos horas.

Para grupos formados por alumnos con poca base en programación orientada a objetos o que nunca habían llegado a escribir y probar un programa orientado a objetos en el ordenador, se añadió una práctica previa introductoria, con una duración de una sesión de dos horas. La práctica consistía en representar varios puntos (objetos) en la pantalla que el usuario podía mover utilizando para ello el teclado [5].

En esta práctica se explica detalladamente y paso a paso todo el proceso de diseño, obteniendo una descomposición modular consistente en un programa principal y una unidad donde se definen dos clases. La primera de las clases representa una coordenada de la pantalla, y a partir de ésta se define un descendiente (utilizando la herencia) que representa un punto de la pantalla.

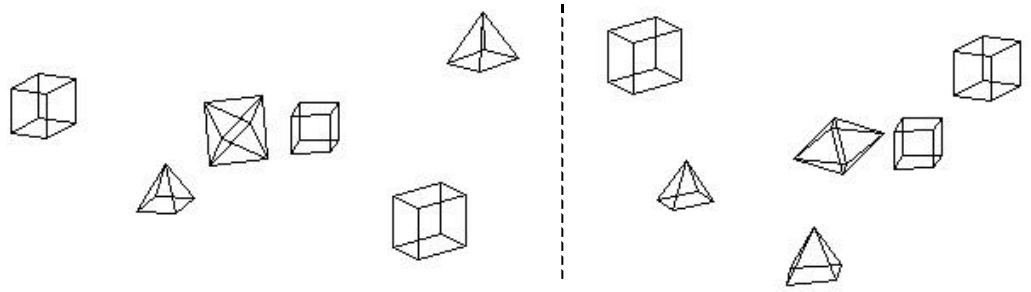


Figura 2. Dos instantáneas de un ejemplo de la segunda práctica

La práctica introductoria permite que los alumnos se familiaricen, entre otros, con los conceptos de clase, objeto y encapsulamiento, siendo también un primer acercamiento a la herencia. Además, al tener que utilizar gráficos, les ayuda a afrontar con más confianza las prácticas 3D posteriores.

Es de destacar que las prácticas aquí presentadas se destinaron fundamentalmente para la enseñanza de programación, por lo que se proporcionaron las ecuaciones necesarias para realizar las proyecciones sobre la pantalla, así como una unidad donde se definía la clase punto, debido a que esta clase debe contener métodos de traslación y rotación, para lo cual es necesario tener ciertos conocimientos de geometría computacional [2]. Si los alumnos poseen estos conocimientos, pueden desarrollar partiendo de cero toda la práctica, pero entonces sería conveniente destinar algo más de tiempo.

Además, para que el alumno concentrara su atención en la programación orientada a objetos y no se perdiera en detalles auxiliares, se le proporcionaron las rutinas para inicializar y finalizar el modo gráfico, los nombres y sintaxis de las funciones necesarias para dibujar líneas (y puntos, en el caso de la práctica introductoria) en la pantalla, así como vértices predefinidos que le facilitaran la labor de definir tanto el cubo como la pirámide.

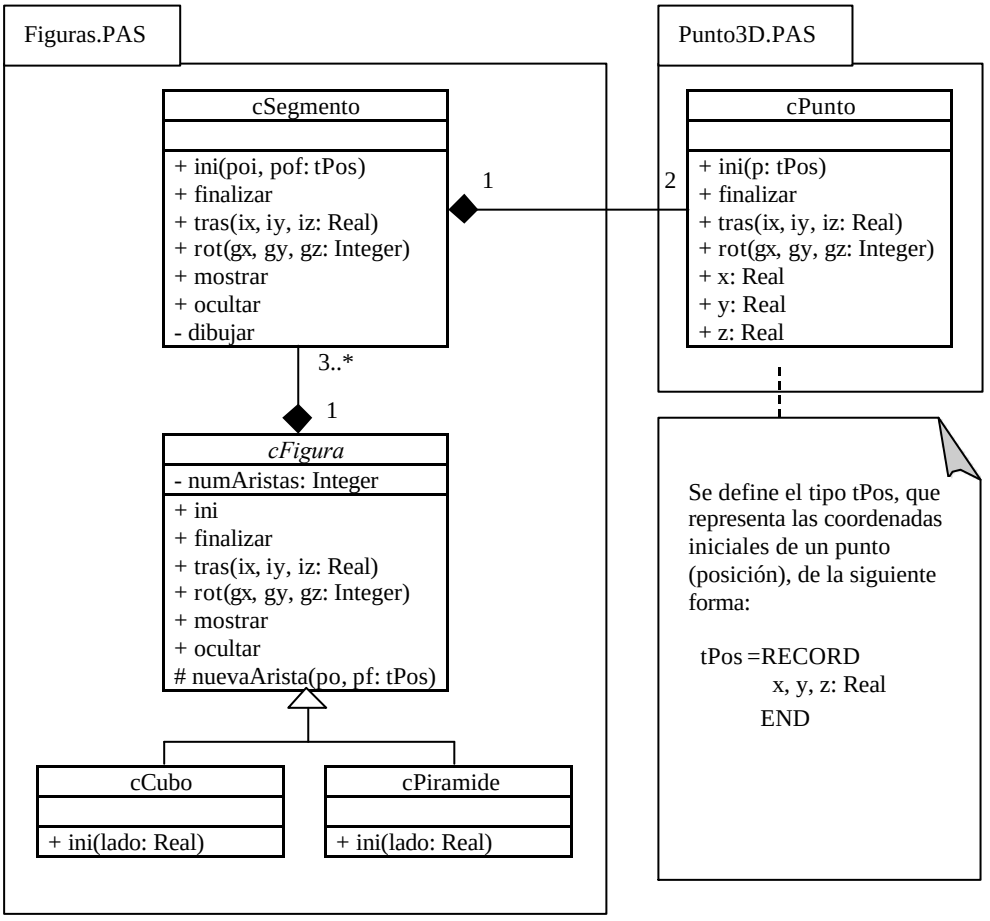


Figura 3. Diagrama simplificado de clases de la primera práctica

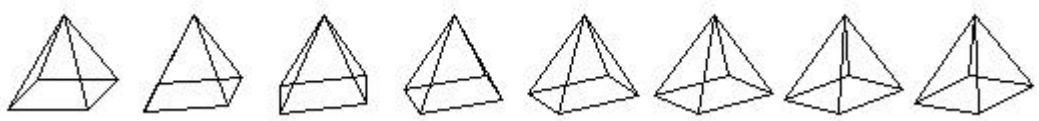


Figura 4. Instantáneas de la secuencia de giro de una pirámide

Por último, indicar que estas prácticas han sido utilizadas en el laboratorio de la asignatura "Estuctura de Datos" de las titulaciones I.T. en Informática de Sistemas e I.T. en Informática de Gestión de la Universidad de Alcalá desde el curso 1999/00. También han sido utilizadas en la asignatura "Técnicas de Programación Estructurada", impartida a los Sargentos Alumnos del Cuerpo de Especialistas Escala de Suboficiales, en su segundo curso de Enseñanza Militar de Formación, en la Escuela de Técnicas Aeronáuticas (ESTAER) en la Base Aérea de Torrejón (Ejército del Aire).

3.1. Primera práctica

El diseño de la primera práctica se puede acometer fácilmente empleando una estrategia de arriba a abajo (*top-down*), esto es, partiendo de un cubo o una pirámide se va descomponiendo el problema hasta llegar al punto, o empleando una estrategia de abajo a arriba (*bottom-up*), esto es, partiendo del punto, vamos obteniendo elementos más complejos hasta llegar a formar un cubo o una pirámide.

Cualquiera que sea la estrategia empleada (podría usarse una y luego la otra para que los alumnos aprecien ambos enfoques, o incluso un único enfoque mixto), es necesario que el alumno entienda que una figura (poliédrica) no es sino un conjunto de segmentos, y que un segmento se puede definir por sus puntos extremos. Nótese que se puede asociar inmediatamente el concepto de objeto (y, por tanto, de clase) a las figuras, segmentos y puntos.

Por otro lado, rotar una figura consiste en rotar todos sus segmentos. Y para rotar un segmento, basta con rotar sus puntos extremos y luego trazar una línea entre ambos. La traslación de una figura se hace de forma análoga. Como se puede apreciar, esto no es más que una aplicación del mecanismo de abstracción, que es fundamental en el diseño de sistemas complejos, hecho que se debe recalcar a los alumnos.

También es importante que el alumno entienda la utilidad de definir una clase abstracta figura, de la cual desciendan todos los poliedros. Esto además le permite ver lo sencillo que resulta añadir nuevas figuras. Para que los alumnos puedan entender fácilmente el diseño (Figura 3), se utiliza UML [9]. Nótese que en el diseño aparecen, entre otros, los conceptos de encapsulación y las relaciones de herencia y composición.

En el programa principal, el alumno debe instanciar un objeto cubo y otro pirámide. También definirá una función que reciba como parámetro una figura y muestre en la pantalla una animación consistente en rotar grado a grado dicha figura hasta completar 360º. Esta función se utilizará para introducir el concepto de polimor-

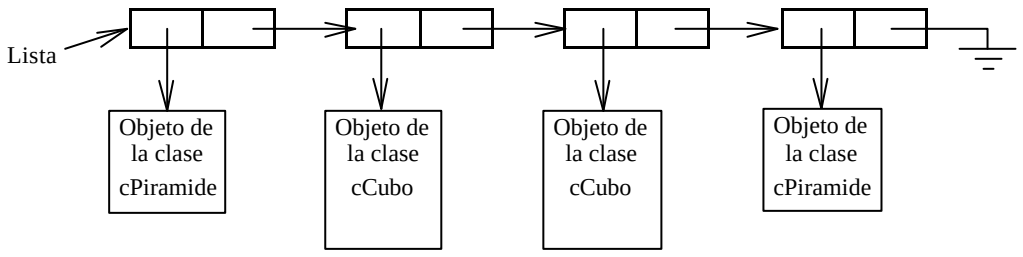


Figura 5. Ejemplo de lista de figuras

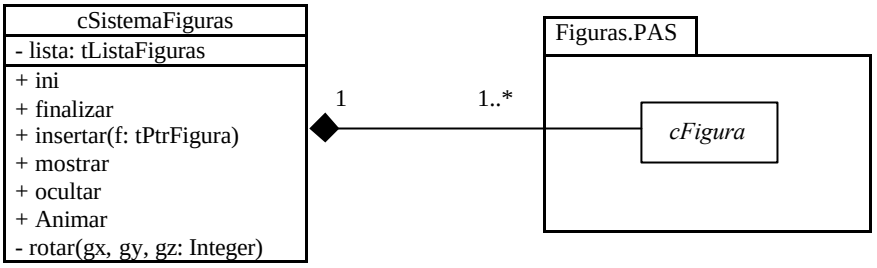


Figura 6. Diagrama simplificado de clases de la segunda práctica

fismo, ya que el alumno podrá comprobar que puede utilizar la misma función tanto para animar un cubo como una pirámide (Figura 4).

3.2. Segunda práctica

La práctica segunda está pensada para que el alumno se familiarice con el uso de punteros a objetos, para lo cual se utiliza una lista dinámica en la que se guardarán los objetos que formen el "sistema solar de figuras" (Figura 5). El diseño de esta práctica es muy sencillo (Figura 6), y en él se utiliza la unidad figuras creada en la práctica anterior.

Nótese que en esta práctica aparece de nuevo el polimorfismo, ya que la lista permite almacenar todo tipo de figuras.

4. Opinión de los alumnos

Es destacable la buena acogida que las prácticas aquí presentadas han venido teniendo estos años por parte de los alumnos. Una anécdota: el primer año de puesta en marcha de las mismas, alumnos del curso anterior, con la asignatura aprobada, mostraron su interés por las nuevas prácticas. Otra curiosidad: recientemente, se preguntó sobre las prácticas a alumnos que las habían realizado hace dos cursos, resultando grato el comprobar que todavía las recordaban y las guardaban.

Algunas de las opiniones expresadas por los alumnos sobre las prácticas han sido las siguientes:

- Muy interesantes y entretenidas.
- Llamativas: las mostraban a amigos y familiares.
- Pedagógicas y útiles. Les ayudaron a entender el diseño orientado a objetos, las ventajas de este tipo de programación, la diferencia entre clase y objeto, la herencia y otros conceptos que les han resultado muy útiles en asignaturas posteriores.
- Divertidas, ya que se podían ampliar fácilmente para, por ejemplo, construir nuevas figuras.
- Fomentadoras de la imaginación.

Por otro lado, señalar que resultó ligeramente más fácil entender las prácticas a los alumnos que habían visto dibujo técnico en bachillerato, aunque el resto de alumnos indicó que no tuvo especiales problemas para su realización una vez explicado, fundamentalmente, el concepto de proyección.

5. Ampliaciones de los alumnos

Es significativo destacar, como muestra de la motivación que estas prácticas produjeron entre los alumnos, algunas de las ampliaciones que éstos realizaron de forma voluntaria:

- Inclusión de otras figuras geométricas, algunas de ellas compuestas de un gran número de aristas, como las mostradas en la Figura 7.
- Adición de colores a las figuras.
- Representación en pantalla utilizando perspectiva isométrica, en vez de usar las ecuaciones dadas para perspectiva caballera.
- Interactividad mediante el teclado. Permitían al usuario definir el eje de giro, desplazar figuras por la pantalla usando los cursores,

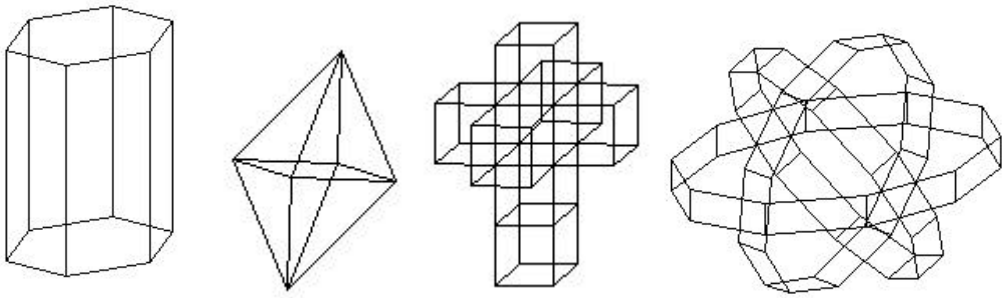


Figura 7. Ejemplos de figuras realizadas por los alumnos

seleccionar las posiciones iniciales y figuras a mostrar en la práctica del sistema solar, etc.

- Adición de un procedimiento (*wait retrace*) para evitar el parpadeo en la visualización de gráficos debido al refresco de la pantalla.
- Giro simultáneo de la figura sobre varios ejes.

6. Conclusión

La utilización de prácticas con gráficos 3D animados han resultado satisfactorias en la enseñanza de la programación orientada a objetos, ya que han permitido introducir de una forma natural los conceptos más importantes de este paradigma y a su vez han resultado estimulantes para los alumnos.

Es importante resaltar que es un factor crítico de éxito el que las prácticas sean animadas, ya que un gráfico estático presentado en la pantalla resulta mucho menos impresionante. También se debe destacar la importancia de utilizar UML o algún otro lenguaje gráfico para modelar el diseño, de forma que éste sea fácilmente entendible por los alumnos.

7. Futuras líneas de trabajo

Tras el éxito obtenido por estas prácticas entre los alumnos, se ha empezado a trabajar en una nueva práctica continuadora del espíritu aquí presentado basada en la visión estereoscópica, empleando para ello estereogramas de puntos al azar [3] en la pantalla. Para experimentar la sensación de tridimensionalidad sólo es necesario el uso de gafas con filtros rojo y verde, de fácil construcción, por otra parte, por los propios alumnos.

Las pruebas iniciales realizadas hasta ahora están resultando alentadoras. Conviene señalar que en su diseño se está utilizando una de las clases definidas en las prácticas de figuras, lo que permitirá, si en un futuro se usaran todas ellas conjuntamente, proporcionar a los alumnos un claro ejemplo de reutilización.

Referencias

[1] ACM/IEEE-CS. *Computer Curricula 2001*. Final report, 2001.

[2] Gisela Bielig-Schulz y Christoph Schulz. *3D graphics in Pascal*. Wiley, 1990.

[3] John P. Frisby. *Del ojo a la visión*. Alianza, 1987.

[4] Bertrand Meyer. *Construcción de software orientado a objetos*. Prentice Hall, 1999.

[5] Práctica introductoria: http://www.cc.uah.es /jmacias/jenui2002/Introductoria/

[6] Primera práctica: http://www.cc.uah.es /jmacias/jenui2002/Primera/

[7] Maria Salomó y otros. *Iniciativas para motivar a los alumnos de programación*. En las actas de las VII Jornadas de Enseñanza Universitaria de la Informática. José Miró (editor). Universitat de les Illes Balears, 2001.

[8] Segunda práctica: http://www.cc.uah.es /jmacias/jenui2002/Segunda/

[9] Unified Modeling Language: http://www.omg.org/uml/

Aprendizaje práctico de patrones de diseño en asignaturas de programación de nivel III

Raúl Marticorena Sánchez, Carlos López Nozal, César Ignacio García Osorio, Carlos Pardo Aguilar

Escuela Politécnica Superior, Departamento de Ingeniería Civil,
Área de Lenguajes y Sistemas Informáticos, Universidad de Burgos,
e-mail: {rmartico, clopezno, cgosorio, cpardo}@ubu.es

Resumen

Uno de los múltiples objetivos de las asignaturas de programación de nivel III[1] es la presentación del concepto *patrón de diseño* y la aplicación por parte de los alumnos, de un conjunto limitado de patrones de diseño. Debido a la amplia cantidad de contenidos que se puede añadir a este tipo de asignaturas, se debe establecer un plan estratégico para poder presentar adecuadamente los patrones y la aplicación práctica de los mismos, siempre bajo un intento de motivar al alumno y con un pequeño coste de aprendizaje. El presente trabajo establece una propuesta de definición de un plan estratégico, una aplicación de ese plan en una asignatura de programación de nivel III, y una evaluación por parte del alumnado de la presentación práctica de la asignatura relacionada con patrones de diseño.

1. Introducción

La última actualización del *Computing Curricula 2001* [1], incorpora en muchas de sus diferentes propuestas de cursos de programación la unidad *SE1.Software Design*[2], 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 obtener el objetivo de aprendizaje de *"selección y aplicación de los patrones de diseño adecuado en la construcción de una aplicación software"* [1].

El siguiente paso es como transmitir este nuevo concepto a los alumnos y en qué asignaturas del plan docente. El presente artículo justifica en una primera sección la necesidad de añadir los patrones en asignaturas de programación un nivel III. En su sección posterior presenta un modelo *de aprendizaje por experiencia* [5] y un caso práctico de aplicación de este modelo. Un ciclo de aprendizaje por experiencia comienza con una experiencia que es guiada sistemáticamente para garantizar que se realizan los enlaces entre lo observado y la teoría o la práctica del concepto sobre el que se esta experimentando.

El articulo termina mostrando una evaluación del modelo de aprendizaje por parte de los alumnos junto con unas conclusiones.

2. Patrones de diseño en asignaturas de programación de nivel III

Aunque el tema de patrones de diseño puede ser introducido en apartados dedicados a diseño en las asignaturas de ingeniería del software, la experiencia nos ha demostrado, que en estas asignaturas, existe una tendencia general en el alumnado de ver este nuevo concepto de patrón como algo teórico difícil de aplicar. Este hecho esta originado por la falta de la maduración necesaria para comprender todos los elementos necesarios en la descripción de un patrón, en concreto todo lo relacionado con un mayor

[1] Se entiende por programación de nivel III la última asignatura de programación impartida en la titulación técnica.

[2] SE significa el área de Software Engineering y 1 el número de unidad dentro del área.

conocimiento de un lenguaje de programación orientado a objetos. Esto dificulta la introducción completa de un patrón en asignaturas de ingeniería del software.

En la Tabla 1 se muestran los diferentes elementos necesarios para la descripción de un patrón según los autores [3] y [4]. En la tabla se han marcado en gris los elementos de las plantillas de descripción de patrones relacionados con la implementación. Para poder explicar estos elementos es necesario la utilización de algún lenguaje de programación orientado a objetos. Se justifica de esta forma la descripción detallada de estos elementos en las asignaturas de programación de nivel III.

Plantillas de GoF	**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 Plantillas de descripción de patrones de diseño

3. Propuesta del plan de aprendizaje de patrones en asignaturas de Programación Avanzada

3.1. Contexto especifico de la UBU[3]

Antes de matizar algunos aspectos concretos del plan de estudios de Ingeniería Técnica de Informática de Gestión en la Universidad de Burgos, comentar que desde el prisma puramente técnico únicamente se pretende presentar los patrones como herramienta aplicable en el desarrollo de un sistema informático. En la titulación de Ingeniería Técnica no se pretende hacer un recorrido completo de un catálogo de patrones. Se delega la profundización en estos aspectos a la titulación de Ingeniería.

Asignatura	Créditos	Cuatrimestre
Análisis e Ingeniería del Software	9 + 3	3º y 4º
Metodología de la Programación	3 + 3	4º
Programación Avanzada	3 + 3	5º

Tabla 2 Asignaturas relacionadas con la docencia de patrones en el plan de estudios de la UBU.

Con el objetivo de poder establecer unos prerrequisitos en la materialización de dicho modelo de aprendizaje, se va comentar brevemente las dependencias con asignaturas relacionas en el contexto particular del plan de estudios de Ingeniería Técnica en Informática de Gestión en la Universidad de Burgos (ver Tabla 2).

Las asignaturas de Ingeniería del Software y Metodología de Programación, ambas troncales, introducen paralelamente en el cuarto cuatrimestre los conceptos asociados a la orientación a objetos. Análisis e Ingeniería del Software desde la parcela del análisis y diseño, para ello utiliza el lenguaje de modelado UML. La asignatura de Metodología de Programación se centra más en un enfoque de implementación. Ambas hacen referencias casi al final del cuatrimestre a los patrones sin que los alumnos tengan tiempo para aplicar prácticamente los patrones de diseño.

La asignatura de Programación Avanzada o programación de nivel III de carácter optativo, se encuadra en una posición temporal perfecta (5º cuatrimestre) para poder madurar y aplicar los patrones de diseño pero también debe satisfacer otros objetivos:

- Mapeo de modelos de objetos. Java versus UML.
- Introducción a la programación orientada a eventos.
- Persistencia en Orientación a Objetos.
- Introducción a la programación concurrente.

Debido a los múltiples objetivos de la asignatura se necesita establecer un plan previo para que el alumno pueda aplicar patrones desde

[3] UBU Universidad de Burgos

un prisma de motivación y con el menor esfuerzo posible.

Es necesario establecer una coordinación entre las asignaturas de ingeniería del software y las asignaturas de programación para introducir completamente los conceptos de patrones de diseño en Ingeniería Técnica Informática. Esta coordinación se basa fundamentalmente en la elección de un mismo conjunto de patrones para poder hacer referencia a ellos en asignaturas que se impartan posteriormente en el tiempo.

A partir de este contexto específico se extraen las precondiciones necesarias que deben cumplir los alumnos para poder aplicar el plan:

- Tener claros los conceptos de orientación a objetos, y haberlos utilizado tanto a nivel de análisis y diseño como desde un punto de vista de programación con algún lenguaje concreto.
- Tener una idea intuitiva de patrón de diseño.

3.2. Propuesta del plan de aprendizaje de patrones de diseño

Típicamente la responsabilidad de la programación orientada a eventos propuesta por *Computing Curricula 2001* [1] suelen recaer en asignaturas de programación de nivel III. Generalmente se utilizan interfaces gráficas para introducir la programación orientada a eventos. Huelga decir que el alumno muestra un alto grado de satisfacción y motivación en este tipo de prácticas a pesar de la complejidad de las mismas. El plan se basa en poder utilizar esta motivación para preparar inconscientemente al alumno en la aplicación práctica del patrón y ver las mejoras que este aporta.

A continuación se enuncian las actividades y los artefactos[4] conseguidos en cada actividad para establecer el plan.

1. Elección del patrón o patrones que se quiere que el alumno aplique. Se obtiene como resultado las plantillas de los patrones a aplicar.
2. Definición de un contexto sencillo donde poder aplicar el patrón. Se obtiene el enunciado preeliminar de un problema que pueda contener el patrón.
3. Proponer un enunciado de práctica envolviendo el contexto y añadiendo un interface gráfico al mismo para practicar la programación orientada a eventos. Se obtiene el enunciado de la práctica a realizar por los alumnos.
4. Realización de la práctica propuesta por parte de los alumnos, sin que tengan una explicación previa del patrón.
5. Después de la explicación teórica del patrón elegido por parte del profesor, refactorizar la práctica utilizando el patrón elegido analizando las mejoras introducidas por el diseño.

Las actividades 1, 2 y 3 requieren un esfuerzo coordinado del personal docente encargado en la planificación de la asignatura.

El objetivo primordial de la actividad 4 es proporcionar al alumno la posibilidad de proponer su solución, sin conocer la solución dictada por el patrón. De esta forma el alumno descubre la gran cantidad de tiempo empleado para solucionar el problema.

La actividad 5 se aprovecha del gran esfuerzo realizado inconscientemente por los alumnos en la actividad 4 para conseguir una aplicación del patrón elegido y que vean los beneficios que aporta la aplicación del patrón.

3.3. Precedentes

La elaboración del plan nace de la experiencia obtenida en el curso 2000-2001. En dicho curso, se planteó en primer lugar una práctica en modo texto y posteriormente llevarla a cabo dotándola de una interface gráfica, pero condicionados a la reutilización de las clases iniciales.

Como resultado de la misma, se observó que el alumno capta rápidamente la necesidad de diseñar módulos independientes reutilizables y a su vez comprueba la utilidad de que dichas clases se puedan incorporar rápidamente en la construcción de la segunda práctica, sin modificaciones, y acelerando de forma inmediata el proceso de desarrollo.

Pese a este éxito parcial, se observó que los diseños divergían en varias soluciones, que no

[4] producto obtenido como consecuencia de una actividad

siendo del todo incorrectos, eran complejos e introducían problemas de implementación, no siempre fáciles de explicar por parte del docente. De hecho, la comunicación docente alumno se entorpecía debido al volumen de distintas soluciones, con la complejidad asociada a cada una y no siempre bien entendida por el propio alumno.

Ante esta situación en la que el alumno llega a la solución, pero no de la forma más adecuada, se plantea la necesidad de imponer algún tipo de estrategia que lleve a los alumnos hacia un diseño adecuado de la solución.

Dicho diseño no debe ser impuesto, sino razonado, a través de la idea de patrones, describiendo su necesidad y ventajas aportadas, a través del plan de aprendizaje.

3.4. Aplicación del plan

En este apartado se va a describir la aplicación del plan de aprendizaje propuesto en la asignatura de Programación Avanzada durante el primer cuatrimestre del curso 2001-2002.

Actividad 1: Elección del patrón. Patrón mediador.

Actividad 2: Definir un contexto sencillo donde poder aplicar el patrón. El patrón mediador puede aparecer en el diseño de sistemas (siguiendo la idea de construcción de sistemas en la orientación a objetos ya introducida en la asignatura de Metodología de la Programación [2]) cuando existen dependencias entre los diferentes módulos del sistema, necesitando tener referencias entrelazadas entre los objetos. Por tanto en la actividad 3 hay que establecer un diseño donde el acoplamiento y cohesión de los diferentes módulos sea el correcto, manteniendo todo el conjunto de propiedades inherentes a la programación orientada a objetos. En nuestra situación particular se buscará dependencias entre diferentes componentes gráficos.

Actividad 3: Proponer un enunciado de la práctica que contenga el patrón elegido. Para aumentar el grado de motivación de los alumnos se elige la realización de un juego donde esté contenido el patrón mediador en alguna de las pantallas de interacción con el usuario.

Enunciado propuesto: Caza del Barco

La práctica consiste en realizar un applet java que permita jugar a la caza del barco. El juego está pensado para que un jugador sobre un tablero intente hundir un barco que ocupa físicamente una casilla del mismo. La dificultad del juego estriba en que el jugador, no sabe donde esta el barco, y lo único que se le muestra es un radio de posibles casillas alrededor de su tirada en la que el barco puede estar localizado. Para aumentar la dificultad el barco se mueve aleatoriamente siguiendo los movimientos del rey en el juego del ajedrez (en un radio de 1 casilla puede posicionarse en cualquiera de ellas). La interfaz debe aportar al usuario una forma sencilla e intuitiva de jugar de tal forma que pueda visualizar:

- el tablero con el radio de posibles casillas.
- el radio de acción bajo el cual puede estar situado el barco
- el numero de tiradas
- opción de visualizar el barco (facilidad en el juego)

El juego debe ser configurable de tal forma que se puedan leer el número de casillas del tablero tanto horizontales como verticales y el numero de tiradas. El applet debe ser visible desde cualquier navegador con soporte de Java.

Se deja libertad al alumno para emplear todas las posibilidades gráficas de AWT para el diseño de la interfaz. La Figura 1 muestra la pantalla de interacción con el usuario del juego Caza Barco, configurado con 10 casillas por 10 casillas y 10 tiradas

Aclaraciones:

Para el cálculo del radio nos basamos en colorear todas aquellas casillas que están a igual o menor distancia que el barco respecto a la tirada. (distancia Manhattan) Considerando distancia como:

- Si la posición del barco es (x,y)
- Si la posición de la tirada es (x1,y1)
- Distancia = abs (x-x1) + abs (y -y1)

Se colorean todas aquellas casilla (i,j) donde: abs(i-x1) + abs(j-y1) <= Distancia.

No se colorean casillas fuera del tablero.

La Figura 1 muestra una instantánea de interacción del juego donde se observan distintos resultados ejecutando la opción de visualizar el barco activada:

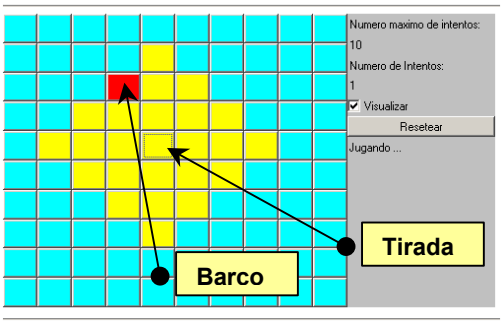


Figura 1 Instantánea de interacción del juego

En cada tirada el barco se mueve aleatoriamente con los siguientes parámetros

- en el eje x con valor +1, 0 y -1 (con igual probabilidad)
- en el eje y con valor +1, 0 y -1 (con igual probabilidad)

Por lo tanto se puede mover a cualquiera de las ocho casillas circundantes, lo cual aumenta la dificultad del juego, incluso se puede dar la posibilidad de no moverse si los dos desplazamientos son cero (ver Figura 2).

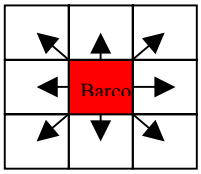


Figura 2 Posibles movimientos del barco

Sin entrar en detalle respecto a las especificaciones técnicas de la prácticas, por no afectar de manera directa al plan estratégico, se pasa a la siguiente fase.

Actividad 4: Realización por parte de los alumnos de la práctica propuesta. El diagrama de clases de la Figura 3 se corresponde con un mapeo estructural de la práctica propuesta por un grupo de alumnos.

Comentar que casi la totalidad de las prácticas funcionaba correctamente pero ningún diseño se adaptaba a la solución dictada por el patrón mediador a pesar de haber introducido este patrón en la asignatura de Análisis e Ingeniería del Software.

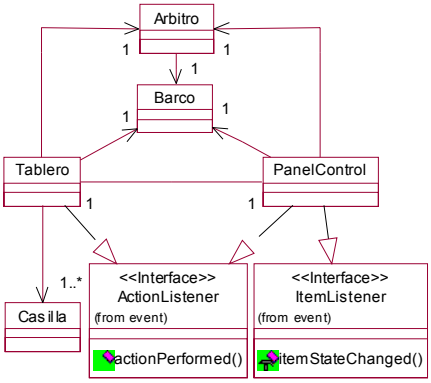


Figura 3 Artefacto actividad 4. Diagrama de clases

Los alumnos identifican claramente las abstracciones necesarias para construir software propuesto, pero se encuentran con dificultades a la hora de poder representar las dependencias existentes entre las abstracciones. El diagrama de clases de la Figura 3 presenta un mayor acoplamiento y una menor cohesión en sus clases que la solución propuesta por el patrón.

Después de evaluar las diferentes prácticas de los alumnos, se pudo observar que se encontraban en una situación idónea para que pudiesen ver los beneficios de la solución de aplicar el patrón mediator.

Actividad 5: Refactorizar la práctica aplicando el patrón elegido. El profesor propone la reestructuración de la práctica aplicando la solución del patrón (ver Figura 4). Este patrón presenta dos tipos de clase, las clases colegas (Tablero, Barco, PanelControl) cuyas instancias presentan dependencias entre ellas, y la clase mediadora (Mediador) cuya instancia coordina las dependencias entre las instancias de las clases *colegas*.

La clase mediador proporciona como parte de su interface métodos (registrarBarco, registarPanelControl, registrarTablero) que permiten a los objetos de las clases colegas registrarse, para que el objeto mediador pueda gobernar todas las dependencias entre todos los colegas registrados. De esta forma, cuando alguna instancia de las clases colegas sufre algún cambio de estado se lo notifica al mediador a través de un evento, y éste aplica la lógica de dependencias de estados.

En este diseño propuesto por el patrón se aprecia que se reduce el acoplamiento, y aumenta la conexión de las clases participantes ya que las clases colegas se liberan de la responsabilidad de controlar su dependencia con otras objetos de las clases colegas.

En el contexto específico de la práctica las dependencias existentes son:

- Dependencia entre tablero y el panel de control. Si se activa la casilla de verificación visualizar en el panel de control el tablero debe visualizar la última posición en que estuvo el barco.
- Dependencia entre tablero y posición del barco. Cada vez que el barco haga un movimiento se lo debe comunicar a tablero para poder visualizar el radio en el que se puede encontrar el barco.

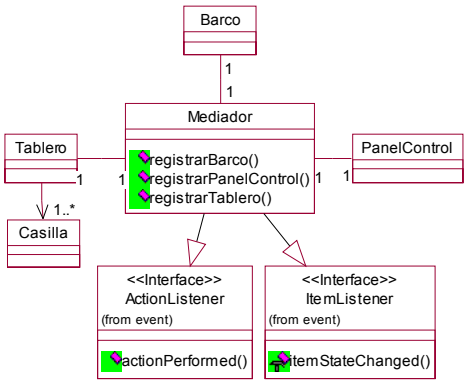


Figura 4 Artefacto actividad 5. Diagrama de clases.

3.5. Evaluación de resultados prácticos

Para poder evaluar los resultados obtenidos en la aplicación de este método se añadieron algunas preguntas en la encuesta de evaluación que se pasa a los alumnos al final del curso de la asignatura.

La Tabla 3 recoge los resultados de la encuesta sobre una muestra de 32 alumnos. En la columna de la izquierda se muestra el enunciado de las preguntas realizadas relativas a la evaluación de la asignatura que se corresponde de alguna forma a la evaluación del tema presentado. En el resto de columnas se indica el resultado de la evaluación por parte de los alumnos indicado en tanto por ciento. Comentar que en la evaluación existían dos tipos de preguntas, unas ponderables con valores de 1 a 5 (P1, P2, P3, P4, P5), el otro tipo únicamente poseía las respuestas si, no o sin contestar y un campo textual para indicar las razones (P6).

En las preguntas ponderables con un rango de valores se añade un campo donde se indica el valor medio obtenido de la encuesta.

Ya sea con los datos estadísticos de la Tabla 3 o con su representación gráfica mostrada en la Figura 5 se aprecia la buena aceptación por parte del alumnado de esta idea. Aun siendo conscientes de que algún alumno no acepta o simplemente no está convencido, de que sea adecuado introducir este temario y por otro lado, tampoco se ha conseguido un éxito pleno en conseguir la puntuación más alta. Pero como norma general, se observa que la mayoría se mueven en una puntuación de aprobado y notable a la utilización e inclusión de los patrones.

Uno de los datos más destacables de la encuesta son los resultados obtenidos en la pregunta 6, donde se deja al alumno con una gran motivación para que pueda aplicar patrones en su trabajo fin de carrera.

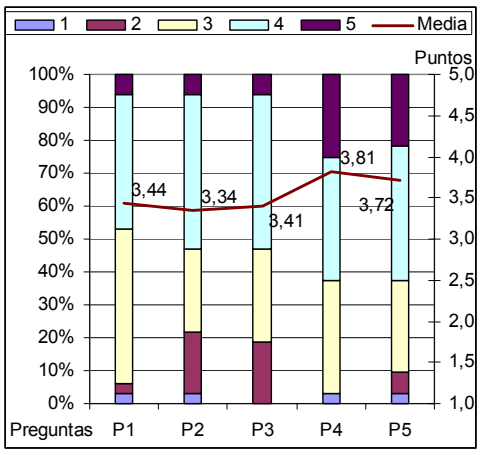


Figura 5 Gráfica de resultados de la evaluación

		1	2	3	4	5
Enunciado preguntas	Media	%	%	%	%	%
P1 Se ha incidido suficientemente en la aplicación práctica de la teoría	3,44	3,13	3,13	46,88	40,63	6,25
P2 Satisfacción del alumno con la parte práctica de la asignatura	3,34	3,13	18,75	25,00	46,88	6,25
P3 Grado de satisfacción con los ejercicios de prácticas	3,41	0,00	18,75	28,13	46,88	6,25
P4 Consideras interesante la explicación de patrones de diseño	3,81	3,13	0,00	34,38	37,50	25,00
P5 Ves útil la aplicación de patrones de diseño en prácticas	3,72	3,13	6,25	28,13	40,63	21,88
		% Si	% No	% NS		
P6 Intentarías utilizar patrones en tu trabajo final de carrera SI o, NO o no sabes ¿Por qué?		87,50	3,13	9,38		

Tabla 3 Resultados de la evaluación docente

4. Conclusiones

Los resultados obtenidos en las encuestas realizadas, confirman la consecución de los objetivos planteados en el plan estratégico. No sólo por parte de los docentes, existe una seguridad en la adecuación de los patrones de diseño como parte del temario de la asignatura de Programación Avanzada, sino que por parte de los alumnos, se logra transmitir y concienciar de dicha idea.

Los alumnos de ingeniería técnica no ven el concepto de patrones de diseño como algo puramente teórico difícil de aplicar. Elevan el nivel de abstracción, facilitando de esta forma la comunicación con el profesor a la hora de plantear y resolver nuevos problemas en los que se vislumbra la aplicación de patrones ya explicados. Se aprecian en la práctica los beneficios de la aplicación de patrones tanto a nivel de diseño como de implementación:

- Reducción del tiempo de desarrollo
- Mejora de documentación
- Aumenta la calidad del software

Además, se ha podido constatar el como llevar estas ideas más allá, en la realización de Proyectos Final de Carrera, facilita la comunicación del docente y alumno. Se maneja un vocabulario común y las soluciones planteadas por uno y realizadas por otro, son comprendidas y discutidas en un corto periodo de tiempo.

Experiencias prácticas en esta línea se confirman en la utilización de patrones observador y mediador en el caso de proyectos con una fuerte carga de interfaz gráfica. Igualmente el uso de patrones como composite, de cara a estructuras complejas, en las que las composición es clave o patrones mucho más sencillos como singleton, que garantiza unas propiedades muy concretas, llevan a que los tiempos de realización se acorten, la documentación sea correcta y quede una buena base para que posteriores proyectos usen dichos documentos como base de trabajo. De hecho, el que las soluciones adoptadas y el vocabulario empleado en la resolución del proyecto sea común, anima aquellos alumno que han cursado la asignatura de Programación Avanzada a documentarse a partir del trabajo de sus compañeros, aumentando la reutilización en el proceso de producción de software.

Referencias

[1] "Computing Curricula 2001. Computer Science" The Joint Task Force on Computing Curricula IEEE Computer Society. Association for Computing Machinery. Ed IEEE-CS y ACM.

[2] Bertrand Meyer, "Object Oriented Software Construction", 2nd Edition. Ed Prentice Hall. 1997.

[3] Erich Gamma, R. Helm, R. Johnson, Vlissides J. " Design Patterns. Elements of Reusable Object-Oriented Software". Ed Addison Wesley 1995.

[4] Mark Grand. " Patterns in Java. Volumen 1". Ed. Wiley computer publishing. 1998.

[5] Miguel Rebollo. " Aprendizaje activo en el aula". Actas JENUI 2001

Robótica e informática industrial

Propuesta metodológica para la impartición de Informática Industrial en la titulación de Ingeniería en Automática y Electrónica en el marco del proyecto EUROPA

Juan Vte. Capella, Rafael Ors
Dept. de Informática de Sistemas y Computadores
Universidad Politécnica de Valencia
46022 Valencia
e-mail: {jcapella, rors}@disca.upv.es

Resumen

La asignatura Informática Industrial se imparte en el último curso de la titulación de segundo ciclo Ingeniero en Automática y Electrónica Industrial. Se trata de una asignatura a la que los alumnos llegan con unos conocimientos importantes de arquitectura de computadores adquiridos en asignaturas previas de primer ciclo. Por ello, los objetivos de dicha asignatura se pueden centrar más en que los alumnos aprendan a aprender y desarrollen las habilidades que necesitarán en su futuro profesional.

En esta línea, la ponencia presenta un nuevo enfoque y metodología que van a ser adoptados para impartir dicha asignatura, y con los cuales se pretende alcanzar los objetivos anteriores. Para ello se propone el empleo de unas metodologías docentes más dinámicas y activas que logren además una mayor motivación de los alumnos, con la consiguiente mejora del proceso de aprendizaje. Todo ello en el contexto del proyecto EUROPA (**U**NA **E**NSEÑANZA **OR**IENTADA AL **AP**RENDIZAJE), que es una nueva actuación de la Universidad Politécnica de Valencia que persigue ilusionar a toda la comunidad universitaria en la mejora integral de la enseñanza y el aprendizaje.

1. Introducción

La asignatura optativa Informática Industrial de la intensificación en informática de la titulación Ingeniero en Automática y Electrónica Industrial, se imparte en el último curso de dicha titulación de segundo ciclo, en la Escuela Técnica Superior de Ingenieros Industriales. Los alumnos que cursan esta asignatura poseen unos conocimientos importantes de arquitectura de computadores, que han adquirido en asignaturas de primer ciclo.

Por ello, los objetivos de la asignatura se van a centrar más en que los alumnos desarrollen las habilidades que necesitarán en el futuro: saber hacer, aprender a aprender y aprender a ser [1], que en continuar incrementando sus conocimientos teóricos. Además esta decisión contribuye a incrementar su capacidad para adaptarse a los constantes cambios que se producen en la aplicación de la informática en la industria. Para ello se debe potenciar la mejora de las habilidades en el campo de la comprensión, la capacidad de búsqueda de información, especialmente a través de nuevas tecnologías (Internet), así como la síntesis y presentación pública de la misma, y mejorar las capacidades de trabajo y discusión en grupo.

2. Objetivos y temario

Con esta propuesta metodológica se pretende conseguir una enseñanza orientada al aprendizaje, para ello se propone enfocar la enseñanza a la consecución del saber hacer del alumno, consiguiendo además que desarrolle al máximo su capacidad de autoaprendizaje, así como mejorar los sistemas de evaluación favoreciendo la evaluación continua y la medida del saber hacer real del alumno.

La temática en la que pretendemos se alcancen los objetivos anteriores está formada por las siguientes unidades:

1. Representación de la información.
1.1 Técnicas de representación convencionales.
1.2 Técnicas orientadas a tolerar fallos.
1.3 Técnicas orientadas a compactar la información.
1.4 Técnicas de encriptación de la información.

2. La Unidad Central de Proceso.
2.1 Circuito de control.
2.2 Técnicas para el aumento de prestaciones.
2.3 Estudio de procesadores comerciales.

3. La memoria.
3.1 Memoria primaria.
3.2 Memoria secundaria.
3.3 Jerarquía de la memoria.

4. Sistemas Operativos.
4.1 Funciones de los S.O.
4.1.1Planificación.
4.1.2Manejo de la memoria.
4.1.3Manejo de la E/S.
4.2 S.O comerciales.

5. Sistemas informáticos industriales.
5.1 Sist. de diseño propio.
5.2 Sist. basados en buses industriales.
5.3 PC's industriales.
5.4 PLC's.

6. Sistemas tolerantes a fallos.
6.1 Fallos, errores y averías.
6.2 Sist. basados en dispositivos físicos.
6.3 Sist. basados en la programación.

En todos los temas se procura dar un enfoque desde el punto de vista de su aplicación industrial más que realizar una explicación teórica clásica.

3. Metodología propuesta

Para lograr los objetivos enunciados, centrados en la obtención de las capacidades y habilidades necesarias para el futuro ejercicio profesional, se hace necesaria la aplicación de metodologías que garanticen su consecución. Para ello, en el diseño de la asignatura se han planteado actividades orientadas a una mayor participación activa del alumno, para que éste genere esquemas conceptuales y no los herede del profesor, aumentando el espíritu crítico de los alumnos que permitirá desarrollar las habilidades descritas [2]. Las metodologías que se han seleccionado para su empleo en esta asignatura son:

- Método del caso: se trata de buscar soluciones a problemas reales en grupo, y discutirlas de forma pública posteriormente. Con ello conseguimos un primer acercamiento al proceso de análisis y propuesta de soluciones a un problema real. Su tratamiento en grupo y posterior discusión mejora la capacidad de trabajo en grupo a la vez que fomenta el pensamiento crítico.
- Miniproyecto: consiste en que los alumnos resuelvan en grupos reducidos (por ejemplo de dos personas), durante todo el curso, un problema real simplificado que podrán elegir. Cada grupo deberá analizar el problema, recopilar la información necesaria para solucionarlo, plantear una solución concreta, y llevar a cabo una implementación hardware y/o software de la misma (prototipado), dejando indicadas las líneas para desarrollar una solución definitiva. Asimismo para llevar a buen término esta actividad los alumnos deberán marcarse los objetivos propios a conseguir, decidir la metodología de trabajo, es decir la forma de lograr sus objetivos, realizar una investigación bibliográfica hasta el punto que consideren suficiente, sintetizar la información recabada, así como extender el ámbito del trabajo (opcional). La realización del miniproyecto en grupo mejora las habilidades de división del trabajo y coordinación [3]. Además, con esta actividad se persigue que el alumno aproveche más una tarea con un enorme potencial educativo como es la *tutoría*, que actualmente suele utilizarse en muy baja medida y principalmente en vísperas de exámenes.
- Presentaciones públicas: cada alumno deberá recopilar la información necesaria sobre un tema concreto relacionado con la asignatura y presentarla posteriormente en público, de forma resumida. El uso de temas relacionados con tecnologías de reciente aparición obliga al alumno a utilizar Internet como fuente bibliográfica. Asimismo, la restricción de exponer la información recopilada en un tiempo limitado obligará al alumno a seleccionar los puntos más importantes y resumirlos.

- Seminarios: consisten en reuniones de trabajo interactivo profesor–alumnos con el objetivo de fomentar el aprender a aprender y adquirir una visión más amplia de los conocimientos implícitos en las materias. Dado el carácter terminal de la asignatura y las características de los futuros ingenieros que la cursan, en los seminarios se plantean problemas interdisciplinares, que ayudan tanto a consolidar los conocimientos de los alumnos como a relacionarlos con otras materias.

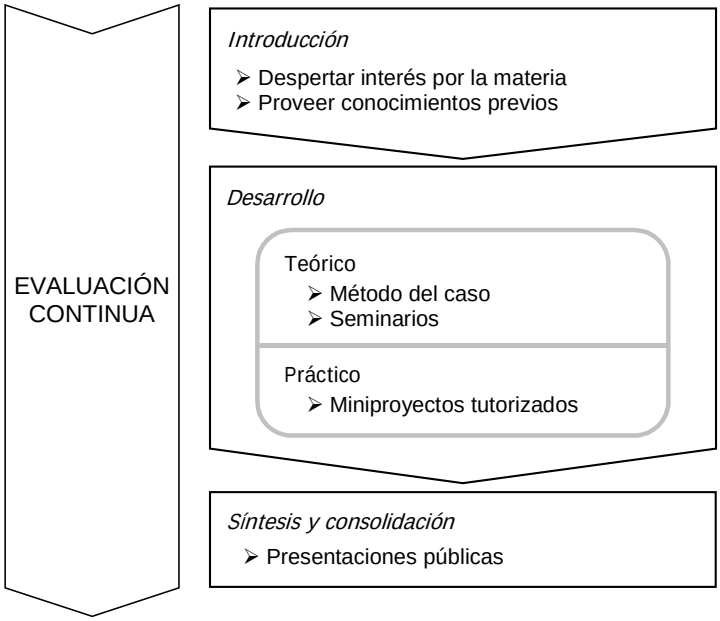


Figura 1. Proceso de aprendizaje

Como podemos observar en la figura 1, se pretende que el proceso de aprendizaje tenga una fase de introducción, en donde se despierte el interés por la materia y se provea de los conocimientos previos necesarios junto con una fase de desarrollo teórico, llevadas ambas a cabo mediante el método del caso y los seminarios, una fase de desarrollo práctico mediante los miniproyectos tutorizados, y finalmente una fase de síntesis, consolidación y comunicación mediante las presentaciones públicas. Esta metodología sigue las directrices de la actuación AME2 *"Nuevos métodos de enseñanza–aprendizaje"* del citado proyecto EUROPA.

Esencialmente se trata de diversificar los métodos e implementar metodologías activas que, en definitiva, vienen a cambiar sustantivamente la actividad del profesor aunque no su dedicación global.

Esta diversificación puede explicitarse en una distribución del tiempo invertido en la actividad docente que se repartiría como podemos ver en la figura 2 de la siguiente manera: 20% para la aplicación del método del caso, 20% para los seminarios, 40% para la realización de los miniproyectos y el último 20% para las presentaciones públicas. En suma, respondería a la estructura que propone el informe Delors sobre la educación para el siglo XXI. Finalmente, el desarrollo armónico de estas cuatro actividades debe conducir al cuarto principio enunciado por Delors de *"aprender a ser"* [1].

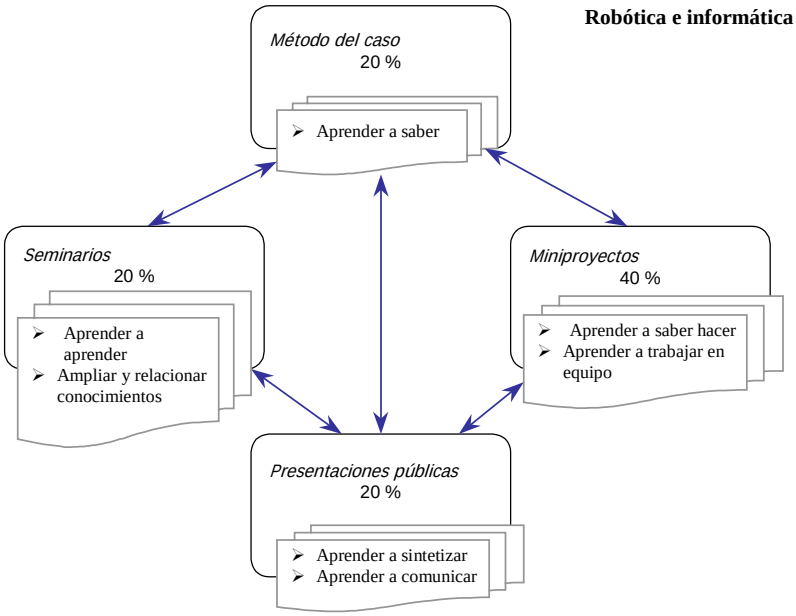


Figura 2. Distribución de los créditos de la asignatura

4. Evaluación

Para evaluar la asignatura se adoptará una evaluación continua multicriterio, es decir, evaluar al alumno teniendo en cuenta todas las actividades que lleva a cabo, de forma que éste tenga en todo momento conocimiento de sus calificaciones. Este planteamiento sigue las directrices de la actuación AME3 *"Mejora de los sistemas de evaluación"* del proyecto EUROPA.

Esta metodología se instrumentará integrando las diferentes técnicas de evaluación, de forma que complementen su efectividad y permitan valorar tanto parámetros objetivos de los conocimientos adquiridos como las habilidades desarrolladas con un mayor rigor y precisión. La modelación de esta estrategia se concreta en tres fases: 1.) Evaluación inicial, 2.) Seguimiento del proceso de aprendizaje y del progreso del alumno y 3.) Verificación de la consecución de los objetivos propuestos.

Todo ello conllevará suministrar al alumno desde el principio del curso una información clara sobre los objetivos y contenidos de la asignatura, metodología docente y criterios de evaluación.

En la aplicación de esta estrategia, la evaluación continua vendría a satisfacer tres objetivos, por una parte motivar al alumno a realizar un trabajo continuado y progresivo, por otra obtener un proceso con retroalimentación para que tanto alumnos como profesores dispongan en todas las fases de la información pertinente para realizar los ajustes oportunos, y finalmente se pretende potenciar un conocimiento mutuo entre alumnos y profesores.

Cada una de las actividades propuestas conlleva una calificación, que el alumno conoce en todo momento, de forma que la nota final de la asignatura se obtiene mediante la media ponderada de éstas. La aplicación del método del caso se puede evaluar solicitando a cada grupo que entregue por escrito la solución que propone. A partir de la misma y de su defensa, se propondrá una calificación. En los seminarios, el profesor valorará tanto la actitud de los alumnos como la calidad de las intervenciones que realizan, proponiendo de esta manera una nota para cada sesión.

Para la evaluación del miniproyecto se va a utilizar el diario de laboratorio, que consiste en un registro donde el alumno anota todas las tareas que va realizando en el laboratorio y donde se marcan unos ítems u objetivos, de manera que cuando éstos se alcanzan el profesor valora y valida los mismos mediante una nota y la estampación de su firma.

La evaluación de la presentación pública se realizará terminada ésta y teniendo en cuenta la opinión del resto de los alumnos.

Con el sistema de evaluación propuesto, se pretende eliminar el dramatismo del examen único o casi único mediante la incorporación de la evaluación analítica de todas las actividades realizadas por cada alumno, por otra parte tender a la evaluación global de cada alumno por medio de trabajos individuales interdisciplinares, y por último fomentar el trabajo en equipo de los alumnos.

5. Conclusiones

Se ha presentado una propuesta que lleva madurándose un tiempo por parte de los profesores de la asignatura y que es fruto de la motivación que han propiciado diversas actuaciones de la Universidad Politécnica de Valencia, como experiencias previas en Proyectos de Innovación Educativa, cursos de mejora de la cualificación pedagógica del profesorado, etc.

El nuevo enfoque y diseño de la asignatura van ser llevados a la práctica durante el próximo curso académico en el marco del proyecto EUROPA, actuación de la UPV con el objetivo de mejorar la calidad de la docencia en dicha universidad, y que está aportando los recursos necesarios para la puesta en marcha de experiencias como la presente.

Las metodologías propuestas han sido parcialmente probadas en otras asignaturas de la titulación Ingeniero Industrial, como por ejemplo Microprocesadores y Computadores [3]. Dado que los resultados han sido muy positivos, nos han motivado para acometer esta experiencia a la vez que nos proporcionan la base necesaria para afrontarla con éxito.

Referencias

[1] Delors, J. y otros. *La educación encierra un tesoro. Informe a la UNESCO de la Comisión Internacional sobre la educación para el siglo XXI*. UNESCO-Santillana Ed., 1996.

[2] Vicerrectorado de Coordinación Académica y Alumnado. *Proyecto EUROPA*. Servicio de Publicaciones UPV, 2001.

[3] Martí, A.; Ors, R. *Una experiencia innovadora en la asignatura "Microprocesadores y Computadores"*. I Jornadas de Innovación Educativa. Valencia, Septiembre 2001.

Robótica para las Ingenierías en Informática en la Universidad de Alicante

Miguel Ángel Cazorla, Otto Colomina, Juan Manuel Sáez
Dpto. Ciencia de la Computación e Inteligencia Artificial
Universidad de Alicante
03080 Alicante
e-mail: {miguel, otto, jmsaez}@dccia.ua.es

Resumen

En este artículo describimos una propuesta para los contenidos, tanto teóricos como prácticos de la asignatura Robótica en Ingeniería en Informática. Dicha propuesta está implementada actualmente sobre dicha titulación en la Universidad de Alicante.

1. Introducción

Uno de los contenidos cada vez más comunes en las Ingenierías en Informática es la robótica. En este artículo pretendemos hacer una reflexión de cuáles serían los contenidos a impartir en dicha asignatura y describiremos qué contenidos hemos seleccionado en la Universidad de Alicante. Robótica es una asignatura optativa para las tres Ingenierías (una superior y dos técnicas) que se imparten en esta Universidad.

2. El enfoque de la asignatura

En los primeros años de la robótica móvil, el enfoque utilizado para calcular los movimientos del robot se basaba en un conocimiento completo del entorno por el que se movía. Entre las características principales podemos destacar que no contemplaban que un objeto se pudiera desplazar ni tenían en cuenta la aparición de objetos no modelados. En cuanto a la arquitectura empleada para realizar el razonamiento se empleaba un esquema de percepción-cognición-acción. Este esquema primero obtiene las mediciones del mundo exterior y con ellas se produce la fase de cognición o razonamiento que da como resultado las acciones a realizar. Posteriormente se produce un replanteamiento de este tipo de arquitectura a un esquema percepción-acción, pasando a un planteamiento en base a conductas, donde cada conducta es un proceso individual y sencillo. El razonamiento queda implícito en la propia conducta y el conjunto de las conductas permiten el desarrollo de comportamientos complejos.

La robótica es un campo muy amplio que afecta a multitud de áreas en Informática. Se puede abordar desde un punto de vista físico, donde se estudiaría la cinemática, dinámica y el control de robots. Este enfoque, desde nuestro punto de vista, entra más en el campo de la automática que en el de la computación. Dentro del campo de la computación tendríamos contenidos como la planificación de trayectorias y la navegación. Este enfoque más *algorítmico* queda más cerca de la Ingeniería en Informática.

La evolución de las recomendaciones en los distintos curricula de informática ha pasado de la inclusión de Robótica como un ítem dentro de la Inteligencia Artificial a proponer una asignatura (optativa) con cuerpo propio dentro de la titulación, en el último curriculum conjunto de ACM y IEEE [2]. En este último curriculum la robótica que se propone se orienta hacia la robótica móvil.

Consultando los programas de las asignaturas en otras universidades, podemos observar que no existe un contenido unificado y que en general, se pueden englobar dentro de los dos enfoques antes mencionados: el de automática y el de computación. Igualmente, hemos observado que en sólo 23 de las universidades españolas se imparte este tipo de asignatura.

4. Propuesta de contenidos

La asignatura consta de seis créditos, tres teóricos y tres prácticos, repartidos en dos horas de teoría y dos de prácticas semanales. A partir de esta

premisa pasamos a relatar el contenido teórico y práctico que proponemos.

4.1. Temario teórico

El título de los distintos temas teóricos así como su secuenciación temporal se exponen a continuación:

Nombre del tema	Duración
Introducción a la robótica	2 horas
Conceptos geométricos	2 horas
Modelos geométricos de representación y de movimiento	3 horas
Algoritmos de planificación de trayectorias	5 horas
Sensores	4 horas
Visión en robots	4 horas
Navegación evitando obstáculos	2 horas
Navegación mediante conductas	4 horas
Localización del robot	4 horas

Hemos agrupado los temas por bloques temáticos. El primer bloque está compuesto por los dos primeros temas. El primero de ellos es un repaso del desarrollo histórico de la robótica, en el que se comienza comentando cuál es el origen de la palabra Robot (del checo *Robota*) y cuál es la imagen que tenemos de los robots. También presentaremos la evolución y la implantación de los robots en la industria, así como cuáles pueden ser los componentes de un sistema robótico. En el segundo tema introduciremos las herramientas matemáticas necesarias que utilizaremos más adelante en los siguientes temas de la asignatura, como por ejemplo, cómo podemos conocer y manejar las coordenadas de un objeto con respecto a nosotros si conocemos sus coordenadas y las nuestras con respecto a un sistema de referencia.

El siguiente bloque es la planificación del movimiento de un robot. El problema de la planificación se puede enunciar de la siguiente manera: dado un entorno (dimensiones, obstáculos) por el cual el robot se puede desplazar y una posición inicial y final del robot, el objetivo de la planificación es encontrar la secuencia de acciones que hacen que el robot se mueva de la posición inicial a la final sin colisionar con los obstáculos. Estas acciones pueden ser comandos de movimiento (distancia, velocidad lineal) y/o giro básico (giro, velocidad rotacional). Para realizar esto primero debemos definir la forma de representar tanto al robot como al entorno por el que se va a mover. Esto será el contenido del primer tema del bloque (tercero de la secuencia). En él, veremos las distintas formas de representación que se usan actualmente. También veremos las ecuaciones que gobiernan el movimiento del robot, para poder saber en todo momento cuál es la posición del mismo. El siguiente tema del bloque trata sobre los distintos algoritmos de planificación de trayectorias. Empezaremos simplificando el problema y tratando al robot como si no tuviera geometría (robot puntual), y no permitiendo el giro. Con esta simplificación se presentan dos algoritmos, el de grafo de visibilidad, que busca el camino más corto entre los vértices de los obstáculos, y el de descomposición del espacio libre, que intenta pasar lo más alejado posible de los obstáculos. A continuación permitiremos que el robot tenga una cierta geometría. Para ello, operaremos sobre los obstáculos para que sean estos los que contemplen la geometría del robot, mediante la suma de Minkowski. Con ello hemos reducido el problema de nuevo a la planificación con un robot puntual, pues la geometría de éste ya se contempla en los obstáculos. La última restricción es la del giro del robot cuya solución es inmediata haciendo uso de la suma de Minkowski. Por último, veremos un algoritmo de planificación distinto, haciendo uso de mapas topológicos, donde ya no utilizamos posiciones absolutas sino que intentamos detectar ciertas características del entorno (aperturas, pasillos, puertas, etc.), obteniendo un grafo que nos relaciona dichas características.

En el bloque anterior hemos visto distintos algoritmos de planificación del movimiento del robot. La mayoría de ellos asumen que la posición del robot es conocida en todo momento. Sin embargo, por problemas de odometría, la posición es cada vez más imprecisa conforme avanza el robot. Por ello es necesario que el robot obtenga información del mundo exterior. Para ello, hemos seleccionado aquellos sensores que pueden ser útiles a la robótica, pues existen multitud de sensores que pueden ser utilizados para una gran variedad de tareas. Dentro de este tema veremos dos tipos de sensores especiales: el GPS y los

Active beacons. Los dos tienen el mismo objetivo, esto es, obtener la posición de un objeto. El primero proporciona coordenadas relativas a la Tierra y el segundo lo hace con respecto a un entorno más reducido, típicamente una nave industrial. Veremos, por último, los sensores de rango que proporcionan discretizaciones del entorno próximo al robot, devolviendo distancias de los obstáculos justo enfrente del sensor. En cuanto a los sensores de rango, el sónar es el más utilizado en la navegación de robots debido a su bajo coste y a su gran velocidad de respuesta. El siguiente tema es el de visión para robots. Este tipo especial de sensor, la cámara, permite obtener mejores mediciones del entorno. Por un lado, veremos los fundamentos de la formación de la imagen: cómo se crea una imagen cuando usamos una cámara de *pin-hole* y qué es, en definitiva, una imagen. A continuación repasaremos algunos de los métodos de detección de características más básicos. Estos métodos permitirán obtener ciertas características de la imagen para determinar la posición o presencia de objetos o puntos del entorno que nos sirvan para alguna tarea concreta del robot. Por último, una aplicación directa de la visión en robótica es la detección de la profundidad de los objetos en la imagen, ya sea para tareas de navegación evitando obstáculos, como de reconocimiento de objetos. Para ello hemos seleccionado un algoritmo de estéreo para obtener dicha profundidad. Previamente a la obtención de la profundidad deberemos haber calibrado la cámara, es decir, haber obtenido los parámetros que permiten relacionar los puntos de la imagen con los del mundo.

El último bloque lo dedicaremos a la navegación de robots móviles. Hemos dividido el bloque en tres temas. En el primero se comentan dos enfoques para evitar y detectar obstáculos. El primero de estos enfoques se basa en los métodos de campo de fuerza. Estos métodos asumen que las lecturas de los sónares se pueden interpretar como fuerzas repulsivas y el objetivo como una fuerza atractiva. De esta manera, al sumar las dos fuerzas, tendríamos un vector que nos indicaría el comando a proporcionar al robot. Presentamos tres métodos, comenzando por el más intuitivo y sencillo de implementar, en el que vamos aumentando la complejidad para solucionar ciertos problemas inherentes en estos métodos. Cambiando de enfoque, pasamos a presentar un método que tiene en cuenta las velocidades (linear y angular) que lleva el robot y las que es capaz de conseguir debido a las restricciones físicas del propio robot. Este nuevo enfoque, el de la ventana dinámica, permite alcanzar velocidades mucho mayores que las conseguidas con los métodos anteriores.

El siguiente tema trata sobre las conductas de robots. Aquí veremos los fundamentos de las conductas así como las distintas arquitecturas que se han propuesto para el uso de las mismas. Terminaremos el tema especificando cómo se pueden implementar conductas sencillas y cómo maneja Saphira las conductas.

El último tema trata sobre la localización de robots. En un primer apartado veremos que el robot necesita conocer el entorno por el que se mueve, por lo que es necesario construir un mapa del entorno. Para construir un mapa el robot vagabundea por el entorno actualizando una rejilla de ocupación con las dimensiones del entorno. Dicha actualización se realiza de acuerdo a las lecturas de los distintos sónares. Una vez construido el mapa, podemos utilizarlo en la localización del robot. La localización sirve para averiguar dónde está el robot dentro de un entorno conocido. Para llevar a cabo esto detallaremos un método novedoso que hace uso de teoría bayesiana para determinar de forma dinámica la posición del robot.

4.2. Temario de prácticas

Para la realización de las prácticas haremos uso de la librería *Saphira* (disponible en http://www.ai.sri.com/~konolige/saphira). Como ventajas fundamentales de esta librería destacamos la programación en C ó C++ y la posibilidad de trabajar con el simulador y/o con un robot real. Al inicio de las prácticas haremos un pequeño seminario sobre la utilización de esta herramienta.

La propuesta de prácticas se divide como se muestra en la siguiente tabla:

Nombre del tema	Duración
Herramienta *Saphira*	4 horas
Planificación del movimiento de un robot	10 horas
Percepción del robot	6 horas
Navegación topológica basada en comportamientos	10 horas

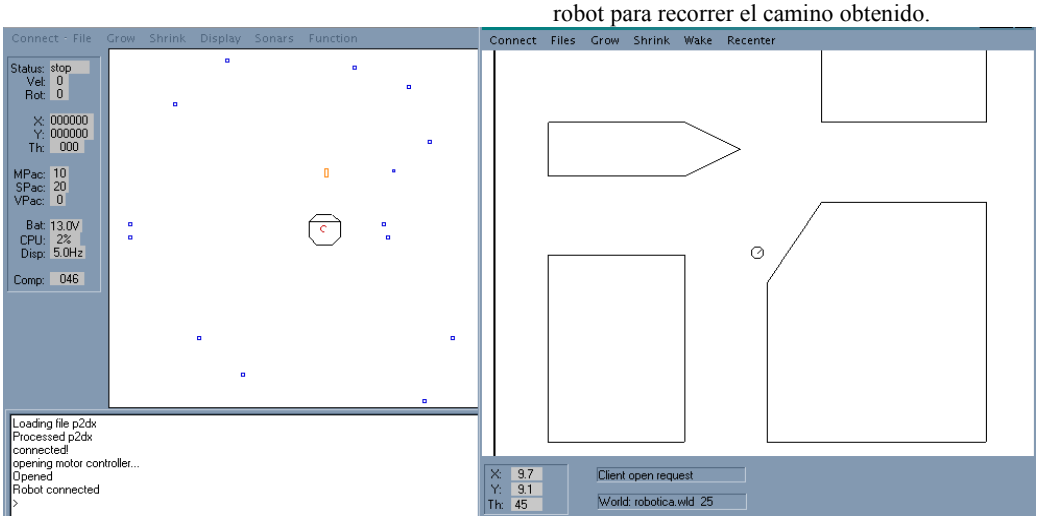


Figura 1. Entorno *Saphira*

El primer tema de prácticas tiene dos objetivos fundamentales. El primero es dar a conocer al alumno el funcionamiento de la herramienta de control de robots móviles *Saphira*. Para ello, detallaremos todas las funcionalidades de la herramienta (descripción de mundos, descripción de distintos tipos de robots, conexión con el simulador-robot real, etc.). El segundo objetivo es que el alumno entienda y sepa aplicar los conceptos de transformación de coordenadas entre sistemas de referencia. La práctica consiste en guiar al robot dentro de un entorno, pasando por una serie de puntos indicados en un fichero. Las coordenadas de los puntos están dadas con respecto al sistema de coordenadas del mundo donde se encuentra el robot. Estas coordenadas se deben transformar al sistema de referencia del robot.

En la segunda práctica se pretende guiar al robot por un mundo conocido. Disponemos de las posiciones y las dimensiones de los obstáculos en un fichero de descripción del mundo. También disponemos de la geometría del robot definida mediante sus vértices. El desarrollo de la práctica será el siguiente: 1. Lectura del fichero de descripción del mundo. 2. Aplicación de la suma de Minkowski a los obstáculos. 3. Obtención del grafo de visibilidad. 4. Cálculo del camino más corto desde el punto inicial al final. 5. Guiado del robot para recorrer el camino obtenido.

En la parte de percepción proponemos el desarrollo de una pequeña práctica en la cual el alumno pueda guiar al robot hacia un objeto dentro del campo de visión del robot. Este objeto debe ser de un color uniforme y fácilmente identificable del fondo. Por ejemplo, podemos hacer uso de una pelota de tenis. Al alumno se le suministrará una pequeña aplicación que captura imágenes de una cámara colocada sobre el robot y devuelve la imagen binarizada a partir de un color previamente seleccionado. El objetivo es que guíe al robot hacia el objeto en cuestión.

El objetivo de la última práctica es implementar un algoritmo de navegación reactiva basado en comportamientos. El algoritmo tomará como entrada una ruta a seguir en forma de mapa topológico (es decir, una secuencia de *lugares distintivos* por los que hay que pasar, en lugar de una secuencia de coordenadas a recorrer como hemos hecho hasta el momento).

Figura 2. Binarizado de un objeto y orientación del robot

4.3. Bibliografía para la asignatura

En este apartado vamos a detallar algunas de las referencias que se ajustan mejor al temario propuesto. En [3] se realiza un repaso exhaustivo a la robótica basada en conductas. El autor, R. Arkin, es uno de los precursores de la robótica basada en conductas. Es un libro monotemático sobre conductas de robots. El libro [4] recorre todo el campo de la robótica móvil. Está divido en tres partes. La primera trata sobre la parte física del robot: distintos tipos de conducción y distribución de ruedas, dinámica del robot, etc. En la segunda, se estudian las distintas formas de percepción que tiene un robot, diviéndolas en dos grupos: sensores y algoritmos no visuales y visuales. La última parte, más extensa, trata sobre el razonamiento, qué cosas o tareas puede hacer nuestro robot. En esta última parte tenemos un capítulo dedicado a la planificación de robots, al control reactivo, la localización, la construcción de mapas y tareas reales en las cuales se están utilizando robots. En [5] se trata en profundidad toda la parte de planificación de trayectorias de robots. Define el problema formalmente y con una matemática rigurosa. Trata el problema de la planificación de la forma más genérica posible, con obstáculos n-dimensionales y permitiendo formas curvas para su definición. Empieza el libro con varios capítulos dedicados a la formalización del problema, para luego pasar a tratar a fondo el problema de la planificación desde tres puntos de vista: la descomposición en celdas exactas y aproximada, los métodos de mapa de carreteras y los métodos de campo de potencial. Continúa con la coordinación de varios robots, imponiendo restricciones cinemáticas al robot y permitiendo, por último, que los obstáculos puedan desplazarse. El tema principal de [6] es la Inteligencia Artificial aplicada a la robótica. Estudia la robótica desde un punto de vista computacional, sin entrar en aspectos más de bajo nivel como la cinemática y el control dinámico. El libro está dividido en dos partes. La primera, Paradigmas de la robótica, trata sobre los distintos paradigmas que se pueden utilizar en robótica para controlar un robot, el enfoque jerárquico más clásico y el control reactivo, revisando otros paradigmas en menor grado. Repasa las ventajas e inconvenientes de los dos primeros y se centra en el segundo en el resto del libro. Dentro de esta parte existe un capítulo dedicado a percepción, centrado en sensores y visión. En la segunda parte, Navegación, se centra en la planificación de trayectorias, tanto topológicas como métricas y en el problema de la localización y construcción de mapas. Incorpora bastantes detalles de implementación que lo hacen atractivo en la docencia. Por último, para la parte de visión, podemos consultar [7]. Este libro presenta una serie de problemas en visión artificial de forma clara y concisa. En cada capítulo presenta un problema distinto, empezando por la definición del problema, aplicando las restricciones oportunas y, por último, detallando algún método para resolver el problema en cuestión. Todo esto se acompaña de resultados de la aplicación de los distintos métodos presentados. El contenido del libro se puede estructurar en tres partes. La primera de ellas trata sobre la formación de una imagen y de cómo podemos mejorar la calidad de la misma mediante la supresión de ruido. Dentro de esta parte podemos incluir los capítulos dedicados a la obtención de características bidimensionales de la imagen: aristas y puntos esquina. Otra de las partes tiene que ver con la obtención de la estructura a partir de diferentes métodos: estéreo, movimiento,etc. Dentro de esta parte tenemos un capítulo sobre calibración de la cámara, necesaria para obtener la estructura. La última parte del libro introduce el problema del reconocimiento de objetos y la localización de objetos en el espacio.

5. Conclusiones

En este trabajo hemos presentado una propuesta para el temario de la asignatura Robótica, implementada actualmente en las Ingenierías en Informática en la Universidad de Alicante. Para ello hemos especificado tanto la parte teórica

como la práctica y hemos comentado algunas de las posibles referencias a utilizar.

Referencias

[1] Miguel Cazorla. *Página web de la asignatura Robótica*. http://www.dccia.ua.es/dccia/inf/asignaturas/ROB

[2] ACM/IEEE, *Computing Curricula 2001*, 2001, IEEE Computer Society/ACM Task Force on Computing Curricula

[3] R. Arkin. Behavior Based Robotics. The MIT Press, 1998.

[4] D. Dudek and M. Jenkin. Computational Principles of Mobile Robotics. Cambridge University Press, 2000

[5] J. Latombe. Robot Motion Planning. Kuwler Academic Publisher, 1991.

[6] R. Murphy. Introduction to AI Robotics. he MIT Press, 2000.

[7] E. Trucco and A. Verri. Introductory Techniques for 3-D Computer Vision. Prentice Hall, 1998

La enseñanza de Ingeniería de Sistemas y Electrónica mediante el laboratorio de robots autónomos

Jesús Salido Tercero, Jorge Sanz Alcolea
Dept. de Ingeniería Eléctrica, Electrónica y Automática
Universidad de Castilla-La Mancha
Escuela Superior de Informática
13071 Ciudad Real
e-mail: {Jesus.Salido, Jorge.Sanz}@uclm.es

Resumen

En este trabajo se presenta el empleo de sistemas físicos controlados por computador como un método muy adecuado para introducir a los alumnos de Informática en las áreas tecnológicas relacionadas con la Ingeniería de Sistemas y Electrónica. La idea principal es conseguir estimular la capacidad de aprendizaje de ciertas materias por parte del estudiante mediante la realización de prácticas en un laboratorio docente destinado a la construcción de robots autónomos. Este laboratorio debería permitir materializar las ideas del alumno a través de dispositivos capaces de resolver problemas en el 'mundo real', incluso con el estímulo de que una solución personal sea más eficiente compitiendo con la propuesta por un compañero. En los siguientes apartados se describirá las principales metas académicas del laboratorio propuesto, así como el material empleado y los prototipos desarrollados para ser empleados como modelos en la práctica.

1. Introducción y antecedentes

Durante las dos últimas décadas se han desarrollado esfuerzos considerables en los campos de estudios de la Robótica y el Control Inteligente y no puede negarse los logros tan importantes que se han conseguido. Sin embargo, si se analiza detenidamente los avances en el terreno de los robots autónomos, es evidente que aún existen importantes desafíos y que por el momento podemos hablar de prometedoras 'cibermascotas' [8], muy lejos aún de los androides protagonistas en las historias de ciencia ficción. No obstante los investigadores que han dedicado un gran esfuerzo en el campo de la Robótica han aprendido durante el camino importantes lecciones, *...ahora sabemos como construir fácilmente y de forma barata, robots capaces de resolver tareas que en el pasado resultaban asombrosamente complejas...* Por otra parte en los últimos años la demanda creciente de especialización en áreas relacionadas con la Informática ha forzado la adaptación de los planes de estudios para ofrecer graduados con una formación exigente en las áreas relacionadas con las nuevas tecnologías. Estas tendencias se han visto fortalecidas por las recomendaciones procedentes de instituciones de prestigio en la materia, como son IEEE y ACM[1], acerca de la adaptación del currículo académico a las demandas actuales del mercado laboral.

La propuesta presentada en este artículo se encuadra en el contexto institucional de la Universidad de Castilla-La Mancha (UCLM) en la que los autores realizan sus tareas docentes en la Escuela Superior de Informática (ESI). En este contexto la idea fundamental expuesta es la creación de un nuevo laboratorio para la construcción de robots autónomos controlados por computador, como método de apoyo en la docencia de los contenidos de las titulaciones técnicas en Informática. El planteamiento seguido en la propuesta presentada está estrechamente enraizada en las aproximaciones más recientes para la implementación de *agentes inteligentes*.

El principal objetivo del trabajo presentado es la de proporcionar al alumno la oportunidad de aprendizaje a través de la práctica, plasmando en un dispositivo físico real, los principales

[1] Institute of Electrical and Electronics Engineers, Association for Computing Machinery.

conocimientos adquiridos a lo largo de su formación académica.

En esencia los robots construidos en el laboratorio mencionado deberían estar compuestos de:

- *Sistema de percepción* para adquisición de información sobre el entorno circundante y el propio estado interno del agente,
- *Controlador* o sistema de decisión a cerca de las acciones inmediatas a realizar para alcanzar los objetivos del agente, y
- *Actuadores* capaces de ejecutar las acciones estimadas como más apropiadas por el controlador.

Desde el punto de vista de las infraestructuras, el laboratorio propuesto constituirá una extensión de un laboratorio de computación ya existente y equipado con computadores personales convencionales. Dicho laboratorio se dotará del hardware adicional y las infraestructuras auxiliares necesarias permitiendo el trabajo de grupos compuestos por dos o tres estudiantes por puesto. Los puestos contarán con kits de robótica cuyos componentes sean de gran disponibilidad en el mercado y un coste total no superior a los 600€/kit. Nuestro propósito es que una vez satisfechos estos requerimientos este esquema de trabajo pueda ser fácilmente reproducible en otros centros académicos.

La filosofía aquí presentada está avalada por las propuestas realizadas en otras instituciones de renombrado prestigio tanto dentro como fuera de nuestras fronteras. En estos casos se apuesta firmemente por un método docente basado en la construcción de dispositivos físicos capaces de demostrar su desempeño en la resolución de 'problemas tipo' o incluso mediante la participación en competiciones públicas abiertas [2,7,11,17]. En esta línea, resultan especialmente atractivas las propuestas para la construcción de robots móviles, ya que estos ofrecen la posibilidad de integrar los conocimientos de un número muy significativo de áreas de conocimiento como: lenguajes de programación e ingeniería del software, arquitectura de computadores, ingeniería eléctrica y de control, física, etc. Entre todas las experiencias llevadas a cabo en otros centros académicos merecen una mención especial los llevados a cabo en EEUU, como los del MIT [12, 15] (Cambridge, MA) y Carnegie Mellon University [3] (Pittsburg, PA), señalando que recientemente estas experiencias han sido adoptadas con gran éxito en centros de nuestro país [1, 4, 5, 18].

2. Objetivos académicos

Con el laboratorio de robots autónomos se pretende cubrir el mayor conjunto posible de conceptos clave en las titulaciones de Ingeniería en Informática, entre otros: *interacción hardware-software, complejidad espacial en términos de las limitaciones físicas reales de capacidad de computo y de memoria del controlador del robot, complejidad temporal ya que las decisiones requeridas sobre las acciones del robot deben tomarse en el instante apropiado, y quizás el que puede ser más importante, el incentivo motivador que representa para el estudiante dotar de 'vida' a la ideas propias.*

La aproximación adoptada en el trabajo puede considerarse una instancia de las orientaciones más recientes de la inteligencia artificial basadas en 'agentes' [19]. Sin embargo hay que reseñar la mejora que para dicha aproximación supone el hecho de que los agentes 'habiten' un mundo físico real y por tanto exento de muchas simplificaciones incluidas en los entornos de trabajo simulados. Por supuesto la construcción de un robot móvil en un laboratorio docente no sería posible si no fuera por la disponibilidad actual en el mercado de kits de ensamblaje asequibles. Estos kits son programables mediante empleo de computador personal convencional y son ampliables para cubrir un amplio rango de aplicaciones y experimentos.

En nuestro centro (ESI-UCLM) el laboratorio propuesto puede encuadrarse en la docencia de dos cursos cuatrimestrales:

- *Cibernética Aplicada,* impartido en las titulaciones de ingeniería técnica en informática (grado medio, curso 3º, 1er. cuatrimestre),

- *Robótica,* en la titulación de ingeniería en informática (grado superior, curso 5º, 2º cuatrimestre).

Además de los cursos anteriormente citados, esperamos que el laboratorio permita la realización adicional de prácticas de otros cursos como: Arquitectura de Computadores, Programación, Electrónica, Automatización Industrial, Inteligencia Artificial, Diseño y Síntesis de Hardware, Control por Computador, Sistemas de Tiempo Real, Procesamiento de Imagen, etc. Aunque todos estos cursos pertenecen a los estudios de pregrado, se animará a los estudiantes para que en dicho laboratorio realicen proyectos adicionales con posibles extensiones para completar sus proyectos fin de carrera y otros proyectos de investigación [21].

3. Equipamiento del laboratorio

Como ya se comentó más arriba los grupos de trabajo en el laboratorio estarán compuestos de dos o a lo sumo tres estudiantes a los que se dotará del siguiente material: 1) computador personal, 2) tarjeta controladora, 3) herramientas software de programación, y 4) kit de ensamblaje de robots incluyendo sensores, actuadores y los bloques constructivos del fabricante Fisher Technik [9]. El computador personal se empleará para la edición y depurado de los programas de usuario que más tarde serán descargados desde el computador a la tarjeta controladora a través de una conexión serie.

El componente principal del kit de robótica es el módulo de control adoptado (ver Fig. 1), que en nuestro caso consta de las tarjetas modelos CT6811 (módulo principal) y CT292+ (control de motores) de Microbótica [14]. Al igual que la Handy Board [20], desarrollada en el MIT con similares propósitos, la CT6811 está basada en el microprocesador MC68HC11 de Motorola. La tarjeta puede emplearse en tres modos operativos según se describe a continuación:

1. *Tarjeta de entrenamiento*. La tarjeta permanece conectada mediante enlace serie al computador personal en el que se desarrollan los programas que posteriormente se descargan en la tarjeta.
2. *Tarjeta controladora autónoma*. En este caso no existe conexión alguna con el computador externo, de modo que cada vez que la tarjeta es encendida se inicia la ejecución de un programa previamente almacenado en la memoria EEPROM del microcontrolador. Este es el modo que debe ser empleado en los robots autónomos.
3. *Unidad periférica inteligente de un computador externo*. En este modo la tarjeta está continuamente conectada a un computador externo con posibilidad de comunicación bidireccional como si de un periférico convencional se tratase.

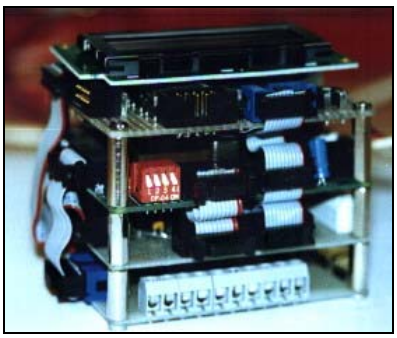


Fig. 1: Ensamblado en formato torre de los módulos de control.

Deben reseñarse algunas características interesantes del módulo de control descrito, que lo hacen adecuado para nuestros propósitos. Se trata, de la capacidad de adoptar un formato de ensamblaje en torre muy compacto que permite la integración del sistema en unas dimensiones superficiales de 8cmx6cm, como se muestra en la Fig. 1. Además el carácter de 'abierto' de la arquitectura hardware del módulo, así como de las herramientas de programación constituyen un valor añadido en las posibilidades de uso del módulo.

La tarjeta CT6811 está equipada con el microcontrolador MC68HC11 cuyas principales características son: 512 Bytes de memoria EEPROM (series A1/A8) y 2Kbytes (serie E2), memoria RAM de 256 Bytes, timer de 16 bits, 3 latches de entrada, 5 comparadores, un acumulador de pulsos, comunicación serie A/S, 8 canales de conversión A/D, interrupción de TR y 4 puertos I/O. La tarjeta CT293+ está diseñada para ensamblarse junto a la CT6811, proporcionando la capacidad para el control de motores DC y la adquisición de datos sensoriales.

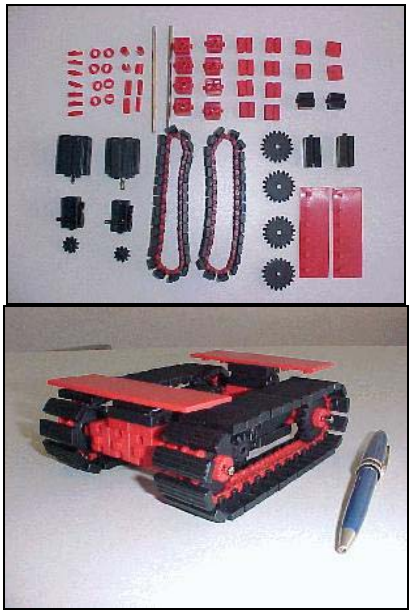


Fig. 2: Prototipo 'oruga'.

La programación del módulo de control se puede hacer fuera de línea desde un PC con el compilador de lenguaje ensamblador AS11 de Motorola y las herramientas de programación para los sistemas operativos Linux y DOS suministradas por el fabricante. Además basta con realizar una búsqueda en Internet para localizar multitud de recursos en línea disponibles gratuitamente debido a la gran popularidad del microprocesador empleado [13].

En cuanto a los módulos de ensamblaje de la estructura mecánica existen multitud de fabricantes que proporcionan kits de construcción con propósitos educativos y de entretenimiento [16] a unos costes asequibles. Nuestra experiencia en este sentido se ha traducido en la adopción de los productos comercializados por Fischer Technik [9], debido a la gran calidad de sus componentes y la contrastada robustez de los prototipos finales. Con los componentes proporcionados por este fabricante y otros adicionales se dispone de los elementos fundamentales para completar los proyectos que se sugerirán. Dichos proyectos deben ser llevados a cabo empleando: los bloques constructivos básicos, motores DC, sensores de contacto (micro-interruptores), sensores de infrarrojo, detectores de metal, codificadores ópticos, etc.

4. Prototipos

Anteriormente se ha comentado la dificultad que encierra la tarea de construcción de robots móviles y como puede superarse este obstáculo mediante el empleo de kits de montaje. En general los fabricantes de estos últimos proporcionan con sus productos instrucciones detalladas a seguir para alcanzar la construcción de los modelos sugeridos. Como parte del trabajo aquí presentado se ha preferido el desarrollo de modelos propios originales demostrando así que el número de posibilidades sólo está limitado por la imaginación del usuario.

En las figuras 2 y 3 se muestran dos de nuestros prototipos ('oruga' y 'tribot') y los elementos constructivos empleados en ellos, destacando el pequeño número de éstos últimos. Como puede apreciarse en las figuras 2 y 3 los prototipos mostrados emplean distintos sistemas motrices. Aunque en estos sistemas motrices sólo se emplean

dos motores DC, el módulo de control permite la expansión para controlar robots con un número mayor de motores como puede verse en los prototipos desarrollados por Microbótica (véase el sitio web.[14]).

Una característica muy importante a destacar en los prototipos desarrollados es que han sido concebidos para acomodar el módulo de control en formato torre de Microbótica de modo que la instalación de este sea lo más cómoda y rápida posible para que el mismo módulo sea intercambiable entre prototipos a la manera de un dispositivo 'plug&play' (ver Fig. 4).

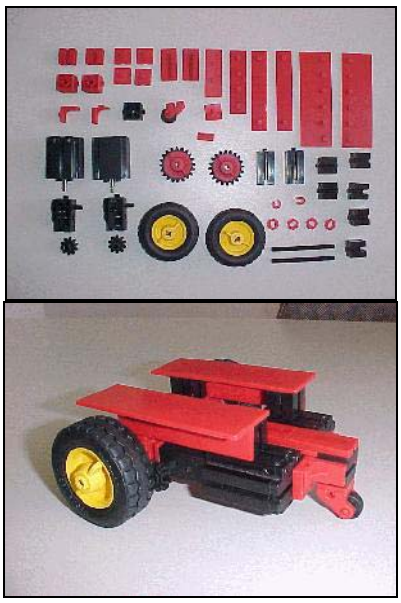


Fig. 3: Prototipo 'tribot'.

5. Trabajos futuros

En esta sección queremos destacar que la propuesta aquí presentada no ha sido llevada a la práctica aún en su totalidad. Nuestra intención es que a partir de próximos cursos académicos podamos ir contrastando la viabilidad de los objetivos marcados y su consecución. Por el momento se han fijado los objetivos y se han concretado los contenidos e infraestructuras para llevar a cabo la propuesta. En las prácticas de laboratorio en curso se han probado con un grupo reducido de alumnos los aspectos principales de la propuesta. A continuación se enumeran las que, en nuestra opinión, deberían ser las tareas más inmediatas para alcanzar nuestros propósitos:

- Publicación de los programas del curso, manuales y programación temporal, así como de otro material auxiliar.
- Desarrollo de entornos de simulación y programación favoreciendo el desarrollo de entornos gráficos y sistemas abiertos.

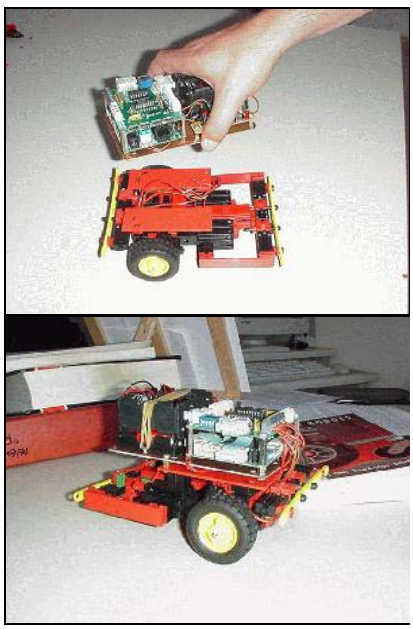


Fig. 4: Prototipo final 'tribot' controlado por la CT6811 y CT293+.

- Incremento de capacidades de procesamiento y memoria del módulo de control añadiendo mayor cantidad de memoria de usuario y dotando al conjunto de sistemas sensoriales complejos como sónares y cámaras de visión.
- Aumentar las posibilidades de actuación de los robots mediante la adaptación de brazos articulador y pinzas de sujeción.
- Adaptación de sistemas de comunicación inalámbrica entre el robot y una estación

externa de procesamiento, o bien entre distintos robots.

Sumario

En este trabajo se describe la propuesta de un laboratorio de robots autónomos como un medio de mejorar el método docente en varios cursos de pregrado en las titulaciones de Ingeniería Informática. En sucesivos apartados se han comentado las principales motivaciones y objetivos, enumerando los principales componentes del equipamiento previsto. Se ha incluido un apartado en el que se comentan los prototipos originalmente propuestos por los autores y se finaliza con un apartado de las futuras acciones para llevar a cabo la propuesta en su totalidad.

Agradecimientos

Los autores desean manifestar su especial reconocimiento hacia la Escuela Superior de Informática (ESI) y la Universidad de Castilla-La Mancha por el inestimable soporte proporcionado para la realización tanto de este trabajo como la de su labor docente.

Referencias

[1] Alcabot-2000. http://www.depeca.alcala.es/alcabot2000/principal.htm

[2] Beer, R.D. and Chiel, H. and Drushel, R. *Using Autonomous Robotics to teach Science and Engineering*. Commun. ACM, 42, 6, Jun. 1999.

[3] Carnegie Mellon Univ. Annual Mobot Slalom Race. http://www.cs.cmu.edu/~mobot/index.html

[4] ChampionBot en la UPM http://www.sia.upm.es/championbot

[5] Concurso Microbótica en ESIDE http://www.eside.deusto.es

[6] Davies, B. *Practical Robotics*. WERD Technology Inc., 1997.

[7] Donnett, J. and Smithers, T. *LEGO Vehicles: A Technology for studying Intelligent Systems*. In J.A. Meyer and S.W. Wilson, Eds., From Animals to Animts: Proceedings of the First International Conference on Simulation of Adaptive Behavior. MIT Press, Cambridge, Mass. 1991.

[8] Entertainment Robot Aibo (SONY). http://www.world.sony.com/Electronics/aibo/

[9] Fischer Technik. http://www.fischerwerke.de/

[10] Jones, J. and Seiger, B. and Flynn, A. *Mobile Robots. Inspiration to implementation*. 2nd. Ed., A.K. Peters LTD. 1999.

[11] Kumar, D. and Meeden, L. *A robot laboratory for teaching artificial intelligence*. Proc. of the Technical Symposium on Computer Science Education (SIGCSE-98), D. Joyce, Editor, ACM Press, 1998.

[12] Martin, F. *Circuits to control: Learning Engineering by designing LEGO Robots*. Ph.D. Dissertation, MIT Program in Media Arts and Sciences, Cambridge, Mass., 1994.

[13] MC68HC11 Microcontrollers Resources. http://fleming0.flemingc.on.ca/~pspasov/mcu/mcu.htm

[14] Microbótica. http://www.microbotica.es/

[15] MIT's Autonomous Robot Design Competition. http://www.mit.edu/courses/6.270/home.html

[16] Raucci, R. *Personal Robotics. Real robots to construct, program, and explore the world*. A.K. Peters LTD. 1999.

[17] Resnick, M. *Behavior construction kits*. Commun. ACM, 36, 7, Dec. 1993.

[18] Robótica en la Universidad Politécnica de Cataluña (Spain). http://citel.upc.es/~aess/robotica

[19] Russell, S. and Norvig, P. *Artificial Intelligence a modern approach*. Prentice Hall Inc., 1996.

[20] The Handy Board. `http://lcs.www.media.mit.edu/groups/el/Projects/handy-board/`

[21] Los Robots Móviles en ISA-UCLM. `http://www.inf-cr.uclm.es/www/jsalido/mancheBotX.html`

Desarrollo de prototipos *hardware* para una maqueta de tren con fines docentes

Vicent Lorente, Silvia Terrasa,
Dept. de Informática de Sistemas y Computadores
Universidad Politécnica de Valencia
46022 Valencia
e-mail: {vlorente,sterrasa}@disca.upv.es

Ana García
Dep. Sistemas Informáticos y Computación
Universidad Politécnica de Valencia
46022 Valencia
e-mail: agarcia@dsic.upv.es

Resumen

El uso de elementos reales con fines docentes es cada vez mas frecuente. El presente trabajo presenta la experiencia de la puesta en marcha de una maqueta de trenes describiendo los problemas surgidos a la hora de realizar el control por computador de la misma, así como las soluciones propuestas. El trabajo se centra en la descripción de la infraestructura hardware desarrollada sobre una maqueta comercial, para permitir el control individual de los elementos móviles (trenes, cambios de vía…) mediante un ordenador.

1. Motivación

Con la puesta en marcha de los últimos planes de estudio y con la gran diversidad de asignaturas optativas y la limitación de recursos que tienen algunos centros (aulas, laboratorios, etc), las asignaturas optativas entran dentro del juego de "la oferta y la demanda" ya que si no mantienen un número mínimo de alumnos por curso corren el peligro de desaparecer. Para contrarrestar este efecto, los profesores no sólo deben hacer que sus optativas sean interesantes intelectualmente, sino que también deben introducir elementos que las hagan atractivas. En este marco, el departamento de sistemas informáticos y computación (DSIC) de la universidad Politécnica de Valencia adquirió el curso pasado una maqueta de trenes para ser controlada desde un computador personal (PC).

El presente artículo describe el trabajo realizado en la puesta en marcha de la maqueta y los problemas surgidos al intentar realizar el control de la misma desde un PC.

El artículo está estructurado de la siguiente forma, en la sección 2 se describe la maqueta de trenes así como el control manual que lleva incorporado. En la sección 3 se detallan los problemas surgidos a la hora de cambiar el control manual por el control por ordenador y en la sección 4 se describen de los prototipos realizados para solventar estos problemas. Finalmente en la sección 5 se comentan una serie de conclusiones y el trabajo futuro.

2. Descripción de la maqueta de trenes.

Durante el pasado curso el departamento DSIC de la Universidad Politécnica de Valencia adquirió una maqueta de trenes proporcionada por MÄRKLÍN, para dedicarla a aplicaciones docentes.

Como se puede apreciar en la figura 1 el montaje inicial es sencillo, no existen desniveles ni túneles, aunque si que hay cambios de vía para permitir un control más elaborado. El problema que tiene este tipo de maquetas es que se necesita un presupuesto muy elevado para montar una maqueta completa con toda clase de detalles. Debido a esto, se decidió invertir inicialmente el dinero suficiente para realizar un montaje útil, y después de analizar los resultados, complicar el diseño y ampliar la maqueta.

En la configuración inicial de la maqueta se tuvieron en cuenta tanto la dificultad que podría tener para el alumno realizar el control simultáneo

de diferentes trenes, como la dificultad relativa a la utilización de las vías (recursos compartidos a utilizar en exclusión mutua) por parte de los trenes.

También el tema económico nos condujo a implementar nosotros mismos las placas prototipo para realizar el control de los trenes desde el PC. Estas placas se han diseñado para resolver los problemas de la identificación y la ubicación de los trenes.

Figura 1 Configuración de la maqueta

Por otra parte, en la figura2, se muestra el control manual original de la maqueta. Este control manual ha sido utilizado para confirmar el buen funcionamiento de la misma antes de realizar pruebas con el PC.

Figura 2 Control manual de la maqueta MÄRKLÍN

Los trenes disponen de un microcontrolador incorporado que lleva implementado un protocolo de comunicación con el panel de control. Este protocolo de comunicación permite actuar individualmente sobre cada tren, siendo algunos de los parámetros sobre los que se puede actuar los siguientes :

- Marcha / Paro.
- Sentido de la marcha (horario/ antihorario).
- Velocidad.
- Aceleración / Deceleración.
- Elementos decorativos (humo, luces, silbato, etc)

Adicionalmente, MÄRKLÍN ofrece un módulo que permite controlar la maqueta mediante un ordenador por medio del puerto serie. Esté módulo, llamado *interface*, sustituye al control manual y realiza la adaptación de los dos protocolos, es decir por una parte envía y recibe comandos desde el PC y por otra envía y recibe comandos desde los trenes (Figura 3).

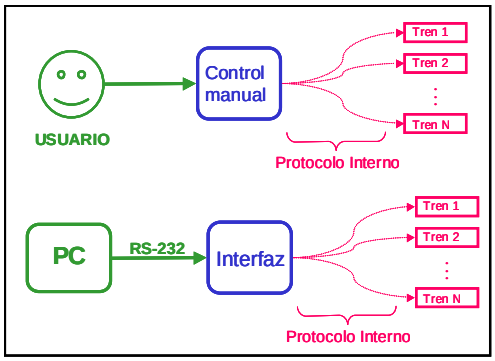


Figura 3 Posibles modos de funcionamiento.

Finalmente la maqueta tiene una serie de sensores de posición que indican la presencia o no de un tren en un punto del recorrido (por ejemplo, es interesante saber cuando hay un tren en un final de vía).

El proceso del montaje de la maqueta con estas características tuvo una duración de 1 mes aproximadamente ya que primero se tuvo que preparar una mesa con las condiciones adecuadas para permitir que toda la parte electrónica estuviese por debajo de la maqueta.

3. Control de la maqueta desde el PC.

Una vez finalizado el montaje de la maqueta, se abordó realizar el control de los trenes desde un PC. La maqueta se adquirió con objeto de realizar trabajos prácticos de asignaturas optativas de primer y segundo ciclo, así como proyectos fin de carrera y asignaturas de doctorado. En la fase de evaluación inicial, ha sido utilizada para realizar trabajos de asignaturas de tercer ciclo relacionadas con inteligencia artificial y con sistemas de tiempo real. La temática de las asignaturas nos da la oportunidad de plantear un control a dos niveles, por una parte la planificación de tiempo real intenta resolver el control a bajo nivel, evitando situaciones de peligro y actuando sobre los trenes directamente, mientras que la inteligencia artificial define un control de alto nivel donde se puede trabajar sobre optimización de trayectorias, planes de recorridos, etc. Este control a dos niveles nos proporciona una visión muy global del control de la maqueta que permite explotar toda su utilidad.

Para poder realizar el control de la maqueta en los dos niveles es necesario:

- Establecer un protocolo de comunicación entre el PC y los trenes.
- Localización de cada tren.
- Control sobre los cambios de vía.

Tal y como se ha comentado en el apartado anterior los trenes disponen de un microcontrolador que lleva implementado el protocolo de comunicación. En principio la información que se puede intercambiar con los trenes mediante ese protocolo es suficiente para realizar el control de los mismos.

Respecto a la localización de los trenes, la maqueta dispone de una serie de sensores que son capaces de detectar la presencia de un tren en un determinado punto de la maqueta. Sin embargo, estos sensores son incapaces de identificar el tren que están detectando. Lógicamente esta es una limitación que hace muy difícil el control de bajo nivel, e imposible el control de alto nivel, si no podemos identificar los trenes ¿cómo podemos planificar sus trayectorias? Para solucionar este problema se diseñaron unos prototipos que realizan tanto la función de localizar como la de identificar y que se explicarán en la próxima sección.

Finalmente, aunque se puede conseguir un módulo de MÄRKLÍN para realizar el cambio de vía a través del estándar RS-232, el hecho de utilizar para la comunicación el estándar RS-232 ralentiza excesivamente el conjunto. Por ello, se decidió construir otra placa prototipo para realizar el cambio de vía. El siguiente apartado está destinado a explicar el funcionamiento básico de estas placas prototipo.

4. Prototipos desarrollados.

En esta sección se describe el funcionamiento básico de las placas prototipo desarrolladas para hacer posible el control de la maqueta desde un PC. Primero se describe la placa que realiza el cambio de vía y en segundo lugar las placas que solucionan el problema de la identificación de los trenes.

4.1. Control de los cambios de vía

Tal y como hemos explicado en el apartado anterior, aunque inicialmente el cambio de vía estaba solucionado por MÄRKLÍN, el hecho de que utilizara el estándar RS-232 para comunicarse con el PC, hacia que el conjunto de la maqueta se comportara de manera lenta.

Por ello se optó por realizar un diseño de una placa que realizara la misma función pero que se pudiera controlar desde una tarjeta de adquisición de datos. En la figura 4 se muestra la placa que realiza esta función.

Este circuito tiene una entrada digital procedente de la tarjeta de adquisición y una salida analógica que actúa directamente sobre los cambios de vía. En función del valor de la entrada (0/1), el cambio de vía estará en una posición o otra (abierto/cerrado).

Este circuito también dispone de dos leds que indican en que posición está el cambio de vía. De esta forma es muy fácil identificar visualmente si el cambio esta en la posición deseada.

Figura 4.- Placa prototipo para el cambio de vía

4.2. Identificación de los trenes

Para solucionar el problema de la identificación de los trenes también se desarrollaron placas prototipo. Estas placas solucionan por completo tanto el problema de la localización como el de la identificación de cada tren en la maqueta.

El sistema de identificación consta de dos placas (Figura 5), una placa emisora que se coloca en la parte superior de los trenes y otra receptora ubicada en puntos estratégicos del recorrido. A continuación se pasan a describir más detalladamente ambas placas.

La placa emisora es muy sencilla. Su sencillez viene determinada por las restricciones de espacio, ya que, como se ha mencionado anteriormente, estará situada encima de los trenes. Su función es emitir constantemente un tren de pulsos a una determinada frecuencia. Con este esquema, la identificación se resuelve fácilmente si asignamos un valor frecuencia predeterminado a cada uno de los trenes. Una de las características de este circuito es que ha sido sintonizado para que el alcance del emisor de infrarrojos no sobrepase los 10 cm. Con esto se pretenden reducir posibles errores debidos a la recepción de una frecuencia emitida por los trenes que circulen por vías paralelas a la deseada. También por este mismo motivo, se han elegido unos emisores de infrarrojos con poca apertura angular (20º aproximadamente) en lugar de utilizar los emisores infrarrojos mas comunes (80º).

En cuanto a la placa receptora , como se ha mencionado anteriormente, se ubica a lo largo del recorrido de los trenes en la maqueta y se conecta al PC a través de la tarjeta de adquisición de datos. Se encarga de detectar la frecuencia del tren que pasa por delante de ella. Esta placa es un poco más compleja que la anterior, y en ella se pueden distinguir dos etapas: etapa de recepción (del tren de pulsos) y etapa de identificación (de la frecuencia).

La primera etapa, recepción del tren de pulsos, tiene como entrada un tren de pulsos en el espectro infrarrojo y como salida el mismo tren de pulsos pero ahora como señal eléctrica. Está compuesta únicamente por un fototransistor, que es el encargado de hacer esta transformación.

La segunda etapa tiene como entrada el tren pulsos (salida de la etapa anterior) y como salida tiene un valor digital (1 bit) indicando si se está detectando la frecuencia (1) o no (0). De esta forma, esta segunda etapa puede repetirse tantas veces como frecuencia s queramos detectar. Para poder realizar correctamente la identificación, cada una de estas etapas de identificación, tendrá que ser sintonizada para que identifique una de las frecuencias predeterminadas y así poder identificar uno de los trenes.

Al igual que en la placa de cambio de vía, cada etapa de identificación dispone de un led que indica cuando se está detectando la frecuencia a la que ha sido sintonizada. De esta forma es fácil identificar visualmente si se esta detectando el tren deseado o no.

En la figura 4 aparecen las dos placas prototipo, en este caso la placa receptora está hecha para soportar no más de cuatro trenes simultáneamente.

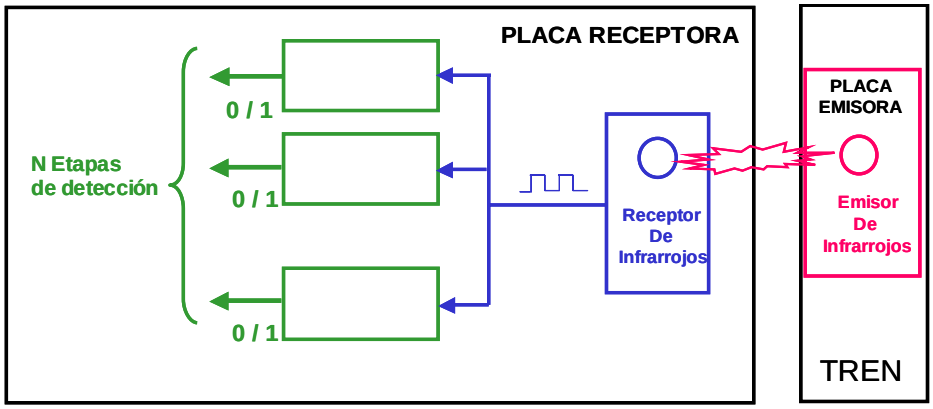


Figura 5.- Diagrama de bloques del sistema de identificación .

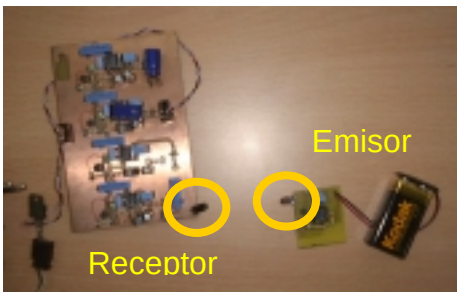


Figura 6.- Placas prototipo para la identificación de los trenes.

5. Conclusiones y trabajo futuro

Como conclusiones destacar que la puesta en marcha una maqueta de estas características es una labor totalmente artesanal. Por otra parte, aunque los fabricantes ofrecen soluciones para casi todos los problemas que surgen cuando se quiere realizar un control desde un PC, éstas no siempre se ajustan a las necesidades del usuario, siendo en la mayoría de las ocasiones soluciones demasiado caras.

Por estas razones el presente artículo presenta una serie de placas prototipo que realizan exactamente la función deseada y resultan mucho más económicas. Finalmente, la realización de prototipos propios hace que la maqueta resultante sea totalmente *adhoc* a la aplicación desarrollada.

Respecto al trabajo futuro se está pensando en incorporar a la maqueta un brazo robot que interactúe con los trenes, simulando la carga y descarga de mercancías en los distintos trenes. Otra de las posibilidades interesantes es poner semáforos en la maqueta, los cuales añadirán un poco mas de complejidad al control. Por último, y con vistas a ampliar el rango de asignaturas que puedan hacer uso de la maqueta, también se está pensando ubicar una cámara cenital que permita la aplicación de algoritmos de visión por computador para poder identificar la posición de los trenes en cada momento.

Referencias

[1] MÄRKLÍN http://www.marklin.com (USA)

iFOTON. Herramienta didáctica de bajo coste para el desarrollo de sistemas basados en microcontrolador

Norberto Cañas de Paz
Dpto. de Informática Aplicada
Universidad Politécnica de Madrid
28031 Madrid
e-mail: norberto@eui.upm.es

Gracián Triviño Barros
Dpto. Tecnología Fotónica
Universidad Politécnica de Madrid
28660 Madrid
e-mail gtrivino@fi.upm.es:

Resumen

Con este documento presentamos un entorno de desarrollo, con interés didáctico, para sistemas con microcontroladores (que hemos llamado iFOTON). Ideado y construido para preparar prototipos con el microcontrolador PIC16F873, está constituido por dos componentes fundamentales: El primero es una tarjeta PCB de propósito general y de bajo coste, en la que se ha incluido una zona pretaladrada para incluir nuevos elementos. El segundo es un programa que permite grabar la memoria no volátil del microcontrolador, sin utilizar un programador hardware externo, valiéndose para ello del puerto paralelo (IEEE-1284) del computador en el que se ejecuta. Dicho programa suministra también un conjunto de utilidades básicas para la organización de proyectos sencillos.

1. Antecedentes

Existen alternativas comerciales muy variadas para el desarrollo de sistemas con microcontroladores, muchas de las cuales tienen distintos grados de interés para la enseñanza en razón de su coste, complejidad, orientación del producto, etc. En este sentido es interesante indicar que la mayoría de los fabricantes de microcontroladores suministran placas entrenadoras, con el soporte software necesario para utilizarlas de una manera adecuada. Las referencias de web [1] [5] [7] [10] [13] [16] constituyen una pequeña revisión de los sistemas de este tipo disponibles en el mercado en este momento. Con un planteamiento específicamente educativo podemos encontrar algunos productos de Lego [8] que tienen la ventaja añadida de proporcionar las conocidas facilidades para la construcción de prototipos de estructuras mecánicas y acoplamiento de sensores.

Dentro de los sistemas construidos, para cubrir exclusivamente objetivos docentes, vale la pena destacar la controladora Enconor [17], desarrollada para alcanzar metas en el ámbito de la enseñanza secundaria, por profesores que imparten docencia en dicho nivel.

Fuera del ámbito comercial, existen antecedentes importantes en el terreno de los sistemas que permiten la programación de microcontroladores de coste reducido. Destacan posiblemente aquellos que pueden reprogramar el dispositivo sin necesidad de extraerlo del circuito en el que se encuentra alojado, como por ejemplo los planteados en las referencias [2] [14]. Sin embargo, en los casos indicados, para leer y enviar datos se utiliza el puerto paralelo de un computador personal según la especificación *Centronics*. Ello obliga a introducir componentes para contar con líneas bidireccionales de datos, distintos de los intrínsecamente necesarios para el cometido original del circuito.

Una alternativa distinta, que también permite eliminar la necesidad de adquirir un dispositivo grabador, consiste en introducir dentro del microcontrolador un programa cargador [15]. Dicho programa, al empezar a funcionar, explora si lo que se quiere hacer es cargar un nuevo programa

en la zona libre de memoria o ejecutar el programa actualmente cargado. Esta alternativa, que también es muy interesante, tiene como contrapartidas la reducción del espacio de memoria disponible para la aplicación, al tiempo que persiste el problema de grabar la memoria del componente la primera vez.

En relación con el software de grabación de la memoria de los microcontroladores, se pueden hacer las siguientes consideraciones: Por un lado los fabricantes de programadores comerciales incluyen habitualmente el software de grabación en sus productos. Dichos programas suelen tener una dependencia muy fuerte con el dispositivo grabador, por lo que habitualmente no tiene utilidad sin el mismo. Por otro lado, las alternativas que tienen como objetivo reducir el precio del grabador o que eliminan la necesidad del mismo, generalmente presentan un software de grabación de características muy limitadas, eliminando en ocasiones la posibilidad de aprovechar todas las características del microcontrolador afectado.

En el entorno que presentamos hemos eliminado la necesidad de añadir nuevo hardware al circuito de nuestro controlador, por apoyarnos en el estándar IEEE 1284 para puertos paralelos (que actualmente respetan los fabricantes de computadores de forma casi generalizada), en el que se incluye, entre otros, un modo de explotación bidireccional del bus de datos (Byte mode [6]). No hemos optado tampoco por introducir un cargador en el microcontrolador, por lo que disponemos de toda la memoria para albergar programas.

2. Introducción

Es sabido que además de enseñar conceptos es igualmente importante enseñar procedimientos que los utilicen. La aplicación mediante experimentación práctica de estos procedimientos es un aspecto clave para afianzar el aprendizaje del tipo de conocimientos manejados en nuestro campo.

Una de las dificultades a las que se enfrenta el profesor que imparte docencia en el área del desarrollo de sistemas basados en microcontrolador, es el problema de disponer de recursos eficaces y de bajo coste, que permitan realizar prácticas de diseño y desarrollo de sistemas con un planteamiento abierto y cuyos resultados hardware y software, al quedar en propiedad del alumno, dejen en sus manos la posibilidad y motivación para nuevos desarrollos personales.

Con el propósito de realizar una aportación hacia la solución de este problema, nos hemos planteado la construcción de la herramienta que presentamos, siendo sus requisitos generales los siguientes:

1. Proporcionar un circuito adecuado para la construcción de prototipos con microcontroladores.
2. Proporcionar la posibilidad de programar la memoria del microcontrolador sin la necesidad de adquirir un sistema adicional.
3. Proporcionar facilidades para la organización de la documentación del proyecto y por lo tanto para la creación de una librería de proyectos reutilizables.

La presente comunicación está organizada de la siguiente manera: en primer lugar presentamos, el circuito que proponemos para desarrollar prototipos. En segundo lugar, para facilitar la grabación de la memoria del microcontrolador y la organización del proyecto, exponemos las características más destacadas del programa construido.

3. Facilidades para el diseño y construcción del hardware del prototipo

En los desarrollos con microcontroladores existe una inversión fija, difícil de eliminar, y es la asociada con la placa que debemos utilizar para hacer prototipos. Es cierto que cuando un desarrollo se encuentra en una etapa avanzada es generalmente adecuado diseñar un circuito impreso específico, pero mientras no se alcanza dicha situación tiene interés contar con placas que tengan los componentes que consideramos fijos, así como algunas zonas pretaladradas (en la misma placa o en placas separadas), de forma que se facilite la inserción de nuevos componentes y nuevas conexiones entre los mismos. Por este motivo proponemos una placa de propósito general, en la que hemos incluido el espacio y las conexiones para los elementos que con más frecuencia se utilizan, así como también una zona pretaladrada.

En relación con el segundo requisito tiene interés reducir el precio del entorno, lo cual se puede conseguir por diferentes caminos. Nosotros

hemos valorado especialmente la posibilidad de grabar la memoria del microcontrolador utilizando niveles TTL. Esta alternativa permite utilizar tensiones fácilmente generables desde un computador personal, eliminando con ello la exigencia de adquirir o construir un dispositivo grabador.

Queremos destacar que el programa de grabación y la placa de prototipos propuesta son elementos separados que pueden utilizarse juntos o de forma independiente. El programa de grabación se puede utilizar con otras placas, siempre que se suministre la interfaz adecuada con el computador y es evidente que no es necesario grabar el microcontrolador con el programa que aquí proponemos.

Para facilitar el desarrollo de prototipos, proponemos un circuito basado en el microcontrolador PIC16F873. Con él introducimos un conjunto de componentes reducido, bien conocidos y fácilmente localizables en el mercado, al tiempo que reservamos una zona pretaladrada para insertar nuevos elementos según sean las necesidades de cada proyecto concreto.

Dicho circuito contiene los siguientes bloques principales: microcontrolador (16F873), módulo generador de niveles RS232 (MAX232), componentes de alimentación y conectores. Describimos brevemente a continuación los mismos, después de introducir las características principales del conjunto.

3.1 Características principales

Las características más importantes de la placa que presentamos son las siguientes (ver figura 1):

1. Hemos intentado promover un ahorro de tiempo a la hora de preparar los montajes que se pretendan evaluar, sin que con ello se merme excesivamente la calidad del prototipo construido.
2. Se han ubicado los conectores de los puertos paralelos cerca de la zona pretaladrada para evitar, en la medida de lo posible, la existencia de cables sobre los componentes fijos.
3. Se incluye un conector para poder efectuar la grabación del microcontrolador desde un PC (o cualquier otro dispositivo que se adapte al protocolo de grabación del PIC).
4. Para minimizar el precio de la placa y facilitar la posibilidad de establecer conexiones por la técnica de "wrapping", se han escogido tiras de *pines* como conectores, salvo para el caso del RS-232.
5. Son accesibles todas las líneas de los puertos paralelos desde sus conectores, con independencia de que algunas de ellas lleguen a otros elementos del circuito (ej. conector de grabación), por tener un cometido especial dependiendo de las circunstancias o de las operaciones que se pretendan hacer con el microcontrolador.
6. Hay conectores dentro de la placa que permiten acceder a GND.
7. Hay un conector dentro de la placa que permite acceder a las tensiones de alimentación existentes antes y después del regulador (7805).

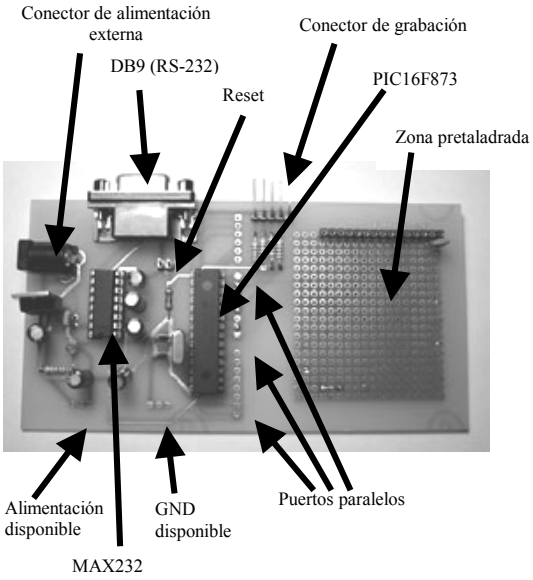


Figura 1. Placa iFOTON con componentes.

3.2 Microcontrolador

El microcontrolador seleccionado es el PIC16F873 de la empresa Microchip [10], [12]. Es difícil defender con detalle, hoy en día, una elección de este tipo frente a otras alternativas, dada la gran variedad de microcontroladores que hay en el mercado con posibilidades extraordina-

rias. Nos limitaremos a comentar algunos aspectos destacados del mismo. Su arquitectura es RISC y tiene integrados los siguientes periféricos: convertidor analógico digital de 10 bits, generador de señales PWM, módulo comparador y capturador, módulos de comunicaciones (USART, I2C), 3 puertos paralelos y 3 timers.

Tiene 3 bancos de memoria separados. Un banco de 4K palabras de 14 bits de memoria FLASH para código. Un banco SRAM de 192 bytes de datos y un banco EEPROM de 128 bytes de datos (regrabable en tiempo de ejecución y en tiempo de programación).

Como característica notable del PIC16F873 (que también tienen otros microcontroladores del mismo fabricante) hay que destacar la posibilidad de grabar su memoria con niveles bajos de tensión (Low Voltage in Circuit Serial Programming [9]), lo que permite programar el microcontrolador, incluso en el circuito en el que se encuentra insertado, sin recurrir a los habituales niveles de 13 o 14 Voltios. En concreto, podemos generar las señales necesarias para la lectura y escritura desde el puerto paralelo de un computador, como más adelante comentaremos al hablar del programa de grabación.

Para terminar este apartado queremos destacar que el microcontrolador seleccionado ofrece una excelente relación calidad precio como demuestra la gran difusión que tiene actualmente en el mercado.

3.3 Módulo generador de niveles RS-232

Como se ha indicado el microcontrolador proporciona una USART que respeta la norma RS-232, pero genera señales con niveles TTL. Para adaptar la amplitud de las mismas a los niveles que permite la norma, se incluye el componente MAX232. Sobra decir que si no es necesario utilizar estas facilidades de comunicaciones, en el prototipo que vamos a desarrollar, podemos utilizar la placa que proponemos prescindiendo del MAX232 y de los condensadores asociados, sin ningún perjuicio para el funcionamiento del resto del circuito.

3.4 Componentes de alimentación

Para alimentar la placa hemos considerado interesante poder utilizar transformadores de los que habitualmente se suministran con los altavoces de los PC's o aparatos similares. Por ello hemos incluido un conector tipo *jack*, seguido de un regulador de tensión 7805 (Fairchild). Se ha incluido también un *led* que nos permite comprobar si la placa está alimentada o no.

3.5 Conectores

Para poder acceder a los diferentes servicios, que suministran los elementos anteriormente mencionados, ha sido necesario dotar al circuito de un conjunto de conectores que permita con facilidad explotar sus posibilidades.

Pueden dividirse de la siguiente manera:

1. Conectores de puertos paralelos. Permiten acceder a los puertos paralelos del microcontrolador.
2. Conector de comunicaciones RS-232. Es un conector del tipo D-Shell (DB9) por el que se puede acceder a las líneas de comunicación suministradas por el componente MAX232.
3. Conector de grabación serie. Por medio del cual podemos activar los servicios de lectura y grabación de la memoria no volátil del microcontrolador desde un programador externo (ej. un computador personal).
4. Conector de alimentación. Permite el acceso a las tensiones disponibles en la placa de circuito impreso: La tensión regulada suministrada por el componente 7805 o la que directamente suministre la fuente de alimentación externa.
5. Conector de tierra. Permite acceder a GND para cubrir las necesidades de tensión de referencia de los nuevos componentes que necesitemos añadir.

4. Facilidades para grabar la memoria y para la organización del proyecto.

El programa de grabación permite borrar, leer y grabar los bancos de memoria no volátil del microcontrolador (PIC16F873). Para ello es necesario conectar el puerto paralelo del computador con el conector de grabación de la placa para prototipos anteriormente presentada y utilizar las diferentes opciones que brinda el programa desarrollado.

eguidamente exponemos los aspectos más destacados relacionados con la utilización de dicho programa.

4.1 Grabación con niveles bajos de tensión

El microcontrolador PIC16F873 permite grabar su memoria con niveles bajos de tensión. En su palabra de configuración existe un bit que determina si está activada esta opción (LVP), o por el contrario hay que utilizar el procedimiento ordinario que exige tensiones del orden de 13.5 Voltios. El componente viene configurado de fábrica con el bit LVP activado.

Hay una pequeña contrapartida cuando se activa el bit LVP y es que la patilla RB3 del microcontrolador deja de ser un *pin* de propósito general en el puerto paralelo B, dado que interviene en el protocolo de entrada en modo grabación. Conectando dicha patilla a GND garantizamos que el microcontrolador no entra en modo grabación, ejecutando al arrancar el programa cargado en memoria. Por ello, el conector de grabación ha sido diseñado de tal manera que cuando no está acoplado el cable se pueda insertar un puente que precisamente establece esta conexión (Figura 2).

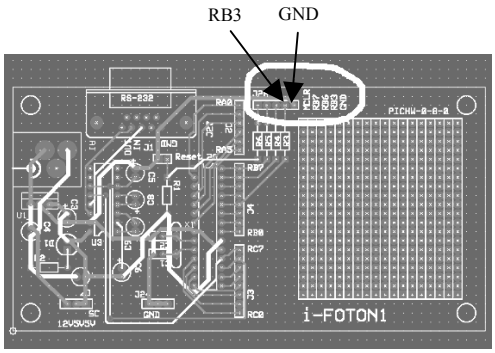


Figura 2. Diseño físico iFOTON.

4.2 Cable de conexión entre el computador y la placa de prototipos

Para entrar en modo programación es necesario que pasen a nivel bajo los pines RB6, RB7, RB3 y MCLR del microcontrolador, seguidamente debe pasar a nivel alto el pin RB3 y posteriormente MCLR (los detalles de temporización pueden verse en [12]). En modo programación el pin RB6 pasa a desempeñar funciones de reloj de sincronización y el pin RB7 pasa a ser de datos. Este último (RB7) debe ser bidireccional, dado que se utilizará en el sentido Computador→Placa cuando se quieran enviar comandos y datos al microcontrolador, mientras que el sentido será Placa→Computador cuando se realicen operaciones de lectura de la memoria.

Cuando el microcontrolador funciona ejecutando el programa que tiene cargado en memoria, los pines RB6 y RB7 vuelven a ser de propósito general, por lo que pueden ser utilizados libremente en nuestros proyectos.

Por lo dicho anteriormente, las señales que debemos contemplar a la hora de construir el cable de conexión con el computador son cuatro, debiendo añadirse un quinto cable para GND. Para que las operaciones de lectura, borrado y grabación se realicen correctamente con nuestro programa es necesario establecer las conexiones que planteamos a continuación (Tabla 1), en el supuesto de que el conector del puerto paralelo del computador sea del tipo D-Shell 25 (el más habitual)[(1)].

PIN puerto paralelo	Conector grabación PIC
1	RB6 (Reloj)
2	RB7 (Datos)
16	RB3 (Control)
17	MCLR
18, 19, 20, 21, 22, 23, 24 y 25	GND

Tabla 1. Conexiones del cable de grabación.

4.3 Arranque del programa

Lo primero que realiza el programa al arrancar es una prueba de velocidad del computador. La información obtenida es utilizada para realizar una correcta temporización de las señales, y así respetar las restricciones que impone el fabricante del microcontrolador. Esta forma de actuar ha sido

(1) Para realizar el cable hemos utilizado par trenzado y no hemos tenido problemas con longitudes inferiores a 2 metros.

necesario implantarla por haber tenido problemas al utilizar los servicios suministrado por el nivel API del sistema operativo utilizado (la versión actual funciona sobre Windows 95-98 y con un sistema de desprotección de puertos también en NT y Windows 2000. Las siguientes versiones que se elaboren tendrán como soporte adicional plataformas LINUX).

Una vez realizada la prueba de velocidad, el sistema abre una ventana en la que puede accederse a dos menús desplegables. El primero de ellos permite administrar la estructura de información de los proyectos que desarrollemos (cuya utilización es opcional), mientras que el segundo permite acceder a los servicios de programación del microcontrolador propiamente dichos.

4.4 Menú de proyectos

Dado que uno de los cometidos del entorno de desarrollo precisamente es ser una herramienta de carácter didáctico que facilite la realización de prácticas con microcontroladores, consideramos adecuado fomentar ciertas costumbres metodológicas en los futuros ingenieros que formamos. El motivo del presente menú es obligar a los alumnos a completar y documentar una serie de etapas (requisitos, diseño, etc.) antes de abordar la programación del microcontrolador.

Las opciones disponibles son: “Crear o modificar proyecto”, “Abrir proyecto” o “Configurar aplicación”.

“Crear o modificar proyecto” permite indicar qué ficheros contienen información relativa a un proyecto en concreto. Los proyectos creados en esta aplicación pueden estar constituidos por los ficheros: programa fuente, requisitos, diseño del proyecto, diagramas del circuito, fichero adicional y fichero ejecutable.

“Abrir proyecto” permite obtener el nombre de los ficheros que tienen información de un proyecto así como editarlos, invocando automáticamente a los programas de edición que se han especificado para tal efecto en la opción “Configurar aplicación”.

4.5 Menú programador

Este menú permite efectuar las operaciones de borrado, lectura y grabación de la memoria del microcontrolador.

Las opciones disponibles son: “Configurar puerto paralelo”, “Grabar microcontrolador”, “Leer microcontrolador” y “Borrar microcontrolador”.

“Configurar puerto paralelo” es una opción que posiblemente no sea necesario utilizar nunca, pero se suministra para dar cobertura a posibles situaciones extraordinarias. Por defecto el programa asume que los registros del controlador del puerto paralelo están ubicados a partir de la dirección $378. Si este no es el caso, esta opción suministra otras direcciones habituales para poder elegir, así como la posibilidad de especificar una distinta a todas las ofertadas.

“Grabar microcontrolador” solicita al usuario el nombre del fichero que contiene el código ejecutable a grabar (sí existía un proyecto abierto toma como elección por defecto el fichero ejecutable del mismo) pudiendo especificarse también dentro de dicho fichero el contenido de la memoria de datos EEPROM[(1)]. El código ejecutable debe suministrarse respetando el formato INHX8M de Intel [11], opción que normalmente suministran los ensambladores y compiladores que generan código para los microcontroladores PIC.

La opción “Leer microcontrolador” permite cargar en el microcontrolador el contenido de la memoria de código, de la memoria EEPROM, la palabra de identificación y la palabra de configuración del microcontrolador. Al activar esta utilidad la aplicación abre una nueva ventana en la que se muestran los datos recogidos, al tiempo que se da la opción de guardarlos en un fichero (con formato INHX8M) por si posteriormente los queremos utilizar para una nueva grabación.

Por último, “Borrar microcontrolador” es una opción que nos permite borrar de manera independiente la zona de código (FLASH), la de datos (EEPROM), o borrar un microcontrolador protegido. En este último caso es el propio microcontrolador quien garantiza la confidencialidad del código y los datos almacenados, dado que antes de desproteger los bancos de memoria los borra.

---

(1) Al grabar no se pueden especificar los 16 primeros bytes (de los 128 disponibles en la EEPROM), dado que en el proceso de escritura el micro utiliza un espacio de direccionamiento, imagen de la memoria EEPROM, que tiene reservados dichos bytes para otros cometidos.

5. Experiencia docente con el sistema presentado

El entorno que hemos presentado ha sido ya utilizado con éxito para la realización de prácticas en la Facultad de Informática de la Universidad Politécnica de Madrid. Concretamente en las asignaturas "Instrumentación y Adquisición de Datos" de 4º curso y "Dispositivos Optoelectrónicos Aplicados a la Informática" de 5º curso. En la primera los alumnos desarrollan un prototipo de estación meteorológica, utilizando el sistema presentado en combinación con la herramienta software LabView de National Instruments. En la asignatura de 5º curso los alumnos deben construir un analizador de la señal infrarroja enviada por distintos mandos a distancia.

Para construir la placa, el estudiante debe seguir un plan detallado, disponible en la página de web de la asignatura [3]. En dicho documento se especifican los pasos a seguir y comprobaciones a realizar, durante el proceso de montaje del circuito sobre la placa para prototipos.

Una descripción más detallada de las aplicaciones didácticas de iFOTON se puede encontrar en [4]

6. Conclusiones

La herramienta presentada ha demostrado ya su versatilidad en pequeños proyectos desarrollados por los autores (Contador de consumo de energía eléctrica, Control de posición del carro de una impresora, Control de velocidad en un motor D.C. con carga variable, etc.) así como en la realización de las prácticas citadas en asignaturas de nivel universitario.

Con estas primeras experiencias se verifican la consecución de los objetivos planteados al comprobarse una mayor facilidad para desarrollar prototipos con el microcontrolador seleccionado, al tiempo que el entorno demuestra tener interés didáctico por permitir realizar prácticas reales con microcontroladores a un precio realmente reducido. Por otro lado, ha sido satisfactorio comprobar como los mismos alumnos han propuesto la reutilización de la placa montada en un curso, para realizar las prácticas del curso siguiente.

Hemos comprobado también el efecto motivador que produce la actividad de montaje del circuito, en la que se incluye la soldadura de componentes y el proceso de verificación.

Como líneas de evolución planteadas está en estudio incorporar al entorno la posibilidad de utilizar otros microcontroladores de la misma casa, así como desarrollar la correspondiente versión para plataformas LINUX.

A partir de los desarrollos de los primeros usuarios de la herramienta, estamos creando una base de datos compartida de proyectos reutilizables hardware-software y acumulando experiencia para la mejora de la sencilla herramienta de gestión de documentación disponible.

Referencias

[1] AMD. *Embedded Processors*. http://www3pub.amd.com/products/epd/. AMD, 28-1-2002.

[2] Antonio Serna y José Vicente García. *Desarrollo y Construcción de Prototipos Electrónicos*. (pag 59-77). Ed. Paraninfo, 2000.

[3] Gracián Triviño. *iFOTON*. htttp://www.dtf.fi.upm.es/~gtrivino/IFOTON, 28-1-2002.

[4] Gracián Triviño y Norberto Cañas. *Recursos didácticos para el diseño y construcción de instrumentación digital a coste reducido*. (Enviado para su aceptación a: SAAEI2002. Seminario Anual de Automática, Electrónica industrial e Instrumentación. Alcalá de Henares, Septiembre 2002)

[5] IBM. *IBM Microelectronics*. http://www-3.ibm.com/chips/products/. IBM, 28-1-2002.

[6] IEEE. *IEEE Standard Signaling Method for a Bidirectional Parallel Peripheral Interface for Personal Computers*. Estándar IEEE 1284-2000 (Revision of IEEE Std 1284-1994), 2000.

[7] Intel. *Embedded Products*. http://www.intel.com/design/embprod.htm. Intel Corporation, 31-12-2002.

[8] Lego. *MindStorm*. http://mindstorms.lego.com/. Lego Group, 28-1-2002.

[9] Microchip. *EEPROM Memory Programming Specification*. Documento DS39025E. Microchip Techno-logy Inc. 2000.

[10] Microchip. *Microchip Product Lines*. http://www.microchip.com. Microchip Techno-logy Inc. 28-1-2002.

[11] Microchip. *MPASM USER'S GUIDE with MPLINK and MPLIB*. Microchip Technology Inc. 1999.

[12] Microchip. *PIC16F87X. 28/40-pin 8-Bit CMOS FLASH Microcontrollers*. Documento DS30292B Microchip Technology Inc. 1999.

[13] Motorola. *Design Resources*. http://e-www.motorola.com/. Motorola, 28-1-2002.

[14] Robert Spur. *A PC-Based Development Programmer for the PIC16C84*. Aplication Note AN589. Microchip Technology Inc. 1999.

[15] Rodger Richey. *Downloading HEX files to PIC16F87X PIC Microcontrollers*. Documento DS91025a, Microchip Technology Inc. 1998.

[16] Texas Instruments. *DSP Developers' Village*. http://dspvillage.ti.com/docs/dspvillagehome.jhtml. Texas Instruments, 28-1-2002.

[17] *Controladora Educativa ENCONOR*. http://www.enconor.com/ 31-12-2001.

Experiencia docente en el desarrollo de aplicaciones empotradas con MarteOS

Silvia Terrasa, Patricia Balbastre, Alfons Crespo
Dept. de Informática de Sistemes y Computadores
Universidad Politécnica de Valencia
46022 Valencia
e-mail: {sterrasa, patricia, alfons}@disca.upv.es

Resumen

Este trabajo presenta la experiencia de realizar trabajos basados en sistemas empotrados, para la asignaturas relacionadas con sistemas de tiempo real. Las herramientas software utilizadas para el desarrollo de los trabajos son el lenguaje de programación Ada'95 y el *microkernel* de tiempo real MarteOS desarrollado en la Universidad de Cantabria.

1. Introducción y motivación.

Actualmente, uno de los aspectos que deben cubrir las enseñanzas universitarias es el de proporcionar al alumno escenarios de trabajo lo más cercanos posible a la realidad. La utilización de entornos reales, no obstante, no debe ser precipitada, ya que no hay que olvidar que para que los alumnos puedan *exprimir* al máximo estas experiencias, primero hay que formarlos con una base sólida. Una vez claros los conocimientos básicos, ya se pueden acabar consolidar con ejemplos reales.

Este es el planteamiento de la mayoría de los planes de estudios que hay actualmente en vigor en las universidades españolas, los cuales se dedican los primeros semestres de las titulaciones a impartir asignaturas básicas con fuerte contenido teórico para finalmente dedicar los últimos semestres a asignaturas optativas con un fuerte carácter práctico. En el marco de las asignaturas prácticas es dónde tiene más sentido ofrecer ejemplos o situaciones reales.

Con esta idea en la cabeza, los profesores implicados en las asignaturas de sistemas de tiempo real (las cuales son optativas de último curso), estamos siempre dispuestos a introducir elementos "reales" en la elaboración tanto de las prácticas como de los trabajos de asignatura [1][2][3].

Este artículo presenta la última experiencia que hemos tenido en esa búsqueda de escenarios y ejemplos reales. Se trata básicamente de la incorporación a la asignatura de aspectos de programación de sistemas empotrados. Para poder llevar a cabo esta labor ha sido indispensable la utilización de herramientas de soporte, en concreto hemos utilizado un *microkernel* de tiempo real llamado MarteOS, el cual ha sido desarrollado por el departamento de Electrónica y Computadores de la Universidad de Cantabria [4]. Por otra parte también ha sido necesario utilizar el lenguaje de programación de tiempo real Ada'95. Finalmente y para hacer más atractivos los trabajos, éstos tenían cómo objetivo el realizar el control procesos físicos reales.

El resto del artículo está esquematizado de la siguiente manera, en la sección 2 se verá con más detalle el marco y los objetivos de las asignaturas implicadas. En la sección 3 se describirán las herramientas utilizadas en el desarrollo de los trabajos para pasar en la sección 4 a describir los trabajos así cómo su planificación en términos de objetivos a desarrollar por los alumnos. Finalmente en la sección 5 concluiremos y presentaremos una serie de retos futuros.

2. Marco y objetivos de STR[1].

La asignatura objeto del presente artículo es la de Sistemas de tiempo real (STR). Actualmente esta asignatura se imparte en cuatro titulaciones distintas: Ingeniero Industrial, Ingeniero en Automática y Electrónica Industrial, Ingeniero en Informática, Ingeniero Técnico en Informática (en la especialidad de sistemas). En todas ellas se imparte como asignatura optativa de último curso excepto en el caso de Ingeniero en Automática y Electrónica que es una asignatura obligatoria también de último curso.

A pesar de la diversidad de las titilaciones donde se imparte, en todas ellas la podemos enmarcar dentro de lo que se denomina Ingeniería de Control y los objetivos más relevantes de ésta son saber diseñar e implementar sistemas de control. Las asignaturas de control, que se ven en todas las titulaciones, aportan el conocimiento y la habilidad para comprender y analizar los procesos industriales y los métodos y técnicas para diseñar e implementar el sistema de control. Los ingenieros deben especializarse en muchos aspectos como:

- Teoría de Control.
- Modelización.
- Simulación.
- Técnicas avanzadas de control.
- Programación de propósito general.
- Planificación.

A parte de los objetivos generales de las asignaturas que se engloban en la llamada Ingeniería de Control, la asignatura de STR tiene una serie de objetivos concretos que podemos resumir en:

- Entender las restricciones de tiempo y los elementos que intervienen en el desarrollo de aplicaciones controladas por computadora.
- Diseñar e implementar aplicaciones de control por computadora.
- Conocer las restricciones que imponen los sistemas empotrados.
- Conocer los lenguajes de alto nivel y sus abstracciones para implementar la concurrencia, comunicación y control del tiempo (Ada'95).
- Conocer la influencia del sistema operativo y los algoritmos de planificación.
- Conocer y asimilar las técnicas de análisis de planificabilidad.

Otra característica u objetivo común que tienen esta asignatura en las diferentes titulaciones es el deseo, por parte tanto del profesorado como del alumnado, de estar en contacto con entornos de desarrollo y procesos reales. Tal y como se planteaba en [1] nos resulta especialmente interesante que el alumno pueda enfrentarse con situaciones reales tales como el control de un depósito de agua o de un robot móvil. Por otra parte estamos interesados en ofrecerle al alumno herramientas que le permitan realizar sus propias pruebas [2]. En curso actual hemos decidido introducir un elemento más a la cadena y permitir que desarrollen aplicaciones para ser ejecutadas en un computador empotrado. Para ello hemos utilizado el *microkernel* de tiempo real llamado MarteOS [4] que en el siguiente apartado pasamos a detallar.

3. Entorno de desarrollo de MarteOS.

MarteOS es un *microkernel* de tiempo real desarrollado en el Departamento de Electrónica y Computadores de la Universidad de Cantabria. Este *microkernel* sigue el subconjunto mínimo POSIX de tiempo real (POSIX.13) el cual ofrece una interfaz estándar para aplicaciones escritas tanto en Ada como en C. En este *microkernel* la mayoría del código está escrito en Ada, aunque también hay partes realizadas en C.

MarteOS trabaja en un entorno de desarrollo cruzado formado por un PC con Linux actuando como *host* y una máquina desnuda basada en la arquitectura del 386 que actúa como *target*, como muestra la figura 1 ambos sistemas se conectan mediante una red Ethernet, para realizar el arranque de la máquina, y mediante una línea RS-232 para las tareas de depuración remota.

[1] STR son las siglas de Sistemas de Tiempo Real.

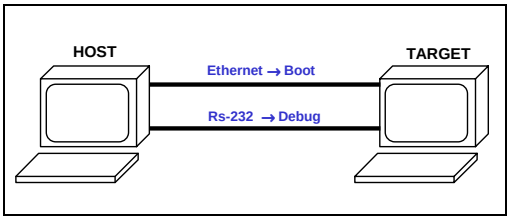


Figura 1.- Entorno de desarrollo cruzado de MarteOS

El ciclo de desarrollo de las aplicaciones sigue los siguientes pasos:

- Se arranca el PC *target* con el disquete de arranque generado en la instalación de MarteOS.
- La aplicación se compila y se enlaza en el PC *host*.
- En el PC *target* el programa de arranque se "baja" la aplicación a través de la conexión de red y la pone en ejecución.
- La aplicación puede ser ejecutada normalmente o puede ejecutarse en modo depuración remota.
- Tan pronto como la aplicación termina el programa de arranque del PC *target* se queda esperando para "bajarse" y ejecutar la siguiente aplicación.

El desarrollo cruzado da la oportunidad al alumno de implementar sus aplicaciones en un entorno conocido para luego probarlo en el entorno real.

Respecto a otros entornos de programación reales utilizados anteriormente en la realización de los trabajos, como RT-Linux [2], la ventaja que ofrece MarteOS es fundamentalmente la posibilidad de desarrollar todas las aplicaciones en Ada'95, que es el lenguaje que ven los alumnos durante todo el curso.

En el siguiente apartado se describen los trabajos planteados el presente curso.

4. Planteamiento de los trabajos.

Como se comentaba en [1], para realizar la evaluación de la asignatura de Sistemas de Tiempo Real se le ofrece al alumno la posibilidad de realizar un trabajo de asignatura en vez de realizar un examen escrito al final de la misma. Para que esta opción les resulte viable, se debe proporcionar un horario de laboratorio suficiente para que puedan desarrollar los trabajos durante el tiempo de laboratorio.

Actualmente la asignatura dispone de 6 créditos, 4,5 de teoría (que incluye 1.5 créditos de problemas de aula) y 1.5 créditos de prácticas de laboratorio. Esta distribución de créditos proporciona tres horas de teoría a la semana (repartidas en dos clases de una hora y media) y dos horas de prácticas cada quince días. Obviamente con dos horas cada quince días no se pueden realizar las prácticas y trabajo de asignatura, por tanto, lo que se hace es destinar ese crédito y medio de problemas de aula a sesiones de laboratorio, de manera que puedan tener dos horas a la semana más de laboratorio. Con este cambio y reduciendo al mínimo el número de prácticas (actualmente sólo realizan cuatro) conseguimos tener el tiempo suficiente para que los alumnos realicen los trabajos antes del periodo de exámenes.

Respecto a las tareas a realizar en los trabajos, normalmente se debe realizar el control de un proceso físico, como robots móviles, depósitos de agua, brazos robot, etc. A continuación se detallan un par de los trabajos que se han planteado el presente curso.

4.1. Control de un brazo robot con MarteOS.

Uno de los trabajos propuestos durante el presente curso fue realizar el control de movimiento de un brazo robot de cuatro grados de libertad (Figura2). El robot dispone de cuatro motores, uno por articulación y también lleva incorporados los finales de carrera. Todos la entradas y las salidas del brazo robot son digitales.

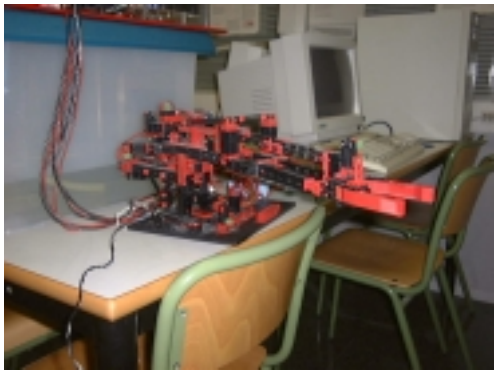


Figura 2.- Brazo robot.

Para realizar el control desde el PC se conectaron las salidas y entradas digitales del robot a una tarjeta de adquisición de datos para que fueran accesibles desde el PC.

Debido a que muchos de los alumnos no disponen de conocimientos de robótica, no se les puede decir que realicen control de movimiento del brazo robot por cinemática inversa (dado un punto en el sistema de coordenadas global hacer que la pinza del brazo robot se posicione en el). No obstante, aún se les puede pedir que muevan las distintas articulaciones teniendo en cuenta los finales de carrera de cada una de ellas.

4.2. Control de un robot móvil con MarteOS.

Otro de los trabajos que propuesto este curso es el control de un robot móvil (Figura 3). El robot en cuestión mide 27 mm de radio y 2 cm de espesor. Está provisto de dos ruedas, controladas por dos motores de corriente continua. Además, estas ruedas tienen cada una un encoder incremental que proporciona información de odometría. El sistema de sensores lo forman ocho sensores de infrarrojos con un alcance de 60 mm, situados a lo largo del perímetro del robot.

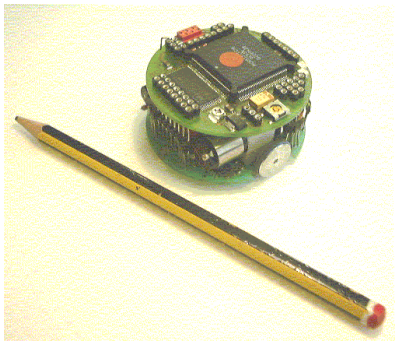


Figura 3 Robot móvil Khepera.

El trabajo consiste en que el robot debe seguir un haz de luz (proporcionado por una linterna) para ello deben analizar la información de los sensores de infrarrojos para determinar si hay luz y que deben hacer para encararse a ella y avanzar, siguiéndola.

Previamente se había utilizado el mismo robot [1] pero con otro tipo de control, en este caso lo que debía hacer el robot era alcanzar objetivos (introducidos por el usuario) evitando todos los obstáculos que se encontrara por el camino. Esta es otra alternativa para realizar trabajos, aunque en el caso de MarteOS las aplicaciones sin E/S funcionan mucho mejor debido al tiempo que se pierde realizando este tipo de operaciones. En el caso del seguimiento de la luz, el robot es mucho mas autónomo y las acciones de usuario no implican ninguna operación de E/S, únicamente han de mover la linterna para que el robot reaccione.

5. Conclusiones y trabajo futuro.

La iniciativa de introducir la programación de sistemas empotrados en los trabajos de asignatura ha sido un éxito entre el alumnado. Este tipo de trabajos prácticos tienen como objetivo que el alumno se enfrente con problemas reales. La experiencia ha sido totalmente satisfactoria, ahora no sólo disponen de procesos reales con los que inter-actuar, sino que el medio a través del que interactúan también está mucho mas cercano al mundo real.

Como trabajo futuro se va a plantear la posibilidad de desarrollar todas las aplicaciones de la asignatura en este entorno cruzado de programación. La idea es que el alumno desde su primera práctica (un simple *Hola Mundo*), utilice el entorno de desarrollo de MarteOS, para que su adaptación sea más paulatina. Esto permitirá que trabajen durante todo el curso con un entorno de tiempo real.

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. "Time Logger: una herramienta software de soporte a la docencia de los sistemas operativos de tiempo real", JENUI'2001, pp 173-178.

[3] S. Terrasa, P. Balbastre, A. Crespo. "Estudio de técnicas de Planificación en Sistemas de Tiempo Real de Laboratorio", Jornadas de Automática 2000, ISBN 84-699-3163-6

[4] MarteOS : http://marte.unican.es

[5] A. Crespo, J. Vila, F. Blanes, I. Ripoll. "Real-Time Education in a Control Engineering Curriculum". Real-Time Systems Education. IEEE Computer Society Press.

Sistemas distribuidos y paralelos

EDIPO: Un entorno para el desarrollo de aplicaciones en entornos distribuidos[1]

F. Almeida, D. González, L. M. Moreno, C. A. De Pablos
Dpto. de Estadística, Investigación Operativa y Computación
Universidad de La Laguna
38271, La Laguna. Tenerife
e-mail: {falmeida, dgonmor, lmmoreno}@ull.es

Resumen

Se presenta una herramienta informática que facilita el desarrollo de aplicaciones paralelas en entornos de redes heterogéneas. El usuario obtiene una versión unificada del conjunto de máquinas sobre el que ejecutará su aplicación. Para garantizar su portabilidad, EDIPO ha sido desarrollado con herramientas estándar en los sistemas Unix/Linux. Facilita la labor de edición, compilación y ejecución de aplicaciones desde una única consola de trabajo.

1. Introducción

En los actuales planes de estudios de los títulos de Ingeniero en Informática es habitual encontrar asignaturas con contenidos relacionados con la Programación Distribuida o la Programación en Paralelo [7]. En ellas el alumno debe desarrollar prácticas sobre máquinas de laboratorios que pueden estar geográficamente dispersos, gestionados por distintos servidores y con diferentes sistemas de archivos conectados a través de una red.

En el desarrollo de las prácticas de estas asignaturas el alumno debe mantener abiertas sesiones de trabajo, con distintos nombres de usuario, sobre varios servidores, en las que combina la edición local o remota, la transferencia de archivos, hacia y desde los distintos sistemas, y la compilación y ejecución distribuida de las aplicaciones.

Se denomina máquina virtual al conjunto de ordenadores interconectados sobre el que el alumno va a ejecutar su aplicación. Cada uno de estos ordenadores (llamados nodos o *hosts*) puede estar situado en una localización distinta y pertenecer a redes independientes. Esta máquina virtual puede ser homogénea si todos los procesadores son idénticos y comparten el mismo sistema operativo o bien puede ser heterogénea si existen procesadores de más de una arquitectura o de más de un sistema de archivos. En este último caso los comandos necesarios para compilar y ejecutar la aplicación pueden variar de un sistema a otro. La herramienta que se presenta, EDIPO (*Environment for DIstributed prOgramming*) [7], constituye un entorno de desarrollo que permite realizar esta tarea. Proporciona todos elementos necesarios para editar, configurar, compilar y ejecutar una aplicación en una máquina virtual que puede ser heterogénea. EDIPO permite construir aplicaciones que hacen uso simultáneamente de varios procesadores desde una única máquina central (*host* de desarrollo) sin tener en cuenta cómo y dónde están distribuidas el resto de las máquinas que conforman la máquina virtual, y ejecutarlas utilizando diferentes herramientas de programación paralela.

La herramienta EDIPO está formada por una `GUI` implementada en `Tcl/Tk` [10] y un conjunto de *scripts* que se han escrito en `tclsh`, que utilizan los mecanismos de comunicación remota `rcp` y `rsh`. Tanto el lenguaje de implementación `Tcl/Tk` como los mecanismos de comunicación remota están habitualmente

[1] El trabajo descrito en este artículo ha sido parcialmente financiado por el Ministerio Español de Ciencia y Tecnología (CICYT). TIC1999-0754-C03

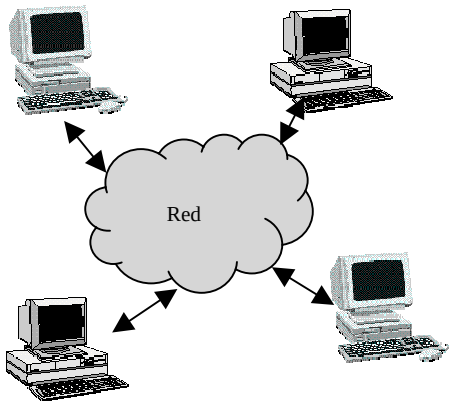


Figura 1. Máquina Virtual

instalados en la mayoría de las versiones de Unix/Linux [5, 8], lo que hace que este entorno sea fácilmente portable a la mayor parte de estos sistemas.

Dos entornos que proporcionan funcionalidades similares a EDIPO son PADE (*Parallel Application Development Environment*) [3] y WAMM (*Wide Area Metacomputing Manager)* [1].

PADE es un entorno construido por el Instituto Nacional de Estándares y Tecnología (NIST) del Gobierno Federal de Estados Unidos en 1995. Proporciona una interfaz gráfica que facilita al usuario el desarrollo de aplicaciones paralelas basándose en la librería de paso de mensajes PVM (*Parallel Virtual Machine*) [9]. Este entorno permite editar archivos, transferir un conjunto de archivos a cada nodo de la máquina virtual y compilar y ejecutar en cada uno de sus nodos.

WAMM es una interfaz gráfica basada en OSF/Motif y PVM y que ha sido desarrollada por el Grupo de Investigación en Procesamiento Paralelo del CNUCE, Pisa, en 1995. Permiten al usuario definir y gestionar una máquina virtual, tener una visión geográfica del sistema, ejecutar comandos remotos y realizar tareas de control y monitorización.

PADE y WAMM obligan a instalar y ejecutar PVM 3.0 en cada uno de los nodos, por lo que se produce un alto consumo de recursos en las máquinas y, además, la aplicación debe estar escrita en PVM. La versión actual de EDIPO soporta la ejecución sobre la librería de paso de mensajes MPI [6] y puede ser fácilmente extendida para la ejecución con herramientas como PVM, Corba o Java RMI. Lo que proporciona una flexibilidad y generalidad superior a la de los entornos similares.

Entornos más complejos como Condor [2] o Globus [4], abordan también el problema desde una perspectiva aún más amplia. Sin embargo, se trata de entornos que implican un esfuerzo de instalación, aprendizaje y un consumo de recursos considerable para poder hacer un uso efectivo de ellos a corto plazo en docencia.

Este trabajo está organizado de la siguiente manera: En la sección 2 se describe lo que denominamos una máquina virtual. En la sección 3 se muestra el entorno de desarrollo que se presenta, cómo configurar la máquina virtual y las operaciones que pueden realizarse a través de esta herramienta. En la sección 4 se explican los pasos necesarios para su instalación. Se finaliza el trabajo en la sección 5 con algunas conclusiones y trabajos futuros.

2. La Máquina Virtual y las Aplicaciones

En el contexto que nos ocupa, una máquina virtual está formada por un conjunto de ordenadores o nodos que están unidos mediante una red de interconexión (Figura 1). Cada ordenador ejecuta su propio programa, accede a su memoria local y se comunica a través de la red con el resto de los nodos. Habitualmente el desarrollo de una aplicación en este entorno implica algún mecanismo de intercambio de mensajes.

La aplicación tendrá componentes que residen en sistemas de archivos de diferentes nodos con distintos sistemas operativos. Esas componentes tendrán que compilarse en cada nodo, normalmente con diferentes directivas y opciones de compilación. Las versiones compiladas se ejecutarán en cada nodo, y los flujos de entrada-salida se gestionarán en cada uno de ellos.

Dado que las fases de compilación y ejecución pueden tratarse de forma genérica, cualquiera de las plataformas software de computación paralela puede beneficiarse de EDIPO, puesto que proporciona al desarrollador una interface de alto nivel que simplifica las etapas de desarrollo de la aplicación desde una única consola de trabajo.

3. El Entorno de Desarrollo EDIPO

EDIPO es un entorno reconfigurable para el desarrollo de aplicaciones sobre una máquina virtual heterogénea compuesta de varias arquitecturas y/o varios sistemas de archivos. Utiliza los mecanismos de comunicación remota `rsh` y `rcp`. Proporciona un marco de trabajo para todas las fases de desarrollo de una aplicación paralela: edición, configuración, compilación y ejecución. El paquete está compuesto por una interface gráfica de usuario y un conjunto de *scripts* escritos en `Tcl/Tk`.

El entorno dispone de una consola de desarrollo para la máquina virtual, de forma que todas las tareas del desarrollo pueden realizarse desde la máquina de desarrollo (Figura 2). La consola permite:

- Definir la máquina virtual, identificando las máquinas que van a ejecutar los distintos componentes de la aplicación paralela.
- Organizar los archivos fuente, un conjunto de archivos que puede ser diferente para cada nodo de la máquina virtual.
- Editar el código fuente apropiado para cada máquina mientras se mantienen todos los archivos en una localización común
- Transferir el conjunto de archivos entre los nodos, cuando sea necesario.
- Compilar el conjunto de archivos en cada nodo, el compilador para cada arquitectura, o para cada nodo, puede variar.
- Ejecutar el programa.
- Comprobar la salida de la compilación y ejecución.

3.1. Configuración de la máquina

La figura 3 muestra la pantalla principal de EDIPO. En el título de la ventana se observa el proyecto que se encuentra abierto en cada momento. Bajo el título se presenta un menú que permite realizar la gestión y el mantenimiento de estos proyectos. En el menú *Edit* se encuentran las opciones para gestionar los proyectos, los archivos de máquinas, activar las preferencias del usuario y salir de EDIPO. *Setup* es el menú que permite introducir los datos de la sesión (la lista de máquinas, los archivos a transferir, los comandos a ejecutar y los directorios de trabajo en las máquinas). Con las opciones que se encuentran en *Make* se administra el archivo de configuración, se realiza la transferencia de archivos y se ejecutan los comandos remotos. En el menú `Run` están las opciones para obtener información de las máquinas y ejecutar las aplicaciones.

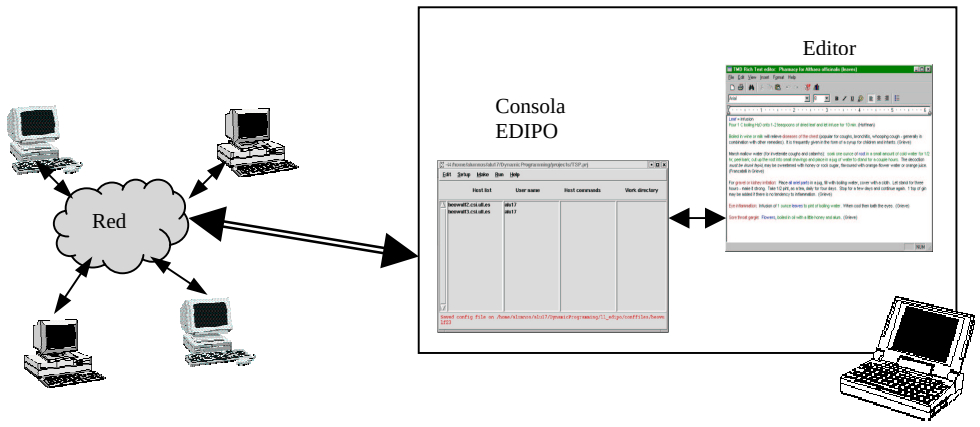


Figura 2. Entorno de desarrollo EDIPO

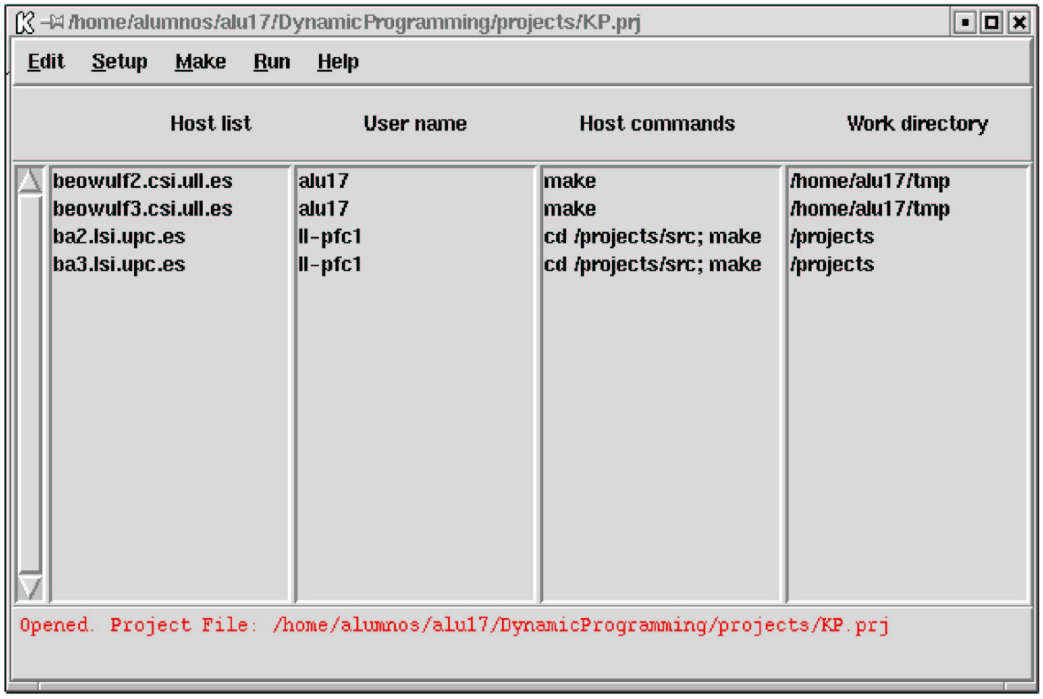


Figura 3. Pantalla principal de EDIPO

La parte principal de la ventana se divide en cuatro columnas: en la primera columna se presentan los nombres de todos los nodos que forman la máquina virtual sobre la que se está trabajando. El ejemplo que se muestra en la figura 3 es un caso en el que la máquina virtual está compuesta por cuatro nodos: las dos primeras máquinas (beowulf2.csi.ull.es y beowulf3.csi.ull.es) se encuentran localizadas geográficamente en Universidad de La Laguna, mientras que las dos últimas (ba2.lsi.upc.es y ba3.lsi.upc.es) se encuentran en la Universidad Politécnica de Cataluña.

En la segunda columna se encuentran los nombres de usuario en cada uno de los nodos. En el ejemplo el alumno tiene un nombre en las máquinas de La Laguna y otro nombre de usuario distinto en las máquinas de Cataluña. En la tercera columna se pueden ver los comandos que se quieren ejecutar en los nodos. Y en la última columna están definidos los directorios de trabajo.

Bajo las columnas se muestra el último comando ejecutado.

Definir una máquina virtual consiste simplemente en añadir cada una de las máquinas a la lista de nodos que aparece en la ventana principal. Esto se realiza mediante la entrada *Add Host* del menú *Setup*.

Otra forma de hacerlo es crear un archivo de máquinas con la opción *New HostFile* del menú *Edit*. Se escribe una máquina por línea, y después se cargan todas las máquinas del archivo a la lista mediante la entrada *Add from HostFile* del menú *Setup*.

Toda la información de la sesión se debe almacenar en un archivo de configuración de la máquina de desarrollo. Este archivo es necesario para realizar la transferencia genérica de archivos y para la compilación remota. Para almacenar el archivo de configuración se utiliza la entrada *Save Config File* del menú *Make*.

Durante la configuración de la máquina es posible incluir la información relativa a los archivos que se van a transferir, los comandos a ejecutar en esa máquina, y el directorio de trabajo y el nombre del usuario para acceder a cada nodo.

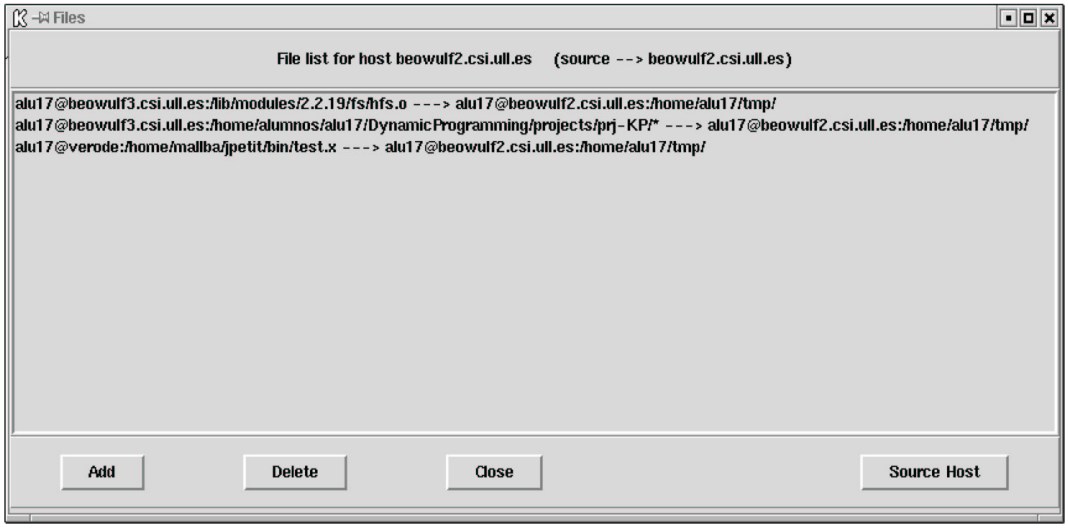


Figura 4. Pantalla de transferencias de archivos entre nodos

En lugar de introducir toda la información cada vez que se inicie una sesión, se puede cargar un archivo de configuración previamente guardado que recupere el estado general de dicha configuración (máquinas, comandos, transferencias, directorios, nombres de usuario, etc.). Para hacerlo hay que pulsar la opción *Load Config File* del menú *Make*.

La entrada *List Config File* del menú *Make* muestra en una nueva ventana el contenido del archivo de configuración. Esto permite detectar errores antes de realizar la compilación remota u otra operación, o simplemente revisar el estado actual de nuestra sesión.

El archivo de configuración sigue la estructura propuesta por PADE. Contiene cuatro tipos de elementos:

- Comentarios: Líneas que empiezan por #.
- Máquina, Nombre de usuario y Directorio de trabajo:
  ```
  + hostname, username, workdir
  ```
- Transferencia de archivos:
  ```
  <filename_sourcehost
  >targethost1
      filename_targethost1
  …
  >targethostN
      filename_targethostN
  <BLANK LINE>
  ```
- Comandos:
  ```
  @remotehost command1; … ;
  commandN;
  ```

3.2. Transferencia de Archivos

La transferencia de archivos entre las distintas máquinas se debe realizar en dos etapas: primero hay que determinar los archivos que se quieren transferir y después se realiza la copia.

Para crear la lista de los archivos a transferir a uno de nodos se elige la máquina en la pantalla principal (Figura 3) y después se selecciona la opción *File Transfer* del menú *Setup*. Este nodo será la máquina destino a la que se quieren enviar los archivos. Se muestra una ventana en la que se pueden ir añadiendo los archivos que se quieren enviar, donde se especifica el nombre del archivo y la máquina donde se encuentra.

La figura 4 muestra la ventana que aparece al elegir esta opción del menú. En el título se puede ver el nodo destino, donde se van a transferir los archivos seleccionados; en este caso de ejemplo se van a transferir tres archivos a la máquina beowulf2.csi.ull.es.

En la parte inferior de la ventana se encuentran los botones que permiten gestionar la transferencia de archivos: se puede añadir un nuevo archivo (botón *Add*) o eliminar una de las transferencias incluidas hasta el momento (botón *Delete*).

El nodo fuente, por defecto, es la máquina de desarrollo, pero también se puede cambiar pulsando sobre el botón *Source Host* que permite elegir un nuevo nodo fuente. En el ejemplo de la figura 4 los archivos que se van a transferir a beowulf2.csi.ull.es tienen dos nodos origen distintos, los dos primeros archivos se copian desde beowulf3.csi.ull.es mientras que el último archivo proviene de verode.

Cuando se ha terminado de insertar todos los archivos que se van a enviar al nodo de destino se pulsa sobre el botón *Close* para cerrar la ventana.

No es necesario almacenar las transferencias sino que al añadirlas ya quedan almacenadas automáticamente.

En la parte central de la ventana se observan las transferencias añadidas. El formato de cada una de las líneas es el siguiente:

```
usuario_nodo_origen@nodo_origen
:/ruta_archivo_origen/nombre_arch
ivo_origen →
usuario_nodo_destino@nodo_destino
:/ruta_archivo_destino/nombre_arc
hivo_destino
```

Utilizando la opción *Save Files in Remote Hosts* del menú *Make* se realizan todas las transferencias de archivos que hayan sido guardadas en el archivo de configuración actual.

Se utiliza el comando genérico de copia remota `rcp` para enviar los archivos. La transferencia se realiza de forma paralela.

3.3. Ejecución de comandos

Los comandos pueden introducirse mientras se añade la máquina si se utiliza la opción *Add Host* del menú *Setup*. Si se utiliza la opción que carga todas las máquinas desde un archivo, se pueden añadir los comandos seleccionando primero una máquina con el ratón y pulsando la entrada *Modify Host* del menú *Setup* que permite modificar los datos de la máquina (nombre, comandos y directorio de trabajo).

Mediante la opción *Remote Make* del menú *Make* se realiza la ejecución de todos los comandos remotos del archivo de configuración. En este caso, cada máquina ejecuta los comandos asociados de forma paralela.

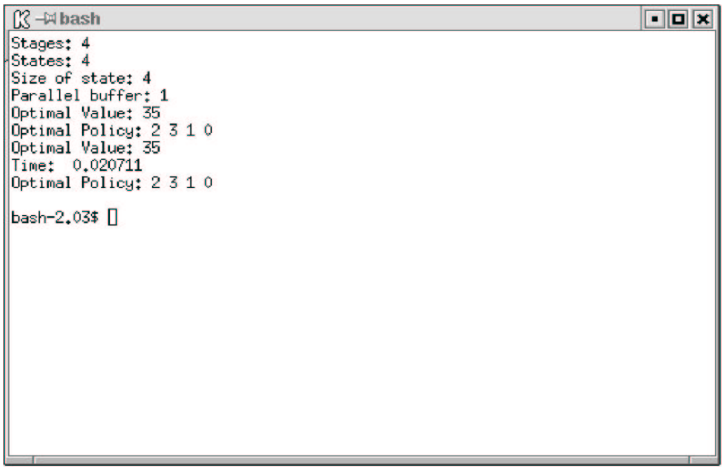


Figura 5. Pantalla de salida de ejecución por terminal

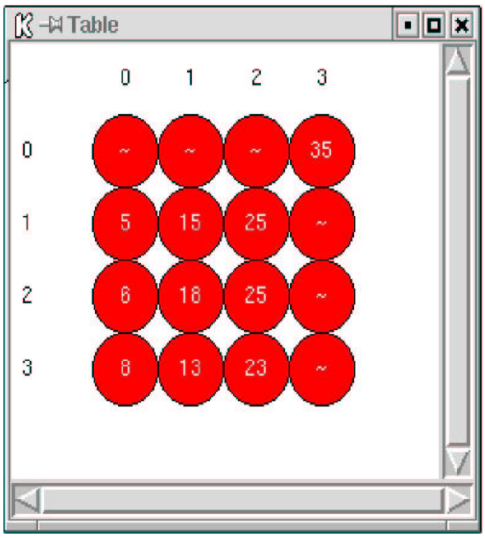


Figura 6. Pantalla de salida de ejecución gráfica

3.4. Compilación y ejecución de Aplicaciones

Tal y como se ha indicado en la subsección 3.3 es posible introducir cualquier comando para una máquina particular, aunque lo más común será usar el comando *make* para compilar los archivos fuente de la máquina remota.

Para la ejecución de aplicaciones EDIPO proporciona varios modos de ejecución: ejecución en modo *batch*, ejecución con salida en terminal, con salida gráfica y con salida en archivo. El usuario sólo debe escoger el modo de ejecución que quiere emplear seleccionando la entrada correspondiente del menú *Run*.

En la figura 5 se puede observar la salida que muestra EDIPO al resolver el problema del Viajante de Comercio (*TSP*) mediante programación dinámica paralela si se escoge la opción de ejecutar el proyecto con salida en un terminal.

En la figura 6 se muestra el resultado de la ejecución del problema cuando se selecciona la opción de salida gráfica. Este tipo de salida depende de la forma de resolución del problema, en este caso el método usado es la programación dinámica.

Para realizar la salida en modo gráfico se establece un protocolo entre la aplicación escrita en cualquier lenguaje de programación tal como `C`, `C++`, `Java`, etc., y el entorno escrito en `Tcl/Tk`.

La aplicación paralela debe escribir en un archivo temporal los valores necesarios para que `Tcl/Tk` represente la salida de forma gráfica.

3.5 Otras Operaciones Disponibles en EDIPO

En cualquier momento se pueden ver las características físicas de los nodos que forman la máquina virtual. Seleccionando la opción *Info Host* del menú *Run* aparece la ventana con todas las características (Figura 7).

También se tiene la posibilidad de gestionar los proyectos que se quieren resolver. En el menú *Edit* se presentan opciones para crear un proyecto o abrir uno existente (opciones *New Proyect* y *Open Proyect*). Al crear un nuevo proyecto crea un fichero de configuración que contiene los nombres y las rutas de cada uno de los archivos fuente que componen ese proyecto.

4. Una Instalación Sencilla

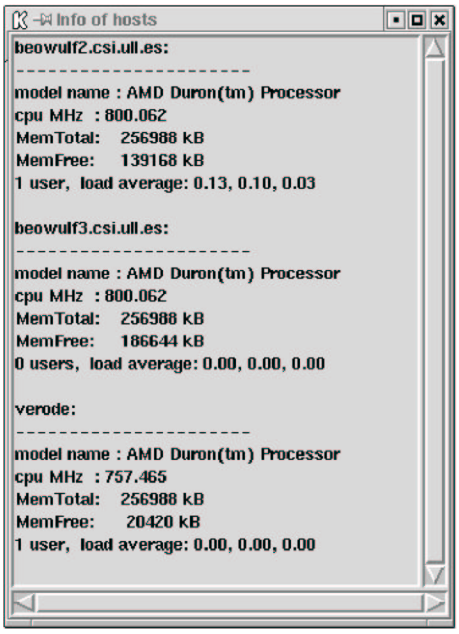
Para su instalación basta con que el alumno descargue un grupo de *scripts* y que defina y modifique algunas variables de entorno. Como se comentó en la sección 1 tanto `Tcl/Tk` como los mecanismos para la comunicación remota, `rcp` y `rsh`, vienen instalados por defecto en la mayoría de las versiones de `Unix/Linux`.

También se debe tener instalada la herramienta elegida para la ejecución de los proyectos (`MPI`, `PVM`, `Java RMI`, `Corba`).

Las variables de entorno que debe definir el alumno son las siguientes:

- `LLEDIPO` - Directorio en el que se instaló EDIPO.
- `TCL_LIBRARY` - Directorio de la librería TCL.
- `TK_LIBRARY` - Directorio de la librería TK.
- `EDITOR` - Editor por defecto para editar los ficheros fuente (opcional).

Debe modificarse la variable de entorno *PATH* para añadir al path de ejecutables los directorios: \$LLEDIPO y \$LLEDIPO/bin



Por último, también es necesario generar un archivo *.rhost* en el directorio *home* del usuario donde se especifican las máquinas en las que se tienen permisos para escribir en su sistema de archivos.

Una vez realizada la instalación, solamente hay que escribir EDIPO en la línea de comandos. Primero se leen las variables de entorno y se buscan en la variable de entorno *PATH* del usuario los programas que se deben ejecutar.

Si están disponibles todos los programas que se necesitan se muestra la ventana principal del entorno (figura 3).

5. Conclusiones y trabajos futuros

Se ha presentado una herramienta genérica que facilita el desarrollo y ejecución de una aplicación paralela sobre una máquina virtual heterogénea, lo que permite a los alumnos de las asignaturas relacionadas con la Programación Distribuida o la Programación en Paralelo realizar sus prácticas sobre redes geográficamente dispersas.

Una de las mejoras que puede llevarse a cabo en EDIPO está relacionada con la seguridad en las comunicaciones, pueden utilizarse los mecanismos de comunicación segura (como `ssh` y `scp`) en vez de utilizar lugar de los comandos `rsh` y `rcp`.

La metodología utilizada para el desarrollo de EDIPO (*scripts*, mecanismos de comunicación remota, etc.) permite adaptar y extender este entorno con nuevas y distintas funcionalidades y podría ser ampliable para que soporte otras herramientas como `Java RMI`.

Referencias

[1] R.Baraglia, D. Laforenza. *WAMM, A Metacomputer Graphical Interface.* Parallel Processing Group. CNUCE Institute, Pisa. cnuce-arch.cnuce.cnr.it/wamm/

[2] Condor: www.cs.wisc.edu/condor/

[3] J.E. Devaney, R. Lipman, M. Lo, W.F.Mitchell, M. Edwards, C.W. Clack. The *Parallel Applications Development Environment (PADE). User's Manual.* National Institute of Standards and Technology. www.itl.nist.gov/div895/pade/

[4] Globus: www.globus.org/

[5] *Linux in a Nutshell*, Second Edition. Ellen Siever and the Staff of O'Reilly & Associates, Inc. 1999.

[6] *MPI: Message Passing Interface.* www-unix.mcs.anl.gov/mpi/

[7] C.A. de Pablos Prieto. *Programación dinámica Paralela: Esqueletos y Herramientas de Desarrollo. Proyecto Final de Carrera.* Centro Superior de Informática. Dpto. de Estadística, Investigación Operativa y Computación. Universidad de La Laguna. Enero 2002.

[8] J. Peek, T. O'reilly, M. Loukides. *Unix Power Tools*. O'Reilly & Associates, Inc. 1994.

[9] *PVM: Parallel Virtual Machine.* www.csm.ornl.gov/pvm/pvm_home.html

[10] B. Welch. Practical Programming Tcl and Tk. Prentice Hall. 1999

Sistemas operativos

La asignatura Sistemas Operativos I en el 2mil2

José Antonio Gómez Hernández
Dpto. de Lenguajes y Sistemas Informáticos
Universidad de Granada
18071 Granada
e-mail: jagomez@ugr.es

Resumen

El artículo expone algunas ideas sobre cuales deben ser los contenidos y enfoque adecuados para la teoría de una asignatura de introducción a los sistemas operativos. Esta concepción de la asignatura pretende satisfacer las necesidades de formación en la materia que, en mi opinión, tienen nuestros alumnos, se basa principalmente en la experiencia acumulada impartiendo diferentes asignaturas sobre la disciplina, y toma en consideración los avances producidos en Sistemas Operativos en los últimos años.

1 Introducción

Existe una evolución continua en la forma de enseñar sistemas operativos dado el reto que esto supone por la cantidad de material que se ha de cubrir y por el equilibrio que ha de existir con la práctica. Si bien las unidades que se han de cursar están suficientemente claras en las propuestas curriculares como [1], mi inquietud surgió hace ya tiempo a la hora de abordar cuales deben ser los contenidos concretos que han de cursar los alumnos de la asignatura y el enfoque que se le ha de dar. Esto me llevó a plantear cual debe ser la formación global que debe recibir un ingeniero en informática.

Diferentes estudios técnicos que se vienen publicando sobre empleo revelan que gran parte de las profesiones del futuro próximo serán de nueva creación. En este sentido, parece razonable que la Universidad debe estar preparada para una avalancha de titulaciones de nueva la creación o bien la modificación de las ya existentes. Aunque los profesionales del mañana los formemos hoy, no podemos enseñar cómo será la ciencia y tecnología venidera. Esto hace que, como en otras parcelas, en el campo de las tecnologías de la información debemos formar ingenieros capaces de operar más allá de su actual práctica, que sean capaces de empujar las fronteras de su educación actual. En la parcela que nos ocupa, no podemos dejar de prepararlos para abordar los avances conceptuales y tecnológicos que se están produciendo en la disciplina [11].

Partiendo de esa idea, he ido depurando y dando forma a una propuesta de curso en la que, cubriendo los conceptos básicos de sistemas operativos, se intenta ir dejando un poso que integre dichos conocimientos con los impartidos en el resto de las asignaturas, al objeto de dejar claros una serie de principios válidos a todas ellas. Este enfoque unificador es el de diseño de sistemas.

De las numerosas propuestas realizadas, que podemos ver en artículos recientes en SIGCSE[1] y OSR[2], tenemos desde enfoques *top-down* [7] hasta *bottom-up* [4]. Desde mi punto de vista hay que evitar aquellas propuestas que:

a. No sean realistas al no contemplar que el funcionamiento de un sistema operativo es más complejo de lo que se deriva de las descripciones de sus partes.
b. Mantienen una visión historicista, como ocurre en algunos de los libros de texto de sistemas operativos. Muchos de ellos son el resultado de un proceso acumulativo de conocimientos desde su primera versión. Esto no sólo tiene el problema inmediato de que su contenido es tan amplio que no es factible cubrirlo en los créditos asignados a una asignatura, sino que, además, dificultan el discernimiento entre lo que es obsoleto y los avances recientes.

[1] Grupo de interés de la ACM en Computer Science Education.
[2] Operating System Review de la ACM.

Citemos algunos ejemplos. Sobre el primer punto, la mayoría de los libros de texto descomponen el sistema en módulos pero luego, no lo recomponen, no se analizan los caminos de código cuando se realiza una llamada al sistema o cuando se produce una interrupción. Sorprende ver cómo se estudian los manejadores de interrupciones, pero no se sabe muy bien cómo llega la información de un dispositivo hasta la aplicación que la solicitó, o cómo se desbloquea al proceso llamador. Sobre el segundo punto, se pueden dedicar algunas páginas a explicar un sistema operativo por lotes y, tan siquiera se describen las características de un sistema operativo empotrado.

Opino que el uso de código real (enfoque bottom-up) no es adecuado en un nivel introductorio. Como he podido constatar en cursos de este tipo, el conocimiento previo del que disponen los alumnos de segundo curso no es aún el adecuado, es muy restrictivo pues no permite el análisis de alternativas de diseño, y es difícil de encajar en una asignatura cuatrimestral.

Un enfoque estrictamente top-down presenta el problema de no ser realista en cuanto al reconocimiento de la complejidad de un sistema operativo real. Por ello, mi propuesta para el curso pretende llegar a un punto de equilibrio entre ambos enfoques, un enfoque de diseño. Según la cual, partiendo de un enfoque top-down llegar a analizar los elementos suficientes de implementación para llegar a comprender el funcionamiento real de muchos sistemas. Un tipo similar de propuesta puede verse en [8].

2 Las asignaturas de sistemas operativos

La ETS de Ingeniería Informática de la Universidad de Granada imparte las titulaciones de Ingeniero en Informática, Ingeniero Técnico en Informática de Sistemas e Ingeniero Técnico en Informática de Gestión. En las tres titulaciones se imparten las asignaturas de Sistemas Operativos I, troncal y con una carga docente de 4.5T + 1.5P, y Sistemas Operativos II, obligatoria y con la misma carga docente que la anterior. Ambas asignaturas se imparten en segundo curso, en el primer y segundo cuatrimestre, respectivamente. Además, la titulación de Ingeniero en Informática contiene la asignatura optativa Diseño de Sistemas Operativos con una carga docente de 3T+3P.

De otra parte, existen una serie de asignaturas que podemos considerar como prerrequisitos de las antes citadas como son Metodología de la Programación I y II, e Introducción a los Computadores, que se imparten en primer curso. En segundo curso, se imparten en paralelo las asignaturas de Estructura de Computadores I y II. Estas asignaturas permiten eliminar algunos contenidos de estructura y funcionamiento del computador de nuestra asignatura e ilustrar parte de la interacción *hard-soft*.

3 Propuesta de contenidos

El programa propuesto para la asignatura recoge los descriptores impuestos por la troncalidad e incluye los temas básicos de una asignatura introductoria: introducción a los sistemas operativos, gestión de procesos, sincronización y comunicación, organización de memoria, gestión de memoria virtual, gestión de entradas/salidas y sistema de archivos.

El temario de Sistemas Operativos II incluye los siguientes puntos: implementación de los sistemas de archivos, planificación de recursos, protección y seguridad, una introducción a los sistemas operativos distribuidos, e implementación del núcleo de un sistema operativo centrado principalmente en el estudio Unix, así como una visión general de Windows 2000.

El temario de Sistemas Operativos I parece bastante clásico, si bien se aparta de esta línea en la forma de enlazar los diferentes temas y en los contenidos asignados a cada unidad. De acuerdo con [10], la experiencia muestra cómo los alumnos acceden a la asignatura con entusiasmo entendiendo que los sistemas operativos son un elemento clave en un sistema de computador, pero, desafortunadamente, finalizan a menudo el curso un poco desencantados, en especial cuando se sigue un enfoque estrictamente *top-down*, en el cual deben memorizar un importante número de primitivas de sincronización, políticas de planificación de procesos, y algoritmos de sustitución de páginas. Evidentemente, estos temas son importantes de por sí, pero no son más que un componente en un gran cuadro. Por ello, es importante entrar en detalles de implementación para hacerles ver que no hay nada mágico en el funcionamiento del sistema. En este punto hay que llegar a un equilibrio entre la casi imposibilidad de explicar código real y explicar suficientes detalles para que el alumno imagine el código a nivel de funciones. Por ejemplo, centrándome en una de las

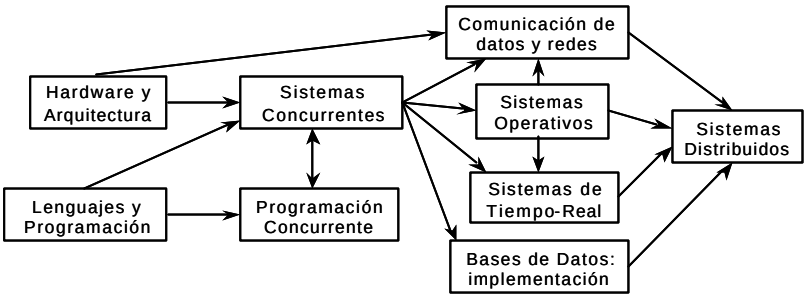


Figura 1.- Visión general de los sistemas concurrentes, modificada de [3]

unidades pilares de la asignatura, la implementación de procesos y concurrencia: explicar cómo se lleva a cabo el cambio de contexto, la operación de bloqueo de procesos, la invocación de una llamada al sistema, la gestión de interrupciones y cómo el diseño realizado en la mayoría de los sistemas operativos esconde éstas a los procesos.

A continuación, comentaré los contenidos más relevantes de cada unidad.

3.1 Introducción a los sistemas operativos

La experiencia ha ido mostrándome cómo los estudiantes tienden a ver en los primeros cursos de la carrera los sistemas operativos como algo aislado. Es interesante y enriquecedor para ellos tener una visión más amplia, por ello es útil en el primer tema dar una visión general de los sistemas concurrentes (Figura 1) y cómo se materializan éstos en las asignaturas de su plan de estudios. Esto les permite crear relaciones entre asignaturas que de otra forma les parecerían compartimentos estancos.

Otro elemento básico a tratar es la creación de abstracciones, ver [5], ya que de hecho el kernel es la implementación de la interfaz definida por las llamadas al sistema. Razonar sobre la interfaz es importante desde el punto de vista del diseño: conocer por qué las funciones de la interfaz tienen la estructura que tienen, por ejemplo, `fork` o `read()` de Unix frente a `CreateProcess` o `ReadFile()` de Windows, o cómo la interfaz define la complejidad de la máquina abstracta que tratamos de implementar. Es bueno ilustrar varias interfaces de programación, por ejemplo, la POSIX de Unix y la Win32 de Windows que son actuales y siguen dos filosofías distintas, como recoge la elogiable propuesta realizada en [6].

El estudio de las llamadas al sistema nos lleva a vislumbrar los diferentes tipos de servicios suministrados por el sistema operativo, lo que nos da pie a estructurar el sistema en grandes módulos, cada uno de los cuales suministra un tipo de servicio. Aquí es importante hacer ver al alumno de manera intuitiva que la dependencia entre módulos es compleja. Por ejemplo, para realizar una actividad necesitamos crear un proceso, el cual necesita memoria y su programa ejecutable está en un archivo al igual que los datos que maneja. Esto da pie para introducirlos en el estudio de arquitecturas de sistemas operativos guiados por el intento de dar respuesta a la cuestión de cómo construir un sistema que sea fiable, extensible y eficiente. Este apartado esta centrado en las arquitecturas monolítica y microkernel, si bien se ven brevemente la estructura de capas y la de Máquina Virtual, por aparecer en muchos sistemas actuales. Para analizar las arquitecturas anteriores es necesario introducir el concepto de espacio de direcciones, recordar el paradigma de llamada a procedimiento ampliándolo, a procedimiento protegido -llamada al sistema- y avanzar el de paso de mensajes.

Para finalizar el tema, se repasan los principales tipos o clases de sistemas operativos, evitando al máximo explicar sistemas en desuso. Se estudian los sistemas multiprogramados, por contener los elementos básicos de un sistema operativo actual, posponiendo para el tema de sincronización una justificación detallada de la multiprogramación. Por supuesto se define y caracterizan los sistemas de tiempo compartido, los sistemas paralelos y distribuido, los de tiempo-real y los empotrados.

3.2 Procesos y hebras

En el tema de procesos, uno de los aspectos más relevantes es mostrar a los estudiantes que un proceso es una abstracción introducida para aislar las diferentes actividades del sistema y poder razonar sobre ellas de forma separada. Sorprende ver como este concepto está tan arraigado en ellos que les parece que se produce de forma natural en la propia estructura del computador. Desde esta idea se abordan las estructuras de datos necesarias para construir esta abstracción. Se explica con detalle esa misteriosa operación de cambio de contexto. En la parte de planificación se hace hincapié en cómo se activa el bucle de despacho, y respecto a los algoritmos de planificación, interesa centrarse en los que han sobrevivido al paso de tiempo. Se introduce el concepto latencia de despacho y se ve por qué es necesario reducirla en sistemas operativos de tiempo-real. Se explica el fenómeno de inversión de prioridad y se indica como se solventa a través de un protocolo de herencia de prioridad.

Por otra parte, se muestran las limitaciones del modelo proceso, creado hace varias décadas, cuando trabajamos con multiprocesadores o con aplicaciones inherentemente paralelas. Esto nos permite introducir el concepto de hebra (o hilo) y sus tipos, y cómo el modelo hebra esta diseñado para ser compatible en alto grado con el modelo proceso ya que ambos conviven en la mayoría de los sistemas. Este es uno de los puntos en el que el descenso a nivel de implementación facilita la comprensión del concepto hebra y los diferentes tipos. Aquí es importante poner algunos ejemplos reales para ilustrar las sutilidades de la nueva abstracción. Abundaré sobre esta cuestión en el apartado de metodología.

Hay ejercicios que creo importantes para conocer si han comprendido los conceptos básico del tema. Por ejemplo, ¿puede un SO atender las interrupciones si utiliza una política de planificación no apropiativa?, ¿debe un proceso cambiar él mismo su estado de “ejecutándose” a “bloqueado”? Desde el punto de vista de diseño, casi más importante que el estudio de un determinado algoritmo de planificación es hacerles ver, por ejemplo, cuales son los mecanismos utilizados para disparar la acción del planificador.

3.3 Sincronización y comunicación

En este tema, el planteamiento inicial es introducirles en la necesidad de la programación concurrente para resolver determinado tipo de aplicaciones. Para ello, les muestro un ejemplo sobre el dibujo de un conjunto de Mandelbrot (proceso acotado por computo) en una ventana Windows y pueden observar como, mientras dibujamos, es imposible acceder al menú de opciones de la ventana. A continuación se utiliza el mecanismo de entradas/salidas asíncronas para dibujar con una hebra y atender la ventana con otra, resultando ahora posible acceder al menú mientras se dibuja. El resto del tema se dedica a ver los problemas clásicos de sincronización utilizando semáforos. Esta primitiva, junto con los cerrojos de espera ocupada (*spin lock*), son las únicas primitivas de sincronización que les muestro, ya que el estudio de gran variedad de primitivas genera más confusión que beneficios. En cualquier caso, les hago notar que en las asignaturas de Programación Concurrente, Diseño de Bases de Datos, etc. verán un amplio abanico de primitivas.

3.4 Gestión de memoria

Respecto a la gestión de memoria, los puntos más importantes a mi entender son la necesidad de diferenciar entre espacio lógico de direcciones y espacio físico, y cómo se hace corresponder uno en otro a través de lo que yo denomino la Caja de Traducción (la MMU). En los últimos años, he ido reduciendo el estudio de esquemas simples de asignación de memoria al estudio de particiones variables, con la sola idea de introducir el problema de la fragmentación. Esto permite centrarse más en los esquemas de gestión de memoria utilizados para evitar este problema: segmentación y paginación. Se sigue viendo el intercambio porque aún no ha perdido toda su utilidad . En paginación es importante resaltar que su buen funcionamiento depende de la localidad de los programas, que influye entre otras cosas en el buen funcionamiento del TLB (Búfer de reconocimiento de traducciones). De otra parte, es necesario acudir a la programación en ensamblador, que se estudia en otra asignatura,

para ver las diferencias entre segmentación y paginación, ya que inicialmente les resultan mecanismos muy parecidos.

En gestión de memoria virtual, aparte de los elementos tradicionalmente vistos, hay que hacerles ver lo que no dicen claramente muchos libros de texto: que al sobrecargar el bit de validez de significado para tratar la falta de página, la tabla de páginas no describe el espacio completo de direcciones de un proceso, y que una falta de página no indica necesariamente que una página no está en memoria, sino que la traducción que tenemos no es correcta aún cuando la página puede estar en memoria. Creo necesario explicar el mecanismo de copia-al-escribir (*copy-on-write*) para explicar con posterioridad archivos proyectados en memoria.

Respecto a los algoritmos de sustitución de páginas, cada vez tiendo más a reducir su número, explicando básicamente el algoritmo del reloj y el de Frecuencia de falta de página, que son los más utilizados actualmente. Es importante explicar el Modelo del Conjunto Activo, en especial por su implicaciones con la planificación de proceso a través de Principio de Conjunto Activo.

3.5 Gestión de entradas/salidas

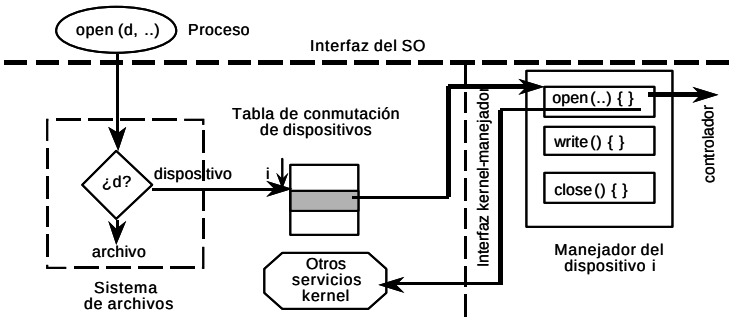
Dado que en asignaturas como Introducción a los Computadores y Estructura de Computadores dedican parte del programa a ver la arquitectura hardware de los dispositivos, este tema tiene como puntos fuertes los siguientes: (a) analizar la arquitectura software del subsistema de entradas/salidas, (b) ver cómo se designan dispositivos haciendo hincapié en su integración en el espacio de nombres de archivos, y (c) estudiar cómo se implementa el concepto de entradas/salidas independientes del dispositivo (Figura 1). Respecto al punto (a), es de especial relevancia el estudio del camino de código para una operación de entrada o salida, que permite visualizar un ejemplo real del problema del productor-consumidor, y nos lleva a plantear la necesidad de diseñar un manejador de dispositivos dividido en dos mitades. La mitad superior se ejecuta en el contexto del proceso llamador, pudiéndose bloquear, y la mitad inferior, que no se puede bloquear, ha de ejecutarse en un contexto diferente.

3.6 Sistemas de archivos

Por último, dado que en Sistemas Operativos II existe un tema dedicado a la implementación de sistemas de archivos, en la asignatura nos dedicamos más a ver las abstracciones suministradas al usuario, como son archivos, directorios, descriptores de archivos, enlaces duros y simbólicos. También les introduzco los temas de las posibles semánticas de consistencia, designación de archivos, y el concepto de archivo proyectado en memoria, por su extendido uso hoy en día y como ejemplo de interrelación entre los subsistemas de memoria y de archivos.

Una visión general de la interfaz de sistema de archivos virtuales les permite comprender cómo en los sistemas actuales es posible manejar simultáneamente diferentes tipos de sistemas de archivos. Así, podemos mostrar cómo este mecanismo permite integrar abstracciones diferentes con la apariencia de archivos, me refiero, por ejemplo, al sistema de archivos de dispositivo, al sistema de archivos /proc, etc.

Figura 2.- Implementación de la independencia del dispositivo.



3.7 Otros puntos tratados

Si bien el tema de eficiencia recogido en el currículum de la ACM/IEEE no se trata expresamente en la asignatura, creo que es conveniente hacer hincapié a los estudiantes en la importancia de un buen rendimiento en el SO. En este sentido, trato de ir dando siempre valores de sistemas reales para el coste de ejecución de funciones como cambio de contexto, cambio de modo, coste de creación de un proceso y de una hebra, tiempo de acceso efectivo, etc. ya que en muchos casos estos valores son los que justifican que se elija entre una de varias alternativas en el diseño e implementación de una función del SO.

También, dejo a mis alumnos algunos temas de libre estudio tratando de enlazar con asignaturas que se cursan con posterioridad como es, por ejemplo, el punto de contacto entre la generación de código del compilador y la gestión de memoria del sistema operativo. Para ello, he elaborado dos temas denominados *Asignación de memoria y enlazadores*, y *Designación y memoria virtual*.

Aparte del temario, he comprobado eficiente y motivador aplicar la idea de T. Anderson en [2] de impartir de una serie de "sermones" (charlas de 15 min.) que, no siendo estrictamente claves en la asignatura, provocan la reflexión de los alumnos sobre determinados temas como son: *La información es propiedad*, *La sencillez*, *Mantenerse abierto*, y *Construcción de sistemas eficientes*.

En otro ámbito diferente, la no existencia de un vocabulario común me lleva a explicar a mis alumnos los términos en inglés y las traducciones más comunes de los mismos así como las que yo empleo, ya que los diferentes libros de texto utilizan en algunos casos traducciones diferentes para los mismos términos. Otras veces, las traducciones son de uso infrecuente o erróneo en España, ya que están pensadas para el mercado latinoamericano, etc.

4 Metodología

Como se indica en [9,12], la mera transferencia de conocimiento suministrada por la clase magistral debe dejar paso a un proceso de aprendizaje individual basado en la motivación. Por ejemplo, emplear parte de una clase en enseñar determinado algoritmo de planificación, que es fácil de aprender en un libro de texto, es menos productivo y motivador que emplear este tiempo en hacer ver al estudiante cómo se activa en algoritmo de planificación con la ocurrencia de una interrupción, por citar un ejemplo. Por ello, procuro dedicar más tiempo a cubrir aspectos fundamentales de diseño, que no quedan claros en la bibliografía, e inducir al estudiante al razonamiento e interconexión de conceptos, que a explicar detalles o ver variantes de un tema que pueden leer en la bibliografía, y que, no aportando gran contenido, requieren un tiempo precioso. Por ejemplo, muchos libros de texto emplean gran número de páginas a explicar las diferentes variantes y aproximación del algoritmo LRU para sustitución de página aunque en la vida real se aplican pocas de ellas, y sin embargo, obvian la necesidad de introducir un proceso/hebra de fondo para realizar dicha sustitución.

La estructura de cada una de las unidades sigue siempre un mismo esquema: (1) se plantea un problema, (2) se estudian diferentes alternativas de diseño propuestas, y (3) se analizan las ventajas y desventajas de las soluciones vistas dejando clara la métrica de evaluación, por ejemplo, rendimiento de las soluciones, consumo de recursos, flexibilidad, etc.

La naturaleza del material cubierto en la asignatura obliga a abordar un enfoque práctico en el que es aconsejable utilizar técnicas de visualización para presentar el material en un formato intuitivo, ver [13]. Estas técnicas exigen un gran esfuerzo en adaptar el material de curso a un formato gráfico, si bien se adaptan bien a diferentes metodologías como enseñanza a distancia mediante Web. Por ejemplo, la Figura 2 ilustra los pasos seguidos para acceder a un dispositivo utilizando la técnica de independencia de los dispositivos de E/S.

En un línea similar, un aspecto importante de las nuevas tecnologías, que he constatado motivador e ilustrativo, es el uso del ordenador en clase. Este nos puede servir para ilustrar los conceptos explicados. Por ejemplo, la programación "real" de ejercicios, como pueden ser los de sincronización con semáforos, y su ejecución en clase permite entre otras cosas: ver cómo realmente se produce el interbloqueo en los filósofos comensales, Figuras 3 y 4.

```
#define NFILOSOFOS 5
HANDLE hMutexTenedor[NFILOSOFOS];
...
void main() {
HANDLE hVectorHebras[NFILOSOFOS];
DWORD IDHebra;
int i;
for (i=0; i<NFILOSOFOS; i++)
hMutexTenedor[i]=CreateMutex(NULL, FALSE,NULL);
for (i=0; i<NFILOSOFOS; i++)
  hVectorHebras[i] = CreateThread (NULL, 0,
(LPTHREAD_START_ROUTINE)Filosofo,(LPVOID)i,0, (LPDWORD)&IDHebra);
WaitForMultipleObjects(NFILOSOFOS,hVectorHebras,TRUE,INFINITE);
}

VOID Filosofo(LPVOID id) {
int izqdo = (int) id;
int dercho = (izqdo + 1) % NFILOSOFOS;
while(1){
  WaitForSingleObject(hMutexTenedor[izqdo],INFINITE);
  printf("Filosofo %d: coge tenedor izqdo.\n",(int) id);
  WaitForSingleObject(hMutexTenedor[dercho],INFINITE);
  printf("Filosofo %d: coge tenedor dcho. y come\n",(int) id);
  ReleaseMutex(hMutexTenedor[izquierdo]);
  ReleaseMutex(hMutexTenedor[derecho]);
  printf("Filosofo %d: libera tenedores y piensa\n",(int) id);
 }
}
```

Figura 3.- Solución simple del problemas de los filósofos-comensales.

Otra técnica que se muestra de gran utilidad para comprender los conceptos y mecanismos vistos en clase, es la de formar grupos de trabajo con el objetivo de dar respuesta a una pregunta formulada al hilo de lo explicado. Por ejemplo, una de las diferentes preguntas formuladas en el presente curso estaba encaminada a que los alumnos abordasen el dilema de cómo es posible apropiar un proceso que se ejecuta en modo usuario teniendo en cuenta que la operación cambio de contexto se realiza en modo kernel y este modo es en muchos sistemas no apropiativo. En estas cuestiones, no importa tanto el que alcancen una respuesta correcta al 100% como que razonen sobre el funcionamiento del sistema.

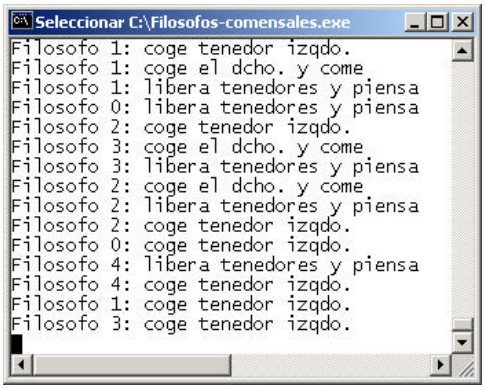


Figura 4.- Interbloqueo en los filósofos-comensales.

5 Evaluación

Si bien no he evaluado formalmente el curso, he podido observar en los alumnos un creciente interés por la asignatura. Muchos de los que han visitado la página Web, me han indicado la utilidad de sus contenidos.

Parte de la motivación encontrada en clase obedece a la actualidad de los ejemplos vistos que permite relacionar a los estudiantes los conceptos vistos en clase con los sistemas operativos usados diariamente, ya sea en su PC, PAD, teléfono móvil, electrodomésticos, etc.

6 Conclusiones

Se han mostrado a grandes rasgos cuales deben ser, a mi entender, los contenidos de una asignatura de introducción a los sistemas operativos. Considero que estos contenidos son necesarios para obtener una visión general del diseño de sistemas que trascienda a los propios contenidos de un curso de esta disciplina.

La propuesta realizada tiene aún algunos inconvenientes y cuestiones sin resolver en su totalidad. A la vista del resurgimiento de la investigación en este campo, que por algunos años parecía aletargada, pretende ser un estímulo para reflexionar sobre la docencia de sistemas operativos en este nuevo milenio

En http://lsi.ugr.es/~jagomez/sisopi.html se puede ver parte del material docente usado para la asignatura Sistemas Operativos I y generado según el enfoque y contenidos propuestos. No obstante, debe tenerse presente que aún queda trabajo por hacer para actualizar algunos de los temas que aparecen en la página con respecto a los principios propuestos en este trabajo.

He omitido intencionadamente a lo largo de la discusión el tratamiento dado a las prácticas de la asignatura debido a que están en fase de remodelación. El objetivo de las mismas será profundizar en el uso de Linux tanto a nivel de usuario, utilizando el shell y ordenes avanzadas, como de programación incluyendo una parte dedicada a la programación con hebras para la mejor comprensión de los conceptos vistos en el tema de sincronización. En este último aspecto, creo que son buenos los enfoques dado en ciertos trabajos aparecidos en el SIGCSE que utilizan la interfaz de llamadas para reforzar los conceptos vistos en teoría.

Referencias

1. ACM/IEEE, *Computing Curricula 2001 Computer Science*, Final Report, December 15, 2001,
2. T. Anderson, http://http.cs.berkeley.edu/~tea.
3. J. Bacon, *Concurrent Systems. Operating Systems, Database, and Distributed Systems: An Integrated Approach*, 2ª Ed. Addison Wesley, 1998.
4. D. P. Bovet y Marco Cesati, “A Real Bottom-Up Operating Systems Course”, *Operating Systems Review, 35(1),* pgs. 48-60, ACM Press, Jan. 2001.
5. P. Bucci, T.J. Long, y B. W. Weide, “Do We Really Teach Abstraction?, *Proc. of the 32nd SIGCSE Technical Symposium on Computer Science Education*, pgs. 26-30, 2001.
6. J. Carretero Pérez, y otros, *Sistemas operativos. Una visión aplicada*. McGraw-Hill, 2001.
7. R. A. Creak y R. Sheehan, “A Top-Down Operating Systems Course”, *Operating Systems Review, 34(3),* pgs. 69-80, ACM Press, July 2000.
8. C. Crowley, *Operating Systems. A Design-Oriented Approach*, Irwin, 1997.
9. M. García-Valero y J. J. Navarro, “Niveles de competencia de los objetivos formativos en las Ingenierías, *VII Jornadas de enseñanza universitaria de la informática*, pgs. 149-154, Palma de Mallorca, 16-18 de 2001.
10. M. A. Holliday, “System Calls and Interruptor Vectors in an Operating Systems Course”, *Proceedings of the 28th SIGCSE Technical Symposium on Computer Science Education*, pgs. 53-57, 1997.
11. D. Milojicic, “Operating systems - now and in the future”. *IEEE Concurrency*, January-March 1999, pgs. 12-21.
12. M. Rebollo, “Aprendizaje activo en el aula”, *VII Jornadas de enseñanza universitaria de la informática*, pgs. 137-142, 2001.
13. V. Sparks, y Shan Suthaharan, “Operating Systems: Visualization Technique for Teaching and Learning”, *Proceedings of the IEEE*, Southeastcon, 2000, pgs. 387-390.

Telemática

Experiencias prácticas sobre el análisis en frecuencia de señales en la asignatura Sistemas de Transmisión de Datos

José Oliver, Alberto Bonastre, José Luis Poza
Dpto. de Informática de Sistemas y Computadores
Universidad Politécnica de Valencia
Camino de Vera 17, 46017 Valencia
e-mail: {joliver, bonastre, jopolu}@disca.upv.es

Resumen

La descomposición de señales mediante el análisis de Fourier es el método clásico más utilizado para estudiar el comportamiento en frecuencia de una señal. Dentro del marco de una asignatura sobre transmisión de datos, el análisis de Fourier es fundamental a la hora de caracterizar el comportamiento de una señal y cómo el medio actúa sobre ella. Sin embargo, debido a la base matemática necesaria, los alumnos encuentran especiales dificultades a la hora de entender y manejar esta herramienta matemática. En este artículo se presentan una serie de experiencias prácticas que se han desarrollado con el objetivo de ayudar al alumno a comprender la teoría que involucra el análisis de Fourier dentro del marco de los sistemas de transmisión de datos.

1. Introducción

La representación de señales de comunicaciones en el dominio de la frecuencia resulta fundamental para la comprensión de los fenómenos asociados a la transmisión de datos [1]. La mayoría de estos fenómenos no se entienden excepto desde este punto de vista, puesto que la respuesta del medio de transmisión, es decir, cómo se verá afectada la señal al ser transmitida por el mismo, depende de la frecuencia de la señal y sus armónicos.

Sin embargo, la teoría matemática correspondiente a este estudio resulta muy poco intuitiva, representando, según la experiencia acumulada, uno de los contenidos menos atractivos de cualquier temario sobre transmisión de datos o tratamiento de señal.

Con el fin de facilitar la comprensión y demostrar la utilidad de las diversas técnicas de representación en frecuencia se ha desarrollado dentro de la asignatura de Sistemas de Transmisión de Datos una batería de experiencias prácticas basadas en dicha representación.

La asignatura de Sistemas de Transmisión de Datos (STD) [2] se centra en los primeros niveles del modelo ISO/OSI (nivel físico y enlace de datos), 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 6 créditos) e Ingeniero Técnico en Informática de Sistemas y de Gestión (3º curso, con 4.5 créditos).

Este artículo presenta las experiencias prácticas que hemos planteado a los alumnos y valora su aplicación a ambas asignaturas de STD, presentando las herramientas diseñadas a tal efecto y los ejercicios planteados.

2. Experiencias prácticas sobre el análisis de señales mediante Fourier

La descomposición de señales mediante el análisis de Fourier, aunque no el único, sí es el método clásico más utilizado para estudiar el comportamiento en frecuencia de una señal. Su capacidad para descomponer una señal en armónicos y la disponibilidad de algoritmos rápidos para su procesamiento hacen de ésta una herramienta muy utilizada en diversas ramas de la Ingeniería.

Dentro del marco de una asignatura sobre transmisión de datos, el análisis de Fourier es

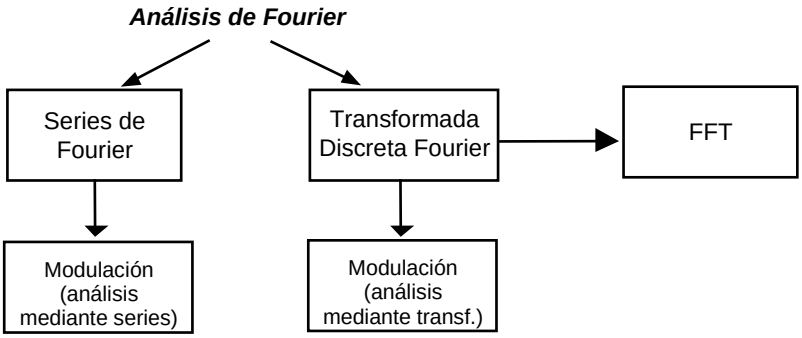


Figura 1. Conjunto de prácticas relacionadas con el análisis de Fourier

fundamental a la hora de caracterizar el comportamiento de una señal y cómo el medio actúa sobre ella. Sabemos que existen fenómenos físicos, como la atenuación y el retardo de grupo, que no afectan de la misma manera a todos los componentes en frecuencia o armónicos de la señal transmitida. A su vez, conceptos como modulación de señales y multiplexación en frecuencia sólo se entienden teniendo una visión espectral de la señal a transmitir, y por tanto resulta fundamental su estudio y mediante una herramienta como la introducida por Joseph Fourier en el siglo XIX.

Con el objetivo de que el alumno entienda y adquiera las habilidades necesarias para utilizar las distintas versiones de Fourier, y que a su vez con ellas sea capaz de comprender mejor aspectos importantes del proceso físico de transmisión de datos, hemos desarrollado un total de cinco experiencias prácticas dentro de nuestra asignatura.

En las tres primeras se analizan distintas metodologías para realizar la descomposición de señales mediante Fourier; como son las series de Fourier, la transformada discreta de Fourier y la transformada rápida de Fourier [3] (más conocida como *Fast Fourier Transform*, *FFT*)

Las otras dos prácticas se centran en el proceso de modulación/demodulación de señales [4] utilizando portadoras analógicas y digitales. En primer lugar se caracteriza la modulación de los datos para, posteriormente, estudiar cómo varía su espectro, de la señal original, a la señal modulada.

La relación existente entre las cinco prácticas es la que se muestra en la figura 1.

Para llevar a cabo estas experiencias hemos desarrollado una serie de aplicaciones que los alumnos podrán usar para efectuar cada uno de los estudios. Por otro lado, y perteneciendo esta asignatura a los planes de estudios de Ingeniería en Informática e Ingenierías Técnicas en Informática de Sistemas y Gestión, no podíamos olvidar un aspecto como la posibilidad de que los alumnos programaran parte de la aplicación. Por tanto se les ofrece el ejecutable de la aplicación funcionando correctamente y se les pide que desarrollen el código correspondiente a la implementación del análisis de Fourier, o a la caracterización de la modulación de la señal.

2.1. Primera experiencia: Series de Fourier

En esta primera experiencia práctica los alumnos empiezan a tomar contacto con el análisis de Fourier. Para esto, se estudia el caso más sencillo que han visto durante las clases teóricas de aula, que es el uso de la descomposición de señales en series de Fourier.

El alumno conoce las expresiones para descomponer una función periódica *f(t)* con periodo *T* (esto es, que *f(t)=f(t+T)*) en series de Fourier. Como sabemos éstas son:

$$f(t) = \frac{a_0}{T} + \frac{2}{T}\sum_{n=1}^{\infty} a_n \cos(\omega_n t) + b_n sen(\omega_n t)$$

donde los coeficientes a_n y b_n se calculan mediante las siguientes expresiones:

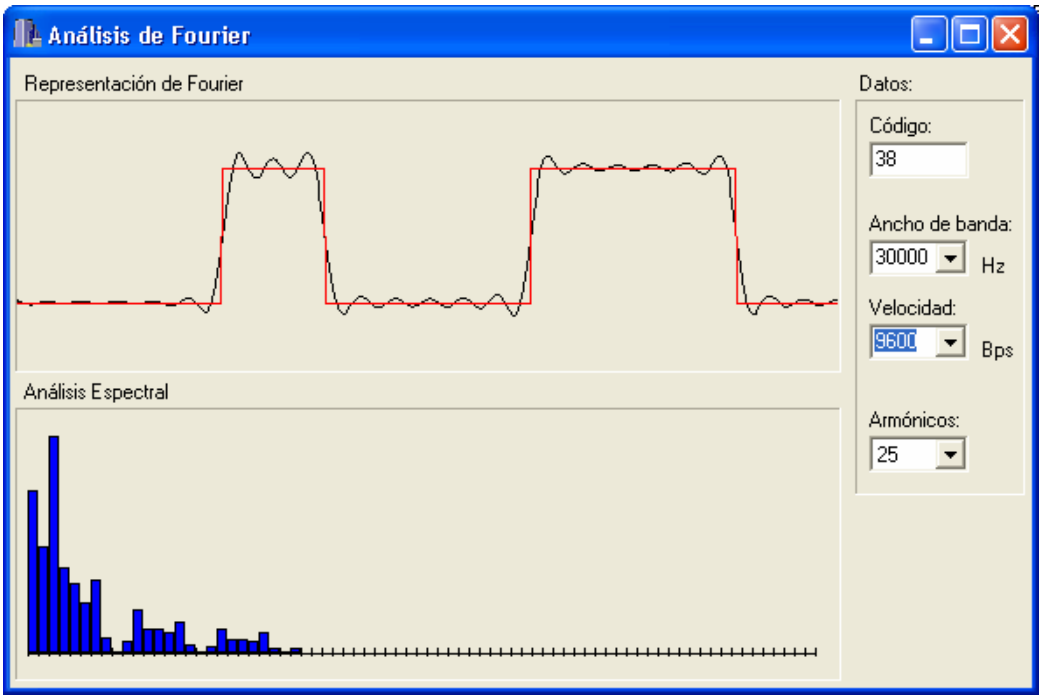


Figura 2. Entorno visual para experimentar con series de Fourier

$$a_n = \int_0^T f(t)\cos(\omega_n t)dt$$

$$b_n = \int_0^T f(t)\,\mathrm{sen}(\omega_n t)dt$$

Lo primero que se pide en esta experiencia es que el alumno obtenga una expresión general que permita calcular los coeficientes a_n y b_n para obtener el espectro de una señal periódica que represente la transmisión repetida de un mismo byte. De esta forma, el alumno realiza un primer ejercicio analítico que le obliga a estudiar y entender las expresiones para la descomposición de señales mediante series de Fourier.

El ejercicio simplemente consiste en descomponer cada una de las integrales definidas de 0 a T en ocho integrales, correspondiendo cada una de ellas al periodo de un bit dentro del byte (de 0 a $T/8$, de $T/8$ a $2T/8$, etc). De esta manera, cada integral estará multiplicada por uno o cero en función del valor del bit.

El siguiente ejercicio a desarrollar es implementar el cálculo de la potencia de un armónico n de la señal, usando la expresión

$$|C_n| = \sqrt{a_n^2 + b_n^2}$$

a partir del valor del byte y de la velocidad de transmisión, que marcará el periodo T de las anteriores expresiones (tiempo que tarda un byte en ser transmitido).

Una vez implementada esta sencilla rutina entramos en la parte fundamental de la práctica, en la que el alumno experimenta directamente con los distintos parámetros que permiten observar el espectro de un carácter o byte. Para esto se facilita un entorno visual como el mostrado en la figura 2. En él se puede introducir el código del carácter, la velocidad de transmisión, el ancho de banda del medio físico de transmisión y el número de armónicos que se desea representar.

Por simplicidad, en nuestro caso, el medio de transmisión se trata como un filtro paso bajo, de manera que el ancho de banda del medio marca el número de armónicos que el nodo destino puede usar para representar la señal. Por lo tanto, al modificar el ancho de banda automáticamente cambiará el número de armónicos que pasan por el medio, y a su vez si lo que se modifica es el número de armónicos que queremos representar, el valor del ancho de banda variará de manera automática para ajustarse al nuevo valor introducido y así permitir el paso de toda la señal.

Otro de los parámetros que se puede cambiar en tiempo de ejecución es la velocidad de transmisión, lo que afectará directamente al tiempo de transmisión del byte (T) y por tanto a la disposición de los armónicos dentro del espectro, ya que sabemos que los armónicos se sitúan en múltiplos de la frecuencia fundamental, o sea, de $2\pi/T$. Por tanto cuanto mayor es la velocidad de transmisión, menor número de armónicos son capaces de pasar por el medio.

Para observar el comportamiento de la señal en función de la variación de estos parámetros, la aplicación dispone de otras dos zonas de pantalla. En una de ellas se representa el espectro de la señal tal y como llega al nodo destino, es decir, después de ser filtrado por el medio de transmisión.

En una segunda área de la pantalla aparece la representación de las señales transmitidas y recibidas. En rojo aparece la señal que se quiere transmitir (la que generaría el nodo origen), sin ningún tipo de distorsión. En negro se representa la señal tal y como le llega al destino, después de ser filtrada por el medio, y se obtiene como la señal aproximada por un determinado número de armónicos usando series de Fourier.

En este punto se invita al alumno a que experimente y observe cómo se comporta la señal en función de los distintos parámetros de entrada, y la relación que existe entre el ancho de banda y la velocidad de transmisión, incentivando la obtención de conclusiones.

Posteriormente se sugiere al alumno que experimente con unos caracteres determinados. En primer lugar se introducen diferentes potencias de dos, que representan los mismos pulsos, de duración $\tau=T/8$, desplazados. De esta manera los alumnos observan como la envolvente es una función subamortiguada que tal y como han visto en las sesiones de teoría se caracteriza por *sen(x)/x*. Otros códigos de caracteres que los alumnos prueban son aquellos que producen un pulso más largo, y por tanto el valor de τ es mayor, de manera que comprueban como cuanto mayor es el valor de τ antes se produce el primer paso por cero del envolvente.

Estos ejemplos son simplemente dos muestras de cómo los alumnos usan esta aplicación, que en parte ha sido desarrollada por ellos, para entender progresivamente el significado de armónico y cómo la variación que introduce el medio físico afecta a la calidad de la señal.

2.2. Segunda experiencia: Transformada Discreta de Fourier (DFT)

Una vez el alumno ha trabajado con el análisis de Fourier a nivel teórico y entiende los conceptos básicos de armónico y análisis en frecuencia, se propone una segunda experiencia en la que se utilice el análisis de Fourier para un caso real.

En este caso la señal a analizar es una serie de muestras de audio obtenidas a partir de ficheros de sonido de Windows (extensión .wav). Para realizar el análisis tendrán que poner en práctica la segunda herramienta relacionada con Fourier que han estudiado durante las sesiones teóricas, la transformada discreta de Fourier (*discrete Fourier transform, DFT*)

En clases de teoría se estudia como las series de Fourier permiten realizar una descomposición de una señal periódica, y se introduce la transformada de Fourier para realizar un estudio de señales no periódicas, que serán tratadas como periódicas con un periodo que tiende a infinito. A su vez estudian como la versión discreta de esta herramienta matemática permite calcular los armónicos correspondientes a una señal muestreada. Justamente esto es lo que van a poner en práctica durante esta segunda experiencia que proponemos.

El primer ejercicio a realizar es programar la DFT para realizar un análisis espectral de un conjunto de muestras. Para esto los alumnos simplemente deben implementar las fórmulas vistas durante teoría, que permiten obtener la parte real e imaginaria de la DFT, de forma que calculan la potencia de cada uno de los armónicos. Una vez implementada la función para el cálculo de la DFT el alumno puede integrarla dentro la

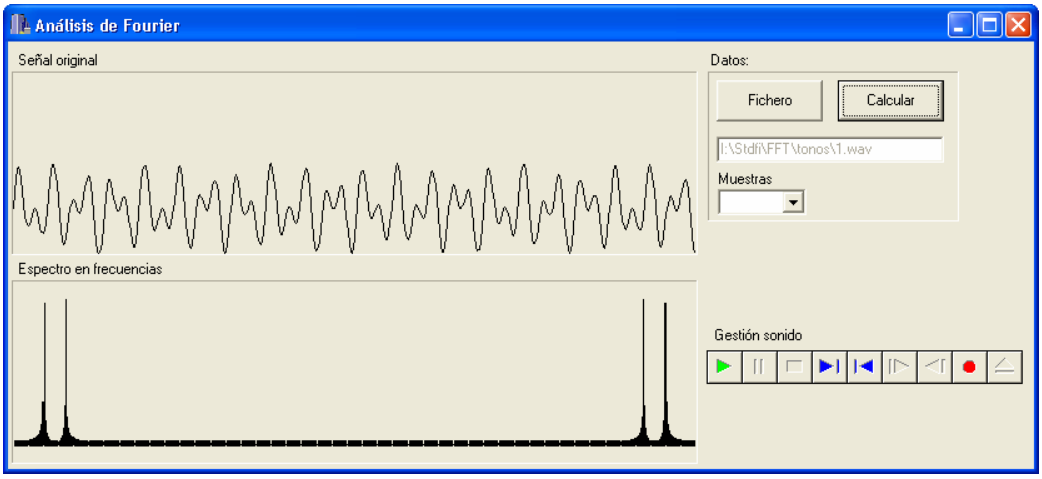


Figura 3: Entorno visual para experimentar con la transformada discreta de Fourier

aplicación visual que le permite observar la señal original y dibujar los armónicos de la DFT calculados.

En la figura 3 podemos observar una captura de la pantalla de esta aplicación. Mediante el botón *Fichero* el alumno puede cargar un fichero de audio y lo muestra sin transformar (en el dominio del tiempo). Para calcular la DFT y obtener su espectro en frecuencias basta con pulsar en el botón *Calcular* una vez cargado el fichero. También se puede escuchar el audio mediante los botones de gestión se sonido.

Para poder experimentar con los resultados de la DFT se propone como ejercicio el cálculo del espectro de los tonos de marcación multifrecuencia.

El sistema telefónico emplea dos posibles métodos para la marcación: por pulsos y por tonos. El primero traduce los números marcados en una serie de pulsos, separados por pausas, de tal modo que reflejan el número marcado. El segundo, más evolucionado, asigna a cada una de las posibilidades de marcación un tono.

Los tonos se generan mediante la suma de dos señales de frecuencias distintas. De esa forma, en lugar de emplear 16 tonos (los números de 0 a 9 y los caracteres especiales #. *, A,B,C,D) nos basta con ocho frecuencias. Así, el teclado de un teléfono funciona según se observa en la figura 4.

	F1	F2	F3	F4
F5	1	2	3	A
F6	4	5	6	B
F7	7	8	9	C
F8	*	0	#	D

Figura 4: Teclado telefónico multifrecuencia

Cuando se pulsa un número, por ejemplo el 1, el teléfono genera un tono como la suma de dos frecuencias, en este caso F1 y F5, enviando dicho tono a través de la red telefónica. Para la recepción, se pasa el tono recibido por ocho filtros paso banda, que permiten pasar las frecuencias de F1 a F8. Sólo dos de los filtros tendrán salida, por lo que mediante una matriz similar al teclado se reproduce el dígito marcado.

En este segundo ejercicio se facilita a los alumnos un conjunto de muestras de audio que representan cada uno de los sonidos emitidos al pulsar una tecla del teclado telefónico. De esta manera, al representar el espectro aparecen claramente los dos armónicos correspondientes a cada una de las frecuencias que componen un tono

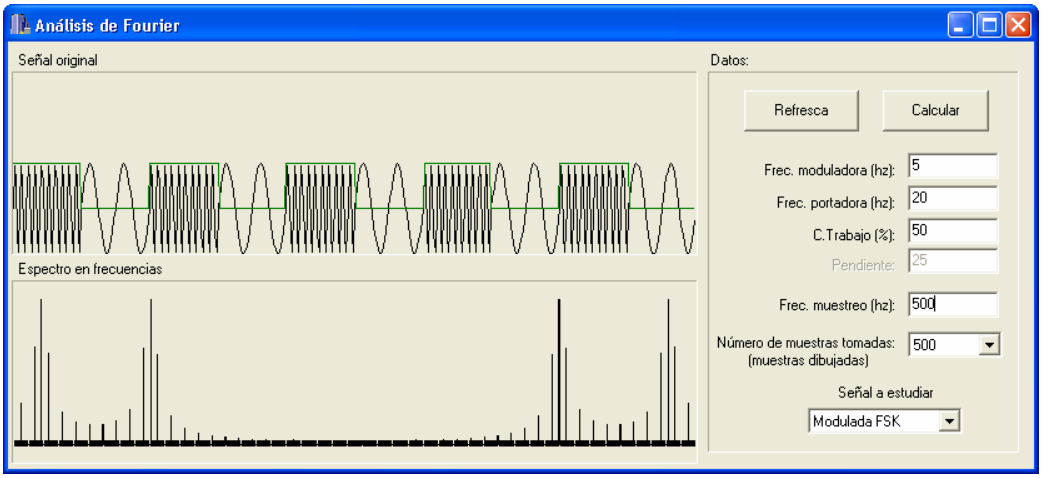


Figura 5: Entorno visual para el estudio en frecuencia usando la DFT de señales moduladas con portadora analógica

telefónico. Así por ejemplo, en la figura 3 se representa las muestras correspondientes a la tecla 1, y su espectro asociado. Otro efecto que el alumno puede apreciar es la réplica simétrica de la segunda mitad del espectro respecto a la primera mitad, que aparece al calcular el espectro de cualquier señal muestreada.

El ejercicio se completa pidiendo al alumno que identifique cada una de las frecuencias F1, F2,…, F8 que se representan en la figura cuatro, mediante la observación de los armónicos de mayor potencia en los espectros de las distintas muestras de sonido.

2.3. Tercera experiencia: Transformada Rápida de Fourier (FFT)

Durante la experiencia anterior el alumno posiblemente note que si el número de muestras a procesar por la DFT es grande, su cálculo se hace muy costoso. De hecho el coste de la DFT es de orden cuadrático respecto al número de muestras a procesar.

En esta tercera experiencia práctica se pide al alumno que implemente la versión rápida de la transformada discreta de Fourier, y que compare el tiempo de ejecución de esta versión respecto a su equivalente discreta, utilizando ficheros con distintos número de muestras.

Como sabemos, este algoritmo desarrollado por Cooley y Tukey en el año 1965 es bastante más rápido, presentando una complejidad del orden $O(n\log_2(n))$

De esta forma, esta tercera práctica resulta menos visual que el resto pero no por ello menos espectacular. De hecho el coste del cálculo de la FFT utilizando 1024 muestras es aproximadamente diez veces menor que usando la DFT, con lo que los alumnos adquieren una idea de lo importante que es el uso de optimizaciones en los algoritmos de procesamiento de datos, sobre todo cuando existen restricciones de tiempo real o el número de muestras a procesar es muy grande.

2.4. Cuarta experiencia: Estudio en frecuencia de señales moduladas usando la DFT

Después de las tres actividades desarrolladas por los alumnos, éstos ya han debido adquirir las habilidades necesarias para entender e interpretar los espectros en frecuencia de señales.

En esta cuarta experiencia se propone que el alumno estudie e implemente la caracterización del proceso de modulación de señal, tanto en el caso de portadora analógica y moduladora digital como en el caso opuesto.

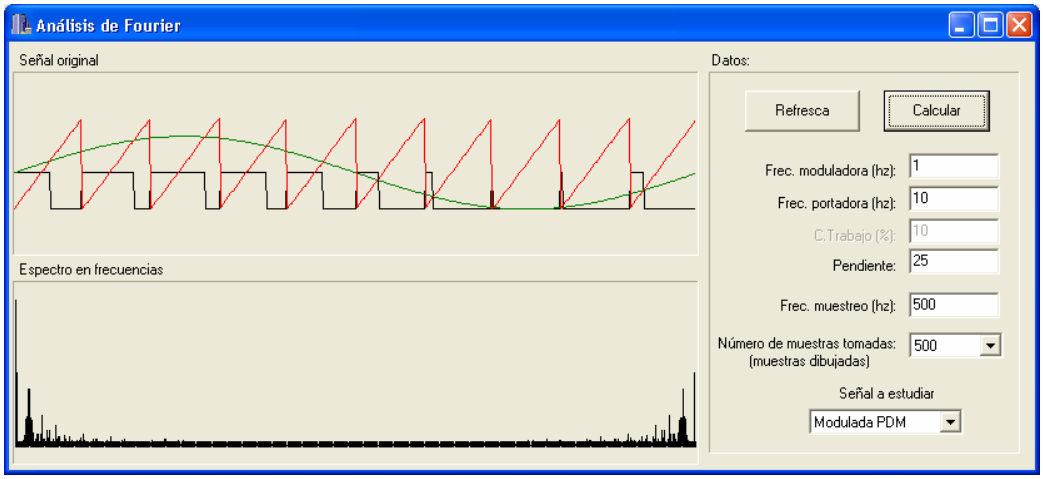


Figura 6: Entorno visual para el estudio en frecuencia de señales moduladas con portadora digital

En el caso de portadora digital se implementará las modulaciones en amplitud (ASK), en frecuencia (FSK) y en fase (PSK). De esta forma se propone al alumno que la señal a transmitir es un tren de pulsos y que deben de caracterizar cada una de las señales moduladas.

El resultado deberá ser un vector de muestras que represente la señal original modulada. De esta manera, usando la DFT o la FFT implementada en las experiencias anteriores, se puede obtener el espectro de la señal modulada y comparar con el de la señal original.

La figura 5 muestra el interfaz visual de la aplicación, que se usará para introducir datos y para la representación de las señales obtenidas. En la parte de arriba aparece el tren de pulsos modulado en frecuencia, en este caso usando FSK. La gráfica inferior muestra el espectro de la señal, con dos armónicos de mayor potencia que indican claramente las dos frecuencias presentes en la señal modulada, la de mayor frecuencia representando el nivel alto y la de menor el nivel bajo. En la parte derecha se puede modificar los parámetros de la modulación en tiempo de ejecución.

De esta forma, una vez se han implementado las funciones que caracterizan la modulación, el alumno puede experimentar introduciendo distintos parámetros a la modulación. Así, una primera experiencia es tomar el tren de pulsos y calcular su espectro, para luego modularlo mediante ASK y observar que el espectro resultante es el mismo que el original pero desplazado en función de la frecuencia de la portadora, y replicado a ambos extremos.

Un segundo grupo de modulaciones a implementar son las que usan portadora digital y moduladora analógica. Nosotros hemos seleccionado trabajar con modulación PAM, PDM y PPM. La figura 6 muestra un ejemplo de modulación PDM obtenida mediante la aplicación de prácticas, una vez implementada esta modulación. La moduladora analógica en este caso es una señal senoidal, y la modulada es un pulso a nivel alto, con un flanco de bajada en el momento en el que la pendiente corta la señal analógica. Por otro lado, en este tipo de modulaciones el espectro tiene menos interés y lo que se pide es que los alumnos calculen la energía total de la señal, de forma que se observa como una modulación PPM presenta como ventaja respecto a la PDM el tener una energía total mucho menor.

2.5. Quinta experiencia: Estudio en frecuencia de señales moduladas usando series de Fourier

Si en la experiencia anterior se realizaba el cálculo del espectro de las muestras de las señales moduladas, en este caso se pide hacer un desarrollo analítico para obtener los armónicos de

las señales moduladas con portadora analógica mediante el uso de series de Fourier.

En la cuarta experiencia se usa como señal moduladora un tren de pulsos y como señal portadora una señal senoidal. Ambas señales son periódicas y el resultado de las modulaciones ASK, FSK y PSK también lo serán. Por lo tanto, en este ejercicio el alumno debe realizar un desarrollo analítico similar al efectuado en la primera experiencia, para poder obtener una expresión general para el cálculo de los armónicos de cada una de las posibles modulaciones. Una vez implementadas las fórmulas resultantes, el resultado debe ser un espectro similar al obtenido con la DFT (ver figura 5).

3. Conclusión

En este artículo se ha presentado una serie de experiencias prácticas que han sido desarrolladas con el objetivo de incrementar la comprensión de conceptos fundamentales que involucran el análisis de Fourier, situándose siempre dentro del marco de los sistemas de transmisión de datos. El resultado obtenido ha sido favorable, observando como el alumno era capaz de comprender contenidos estudiados en sesiones teóricas, difíciles de entender desde un punto de vista exclusivamente matemático.

Como futuras ampliaciones se está trabajando en una caracterización más real de los medios físicos de transmisión, con el objetivo de obtener respuestas del medio que sean más reales que un simple filtro paso-bajo, en el que un número determinado de armónicos pasan completamente y el resto son cancelados.

Referencias

[1] Martin S. Roden. *Analog and digital communication systems* Ed. Prentice-Hall, 1991

[2] A. Bonastre, F. Buendía, M. Pérez. *Equipos y sistemas de transmisión de datos*. Servicio de publicaciones de la U.P.V., 1996

[3] H.J. Nussbaumer, *Fast Fourier Transform and convolution algorithms.* Springer, 1982.

[4] Mischa Swartz. *Transmisión de información, modulación y ruido*. Ed. McGraw-Hill, 1993

Tecnologías Web: una asignatura sobre tecnologías de Internet en las Ingenierías Informáticas

Otto Colomina, Ignacio Iborra, Miguel Ángel Lozano
Dpto. Ciencia de la Computación e Inteligencia Artificial
Universidad de Alicante
03080 Alicante
e-mail: {otto, nacho, malozano}@dccia.ua.es

Resumen

En este artículo se describe una propuesta para los contenidos teóricos y prácticos de la asignatura de Tecnologías Web en las Ingenierías en Informática de la Universidad de Alicante. Dicha asignatura aborda el desarrollo de aplicaciones web centrándose en la parte del cliente, y ha comenzado a implantarse en los nuevos planes de estudio de dicha Universidad en el curso 2001. Para terminar se describe la experiencia docente del pasado curso.

1. Introducción

En la actualidad, resulta superfluo insistir en el impacto que está teniendo el rápido crecimiento de internet y concretamente de la web en la actividad profesional del informático. Este debe enfrentarse al hecho de que la web es un contexto cada vez más común para el desarrollo de aplicaciones, lo que le obliga a dominar una serie de tecnologías asociadas. Por todo ello, se hace necesario introducir estas tecnologías en los nuevos planes de estudio de las Ingenierías en Informática.

En este sentido, el *Computing Curricula 2001*, de ACM e IEEE [1] recoge *"La web como ejemplo de sistema cliente-servidor"*, como una de las unidades pertenecientes al núcleo básico (*core*) de la disciplina, y por tanto necesariamente presentes en cualquier curriculum de un Ingeniero Informático. En esta unidad se propone el estudio de temas como *protocolos de web* o *servidores web*, y tecnologías como *lenguajes de* script, *applets*, CGIs, etc.

Siguiendo esta tendencia, en bs nuevos planes de estudio de la Universidad de Alicante [2] se han introducido diversas asignaturas directamente relacionadas con la web o con tecnologías de Internet. En este artículo tratamos la docencia de una de ellas, la asignatura de *tecnologías web*, que, como se verá se ocupa del desarrollo de aplicaciones web desde la perspectiva del cliente, fundamentalmente.

2. *Tecnologías web* en el contexto de los planes de estudio de la U.A.

En la tabla 1 se recogen las asignaturas de los nuevos planes de estudio de las Ingenierías Informáticas en la Universidad de Alicante directamente relacionadas con tecnologías de Internet.

Asignatura	Descriptores
Tecnologías web	**Lenguajes de especificación de páginas web. Lenguajes de *script*. Programación de clientes web. Animación para web. Seguridad**
Programación en internet	Sistemas de acceso a B.D. de Internet. Planificación, diseño y admon. de sitios web. Migración de aplicaciones a entornos Internet.
Administración de servicios de internet	Intranets y extranets de gestión de red. Servidores web. Servidores WAP. Servlets

Tabla 1: asignaturas relacionadas con internet en los planes de estudio de informática de la U.A.

Las tres asignaturas son de carácter optativo para las tres Ingenierías en Informática, con 3 créditos teóricos y 3 prácticos. En cuanto a

contenidos, como puede verse en la tabla, la asignatura de *Tecnologías Web* se ocupa fundamentalmente de la programación y desarrollo del lado del cliente, la de *Programación en Internet* del lado del servidor y la de *Administración de Servicios en Internet* de la gestión y administración de los servidores y sistemas de red.

3. Enfoque propuesto para la asignatura

Dados los descriptores que tiene la asignatura en el plan de estudios, nos planteamos un objetivo fundamental: que el alumno conozca las tecnologías empleadas en la programación de aplicaciones web y sepa qué aporta cada una de ellas, centrándose sobre todo en la parte del cliente. Como en una aplicación web "real" se empleará una combinación de tecnologías, no se trata únicamente de conocer una serie de lenguajes y/o herramientas, sino además de poder decidir cuándo se emplea cada una de ellas, y de saber integrarlas adecuadamente.

Por todo ello, y con el objetivo de realizar el aprendizaje en un contexto lo más "realista" posible, el hilo conductor de la asignatura es la realización de una aplicación web que integra todas las tecnologías vistas en el temario. Concretamente, en el curso 2001/2002 los alumnos implementaron una "tienda web". Esto se describe con más detalle en el apartado 4.2.

Otra cuestión importante a la hora de plantear la asignatura es su caracter marcadamente *tecnológico*. Es decir, no se trata tanto de aprender conceptos formales o matemáticos como de tecnologías. Así, por ejemplo, los alumnos no van a aprender programación orientada a objetos en general, sino que van a aplicar Java al desarrollo de *applets*. Esto, junto con el elevado porcentaje de créditos de laboratorio (50%) obliga a darle a la asignatura un marcado carácter práctico: en clase de teoría, además de los conceptos fundamentales, se deben mostrar numerosos ejemplos, demostrando en la clase el uso de las tecnologías.

Finalmente, no hay que perder de vista el hecho de que muchos alumnos ven la materia que se imparte en *tecnologías web* como "atractiva", y "novedosa". Esto hace que algunos de ellos empleen voluntariamente un esfuerzo considerable en la realización de los trabajos de la asignatura. Para dar la máxima libertad posible en este aspecto, cada práctica tiene una propuesta de partes optativas, aceptando también las hechas por parte de los propios alumnos. Además se intenta promover la realización de trabajos adicionales que aborden tecnologías que no es posible ver en el temario por cuestiones de tiempo.

4. Propuesta de contenidos

A continuación exponemos el programa teórico y práctico de la asignatura, desarrollado tomando como base las consideraciones expuestas en el apartado previo. Este temario, junto con apuntes y material complementario puede consultarse en el sitio web de la asignatura [3].

4.1. Temario teórico

En la parte teórica de la asignatura se intenta abordar, a un nivel de profundidad razonable dado el tiempo disponible, todas las tecnologías que se pueden emplear para construir una aplicación web. Dados los descriptores de la asignatura, nos centraremos sobre todo en la parte del cliente (navegador).

El temario consta de 6 temas, que se detallan en la tabla 2. La asignatura comienza con una introducción al protocolo HTTP, base de todas las tecnologías que se abordarán en los temas posteriores. Se continúa con *applets* de Java. Previamente se imparten seminarios de Java con el objeto de que los alumnos, que están en su mayoría familiarizados con C/C++ se adapten a la sintaxis de este lenguaje. El tema 3 es el de HTML en el que se intenta, más que ver todas las etiquetas del lenguaje, explicar su filosofía, técnicas de “buen uso” de código y hacer hincapié en la compatibilidad. En el tema siguiente se abordan los lenguajes de *script* en el navegador, centrándose en *JavaScript.* El tema se centra en las posibilidades que ofrece el lenguaje para manejar de forma dinámica el contenido y estilo de los documentos web. El tema posterior es el de XML, en el que se trata la sintaxis de este formato, su filosofía de separación entre contenido y estilo y su utilidad para el desarrollo de webs. Finalmente, el temario concluye con una introducción a multimedia y animación web.

Tema	Duración
Protocolo HTTP y aplicaciones web Sistemas cliente-servidor. Protocolos de Internet. Protocolo HTTP. Aplicaciones web	2 horas
Applets Java Implementación básica de un applet. El AWT: eventos y clases. El paquete java.applet. Seguridad. Threads y applets.	6 horas
HTML Lenguajes de marcado. El lenguaje HTML. Hojas de estilo en cascada (CSS). Normas de buen uso de código HTML.	6 horas
Programación de scripts Lenguajes de *script* en el navegador. El lenguaje JavaScript. Interacción con el navegador. Interacción con el documento HTML: el DOM. HTML dinámico. Seguridad.	6 horas
XML Sintaxis XML. Validación de documentos con DTDs. Hojas de estilo (XSL-T y XSL-FO). XML para sitios web. Lenguajes basados en XML: WML, XHTML,...	4 horas
Multimedia y Animación web Multimedia en web. *Streaming*. Lenguajes y herramientas para animación web	4 horas

Tabla 2: Temario de teoría de la asignatura

Durante las clases de teoría se intenta poner el máximo número de ejemplos posible, con el fin de que el alumno pueda ver el uso real de las tecnologías. Asímismo se hace hincapié en la conveniencia de adherirse a los estándares. En la actualidad, y aunque organismos como el W3C [4] están normalizando las tecnologías empleadas en la web, en la práctica muchos sitios web simplemente se adaptan al modo peculiar de funcionar de algunos navegadores, con los consiguientes problemas de compatibilidad. Aunque en el temario se abordan estas peculiaridades, también se insiste en el uso de "buenas prácticas" de codificación, que proporcionen en la medida de lo posible la independencia con respecto al *software* del cliente.

4.2. Temario de prácticas

La práctica consistirá en realizar una aplicación web utilizando para ello las principales tecnologías correspondientes al lado del cliente. La aplicación a realizar es una tienda online, de la que se proporciona el esqueleto básico de procesamiento y obtención de datos en el lado del servidor para que el alumno incorpore sobre ella todos los elementos necesarios para construir una tienda con todas sus funcionalidades.

La realización de esta aplicación completa se dividirá en una serie de prácticas, cada una de las cuales corresponderá a una de las tecnologías estudiadas, con las que el alumno creará de forma progresiva el *front-end* del sitio web. Estos bloques se dividen como se muestra en la tabla 3.

Figura 1: Ejemplo de página principal de la tienda web desarrollada en el curso

Nombre del tema	Duración
Applets Java: Programación de un *ticker*	6 horas
HTML: Diseño del sitio web	6 horas
Javascript: Carro de la compra	6 horas
XML: Rediseño del portal	4 horas
Flash: Animación	4 horas

Tabla 3: temario de prácticas de la asignatura

En cada tema se han proporcionado apuntes sobre dicho tema para el desarrollo de la práctica en cuestión, así como ejemplos sobre el uso de la correspondiente tecnología. En cada una de las prácticas se realizarán los componentes más adecuados para la tecnología de la que se trate.

En la práctica de *Applets Java* se pretende dar a conocer las técnicas principales de programación en *Java* de este tipo de elementos. Se imparte un seminario sobre programación general en *Java*, pasando luego a detallar los aspectos concretos para la programación de *Applets*. El alumno deberá desarrollar un *ticker,* en el que deberá pasar parámetros a un *Applet,* leer un fichero remoto de donde deberá extraer todos los mensajes a mostrar, y dibujar todos estos mensajes de forma animada, mediante la utilización de *Threads*, evitando el parpadeo mediante la técnica del doble *buffer*. Además de esto se verá también la interacción entre el *Applet* y el navegador, y la obtención de imágenes y clips de audio de localizaciones remotas, así como su reproducción en el *Applet.*

En la segunda práctica el alumno deberá construir el sitio web a partir del esqueleto que se proporciona para obtener la información en el servidor. Se deberá dar una estructura a esta información utilizando para ello las etiquetas del HTML. Se utilizarán marcos, tablas y otros elementos para la organización de los contenidos del documento, se definirán enlaces entre las distintas páginas que componen el sitio web, se estudiará el paso de parámetros y se realizará un formulario de búsqueda de productos. A los elementos utilizados se les dará un estilo utilizando para ello hojas de estilo CSS, y evitando el uso de etiquetas desaprobadas. Se pretende que el alumno genere el sitio web siguiendo las normas de buen uso del HTML.

Como lenguaje *script* en el cliente se estudia *javascript* que es el más común. En la tercera práctica se utilizará dicho lenguaje para añadir contenido dinámico a la página, así como la manutención de una cesta de la compra en el lado del cliente mediante *cookies*. Se verá la forma de acceder a las *cookies* del navegador, la creación, eliminación y modificación de las mismas, así como la encapsulación en objetos de los datos leídos de ellas. Además el carrito se deberá mostrar también como persiana desplegable, viendo así la forma de modificar dinámicamente el documento HTML, estudiando el DOM de los distintos navegadores.

La siguiente práctica consiste en la conversión del portal a XML, debiendo definir el alumno su propio lenguaje con el que se generará la información. A partir de este documento XML, deberá definir hojas de estilo XSLT y XSL-FO para la traducción de los documentos a HTML, WML y PDF

Como trabajo final se propone la utilización de Flash para la realización de algún elemento para la web, como puede ser una presentación o título animado.

4.3. Bibliografía y material de referencia

Si para la mayor parte de asignaturas universitarias es difícil encontrar textos que se adapten al temario impartido, la asignatura de *Tecnologías web* es aún más peculiar en este aspecto: la novedad de los temas impartidos, junto con la constante evolución de las tecnologías, hacen que no sea práctico emplear un texto de referencia. Baste señalar que durante el cuatrimestre en que se impartió la asignatura aparecieron 3 versiones sucesivas del servidor web empleado en las prácticas (*Apache Tomcat*), desde la *beta* hasta la definitiva. Por ello hemos optado por realizar apuntes de la asignatura (disponibles en el sitio web) y dar una referencia a los alumnos sobre los textos disponibles para cada tema a fin de que puedan ampliar conocimientos. En este sentido, hemos visto como referencias interesantes libros como [5] sobre el lenguaje Java, [6] sobre JavaScript, o [7] sobre HTML.

En este sentido, quizá la mejor referencia la tienen en la propia web, donde pueden consultar desde los estándares del W3C hasta multitud de sitios con cursos, tutoriales y ejemplos sobre las tecnologías abordadas en el curso.

5. Conclusiones

Al pertenecer la asignatura a los nuevos planes de estudio, el curso 2001/2002 es el primero en que se ha impartido, por lo que todavía es prematuro sacar conclusiones, aunque podemos adelantar las siguientes, basándonos en la experiencia docente y en las encuestas que cumplimentaron los alumnos al final del cuatrimestre.

En primer lugar, el correcto seguimiento de la asignatura supone una serie de requisitos previos que no todos los alumnos cumplían. Evidentemente, antes de programar *applets* es necesario saber Java, y antes de eso, dominar la programación orientada a objetos. No obstante, algunos alumnos se matricularon en la asignatura sin estos conocimientos, quizá "dejándose llevar" por el atractivo de la programación web. En sucesivos cursos insistiremos en los requisitos previos que tiene la asignatura para poder ser cursada con el máximo aprovechamiento.

En segundo lugar, el que los alumnos hayan tenido que realizar prácticas que funcionen correctamente en todos los navegadores hace que se hayan tenido que plantear ellos mismos la necesidad del uso de estándares en la web, por encima de las tecnologías propietarias.

Finalmente, destacar que en general, los alumnos perciben la materia impartida en la asignatura como "útil" e "interesante". Esto hace que algunos dediquen un esfuerzo considerable a la realización de sus prácticas, mucho más del estrictamente necesario. Por ello, nos planteamos en cursos siguientes profundizar en la posibilidad de realizar trabajos adicionales y proponer ampliaciones de la práctica que atraigan e interesen al alumno.

Referencias

[1] ACM/IEEE, *Computing Curricula 2001*, 2001, IEEE Computer Society/ACM Task Force on Computing Curricula.

[2] Planes de estudios conducentes a los títulos de Ingeniero en Informática, Ingeniero Técnico en Informática de Gestión e Ingeniero Técnico en Informática de Sistemas de la Universidad de Alicante, *BOE de 25 de septiembre de 2001*.

[3] Otto Colomina. *Página web de la asignatura Tecnologías Web* . www.dccia.ua.es/dccia/inf/asignaturas/TW

[4] World Wide Web Consortium, *página web del W3C.* http://www.w3c.org.

[5] P. Niemeyer, J.Knudsen. Curso de Java. Anaya Multimedia, 2000.

[6] D. flanagan. JavaScript: The definitive guide. O'reilly 2001.

[7] Musciano y Kennedy. HTML y XML: la guía definitiva. Anaya Multimedia, 2000.

Trabajos fin de carrera
e investigación de alumnos

El modelo de desarrollo para un Proyecto Fin de Carrera en Ingeniería Técnica en Informática

Agustín Cernuda del Río

Departamento de Informática - Universidad de Oviedo
Escuela Universitaria de Ingeniería Técnica en Informática de Oviedo
Facultad de Ciencias - C/ Calvo Sotelo, S/N - 33007 Oviedo
guti@lsi.uniovi.es

Resumen

El Proyecto Fin de Carrera de la Ingeniería Técnica en Informática presenta al alumno una problemática especial, y la actividad del profesor como guía adquiere gran relevancia. Además, este primer proyecto de un alumno puede tener gran influencia en su posterior comportamiento como profesional. Pensando en ciertos problemas que aquejan a la industria del desarrollo de software, parece apropiado aprovechar el Proyecto Fin de Carrera para que el alumno afronte su vida profesional prevenido contra estos problemas. En este documento se realizan diversas reflexiones sobre un posible modelo de gestión del proyecto que responda a estos fines, modelo que estamos aplicando en nuestros proyectos de Ingeniería Técnica en Informática. Precisamos que la problemática referida se hace especialmente patente en proyectos *nuevos*, no *de mantenimiento*, y nos limitaremos a ese ámbito. En este caso, además, hablaremos sólo de proyectos *individuales*.

1. La gestión de proyectos en el mundo profesional

El desarrollador profesional de software debe afrontar un panorama de notables desafíos. Dos tercios de los proyectos terminan claramente fuera de coste y plazo, los grandes proyectos muestran un retraso medio del 25% al 50% de la estimación inicial, y cuanto mayor es el proyecto mayor es la probabilidad de cancelación [2]. De 600 empresas encuestadas, el 35% tenía al menos un proyecto fuera de control [3]. Existen grandes dificultades para planificar costes y plazos, así como para mantener una calidad adecuada de los productos, un bajo coste de mantenimiento y un conjunto de prestaciones apropiado.

Muchos de los problemas que plantea el desarrollo de software no son de índole estrictamente técnica. Gran parte de estos inconvenientes no obedecen a dificultades con los lenguajes de programación, el análisis o el diseño, sino en el campo de la gestión de proyectos. Algunos de los errores clásicos que influyen de manera determinante en la pérdida del control de un proyecto [3] pueden verse en la Ilustración 1.

I.	Expectativas poco realistas
II.	Hazañas, ilusiones
III.	Planificación excesivamente optimista
IV.	Gestión de riesgos insuficiente
V.	Abandono de la planificación bajo presión
VI.	Escatimar en las actividades iniciales y/o en el control de calidad, en favor de la codificación
VII.	Programación a destajo
VIII.	Exceso de requisitos
IX.	Control insuficiente

Ilustración 1. Algunos errores clásicos.

Estos patrones de conducta (y podrían citarse muchos otros) se cuentan entre los más nocivos para la profesión, y son causa directa del fracaso de buena parte de los proyectos informáticos. Puede verse que en gran medida no se ven involucrados conocimientos estrictamente académicos, sino más bien hábitos de comportamiento. Según Roger Burlton, "las organizaciones que intentan implantar la disciplina de la ingeniería del software antes que la de gestión de proyectos están abocadas al fracaso".

2. El papel del Proyecto Fin de Carrera

A lo largo de sus estudios, un alumno tiene ocasión de caer en este tipo de hábitos e incluso de sufrir las consecuencias. Habrá comprobado más de una vez, por ejemplo, que *desear* terminar una práctica en una noche no lo hace *posible*. No obstante, es nuestra opinión que esa experiencia circunstancial y oportunista no es, en general, suficiente para su formación.

Apoya esta idea el hecho de que los desarrolladores profesionales incurren una y otra vez en este tipo de comportamientos "suicidas", bien porque no los identifican como la causa de fracasos anteriores, bien porque no perciben que las cosas puedan transcurrir de otra manera. Desgraciadamente, está muy extendida la idea de que en el desarrollo de software real "no hay tiempo" para actividades como control de calidad, buen análisis o diseño, saneamiento del código, etc. No parece razonable esperar que un estudiante con tan limitada experiencia en el desarrollo llegue por sí mismo a conclusiones más acertadas que los profesionales en un plazo tan breve.

Se impone, pues, transmitir al alumno unos hábitos que en el futuro le protejan frente a estos errores clásicos, de manera que al menos sea consciente de cuándo está haciendo un mal trabajo. Aunque estas normas básicas de actuación deben promoverse en todo momento y circunstancia durante el desarrollo de los estudios, entendemos que la piedra de toque definitiva es el Proyecto Fin de Carrera (en adelante PFC), puesto que:

- El alumno estará centrado en el proyecto, y podrá ser más consciente de su propio proceso de desarrollo que en los casos en que dicho desarrollo es sólo un medio para alcanzar conocimientos de alguna otra materia.
- El desarrollo del PFC guarda mayor parecido con el desarrollo de un proyecto real en cuanto a objetivos, técnicas y (especialmente) responsabilidades que el alumno debe asumir.
- Esta situación es la primera en la que se da tal parecido (en la mayoría de los casos).

Si se acepta esta idea, durante el desarrollo del PFC sería conveniente adoptar métodos de trabajo que formasen al alumno con vistas a evitar los males endémicos del desarrollo de software que se han citado más arriba. A nuestro juicio, sería muy conveniente contemplar los siguientes aspectos:

- Ciclos de vida
- Estimación
- Planificación
- Gestión de riesgos
- Calidad
- Documentación

El resto de este artículo está dedicado a proponer algunas prácticas cuyo fin es potenciar estos elementos para la consecución de los objetivos planteados anteriormente. Las normas de desarrollo de PFC que aplicamos recogen estas ideas. En cada apartado se presentarán, a modo de resumen, los hábitos recomendados, y una referencia a los "errores clásicos" de la Ilustración 1 a los que se pretende combatir en cada caso.

3. Ciclos de vida

Un punto fundamental para que el desarrollo de un PFC transcurra de manera satisfactoria es la selección de un ciclo de vida para el mismo. Quedarán después por resolver muchos detalles *tácticos* de planificación, pero esa tarea será aún más difícil sin una decisión previa a nivel *estratégico*.

La falta de visibilidad es un problema muy preocupante (y frecuente) en los proyectos de desarrollo de software. Quien va a recibir un producto llave en mano en cierta fecha lejana pero no tiene información fiable sobre el progreso del proyecto está asumiendo graves riesgos. Si hay errores de base se descubrirán demasiado tarde; si el proyecto fracasa, se habrá gastado eventualmente todo el dinero.

En el caso de un PFC este efecto es igualmente pernicioso. No es raro que se dé por supuesto un desarrollo en cascada, con las fases académicas tradicionales de análisis, diseño, codificación y pruebas. Este esquema, sin embargo, es difícil de aplicar con éxito en un entorno profesional (a menos que se gestione con gran habilidad el conjunto de requisitos o el proyecto sea muy similar a otros anteriores); no digamos para un alumno sin experiencia en

planificación ni estimación. Debe desarrollar dotes casi adivinatorias para definir en un solo paso la solución a un problema de envergadura notable. Durante el desarrollo afrontará una gran incertidumbre sobre la fecha de terminación o la marcha de los trabajos, ya que durante gran parte del proyecto no tendrá nada tangible o ejecutable.

Proponemos como remedio discutir de manera explícita cuál va a ser el ciclo de vida a adoptar, dependiendo de las circunstancias del proyecto. En este entorno, creemos que es adecuado casi en todos los casos algún ciclo de vida que produzca resultados intermedios: desarrollo en espiral, entrega por etapas, desarrollo evolutivo o similares. El alumno debe *ver algo* funcionando periódicamente. Eso le dará confianza, le ayudará a concretar el proyecto y le permitirá ejercitar la difícil habilidad de la estimación.

En particular, creemos también que en el caso de estudiantes es imperativo (y así se refleja en nuestras normas) el desarrollo de al menos un prototipo, ya sea de la interfaz de usuario, de viabilidad tecnológica o ambos a la vez. Por supuesto, el prototipo se desechará sin paliativos. El alumno habrá adquirido experiencia sobre cómo *no* debe hacer las cosas la segunda vez; parece la única forma de asimilar técnicas nuevas reduciendo el impacto negativo en la calidad del producto final, y ayuda además a mitigar la notable sensación de incertidumbre del alumno (recordemos que probablemente es la primera vez que debe resolver un problema en el que el enunciado no está cerrado, sino que es muy vago). Simplemente, se trata de aplicar el viejo consejo: *build one to throw away* [1].

Prácticas recomendadas:

- Elección explícita del ciclo de vida al principio del proyecto
- Realización obligatoria de al menos un prototipo (de IU, de viabilidad tecnológica o ambos)

Problemas mitigados: IX, I, II, III, VIII

4. Estimación

La estimación suele resultar especialmente difícil, porque sólo puede realizarse de forma rigurosa si se tiene implantado un buen sistema de medición sobre proyectos anteriores; y aun en ese caso pasa por fases de progresiva precisión, pero inicialmente hay que admitir un abultado margen de error. En la práctica, juegan un importante papel la experiencia y el instinto del desarrollador.

A la luz de esto, es evidente que un alumno de un PFC no se encuentra en una situación ideal para realizar buenas estimaciones (y así suele percibirlo él mismo). La estrategia que proponemos pasa por inculcar los siguientes principios, que serán de aplicación en el mundo profesional:

- Ya que el problema no tiene solución satisfactoria, lo mejor que tenemos es el criterio del desarrollador.
- Una estimación **no se negocia**.
- El desarrollador se compromete con sus propias estimaciones, que debe tratar de cumplir.
- Hay que dividir las tareas hasta llegar a un tamaño muy pequeño, y estimar esas mini-tareas. La estimación final se obtiene combinando las parciales (no se “inventa”).

Conviene hacer especial hincapié en que las estimaciones no se negocien. El alumno debe estimar el tiempo o esfuerzo lo mejor que pueda; una vez lo ha hecho, el deseo de su jefe (o profesor) no altera los hechos, y por tanto tampoco debe alterar la estimación (salvo que se le aporte información adicional que influya en la misma). El alumno debe aprender a dar (y reconocer) las malas noticias con valentía. Ceder y hacer las estimaciones más cortas sin ningún motivo racional es una de las prácticas más nocivas en el desarrollo de software.

En el ámbito de la estimación, puede ser de utilidad que el alumno mantenga una tabla de tareas, en la que vaya incluyendo cada tarea prevista, la fecha prevista de terminación, y la fecha real. En esta tabla quedará constancia de los errores de estimación, tanto por defecto como por exceso. El objetivo de esto no es “penalizar” al alumno por sus errores, sino hacerle consciente de las cifras previstas y reales (dándole así datos para

el futuro) y hacer que practique con frecuencia la estimación. Esta también es práctica obligatoria en nuestros PFC (Ilustración 2). La estimación de tareas pequeñas (de pocas jornadas, o incluso menos) ayuda a disminuir el error global, ya que los errores por exceso y por defecto se anulan. Por último, cada una de las tareas debe ser *binaria*: se entrega en cierta fecha o no, sin términos medios ni porcentajes.

Prácticas recomendadas:

- El alumno realiza sus estimaciones y se compromete con ellas.
- Se estima con frecuencia, y sobre tareas pequeñas.
- Se mantiene un registro riguroso de estimaciones, cumplimientos e incumplimientos.
- Las tareas están terminadas ó al 100% ó al 0%, sin paliativos.

Problemas mitigados: I, II, III

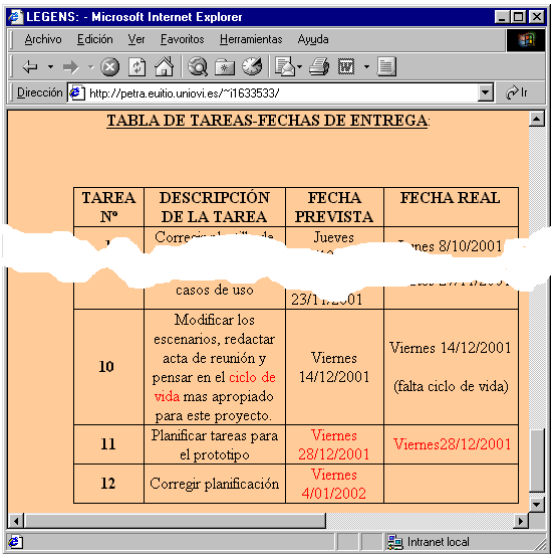


Ilustración 2. Tabla de tareas y fechas en la página web de un PFC

5. Planificación

Las dificultades de la planificación derivan en gran medida de las que plantea la estimación, añadiendo la colocación de las tareas, el establecimiento de dependencias y precedencias, etc. Cobra aquí particular importancia el efecto de las desviaciones.

Con el fin de evitar los problemas mencionados en el apartado 1, conviene vigilar estrechamente el desarrollo del PFC para evitar algunos males frecuentes, que tienen su raíz en una actitud muy común: **el abandono de la planificación bajo presión**. Cuando el plan se incumple, el desarrollador simplemente empieza a trabajar a destajo para acabar lo antes posible, ignorando el plan. Durante el PFC se debería hacer hincapié en evitar esto.

Proponemos que el alumno *siempre* trabaje sobre un plan. Además, este plan debe contener hitos binarios (que se cumplen o se incumplen en su totalidad); cuando se incumple un hito, se analizan las causas y se modifica la planificación de manera adecuada. No se abandona el plan. Sin embargo, es lícito rehacer el plan cuantas veces sea necesario. No se obliga al alumno a acertar en sus apuestas (al fin y al cabo, está aprendiendo), pero cuando falla en su apuesta, siempre debe sustituirla por otra.

Asimismo, el incumplimiento de hitos obliga a replanificar: se supone que el retraso producido probablemente se reproducirá de manera global en el proyecto. No es aceptable asumir que el tiempo perdido se recuperará por sí solo más adelante.

Proponemos también que el alumno lleve un "diario del proyecto", en el que anote los sucesos importantes. Los incumplimientos del plan y las acciones correctoras se reflejan siempre en este diario.

Prácticas recomendadas:

- El plan contiene hitos que se cumplen al 100% ó al 0%.
- El incumplimiento de un hito conlleva una replanificación.
- En el diario del proyecto queda constancia escrita de ambos sucesos.
- El alumno siempre trabaja sobre un plan; puede modificarlo repetidamente, pero no abandonarlo.

Problemas mitigados: V, VII, IX

6. Gestión de riesgos

Es muy frecuente que en el entorno profesional no se realice una gestión de riesgos efectiva; lógicamente, en un PFC esto es aún más habitual.

Sin embargo, el obligar al alumno a realizar una gestión de riesgos siquiera mínima tiene un gran valor didáctico. En primer lugar porque le hace consciente de los peligros que acechan al proyecto, y cuando alguno se manifieste estará sensibilizado; y en segundo lugar porque la gestión de riesgos es también un ejercicio adicional de planificación y gestión. Basta con un sencillo documento de una o dos páginas, en el que el alumno identifique los peligros importantes y adopte acciones preventivas (que pueden ser tan simples como cambiar el orden de ciertas tareas).

Un riesgo muy importante en los PFC (y que no se produce tanto en entornos profesionales) es el alargamiento indefinido del proyecto. Es frecuente en todo tipo de alumnos, pero más en el caso de alumnos que encuentran trabajo antes de obtener la titulación. Para mitigar este riesgo, es aconsejable que el tutor obligue a un progreso constante, aunque sea muy lento; esto entronca con el hábito ya mencionado de que toda tarea tenga una fecha de entrega prevista que el alumno se esfuerce por cumplir.

Prácticas recomendadas:

- Una gestión de riesgos, siquiera elemental.
- Norma de obligado cumplimiento: un avance constante (aunque sea lento) del proyecto. En caso de abandono manifiesto, cancelación del mismo.

Problemas mitigados: IX, V

7. Calidad

Quizás la enseñanza más importante que el alumno puede obtener del PFC es que la calidad no es una cualidad adicional que se puede incorporar al proyecto como una prestación más (o renunciar a ella), sino que debe ser algo intrínseco a cualquier proyecto. Y por una sencilla razón: la calidad es precisamente el camino para terminar el proyecto en menos tiempo y con menos esfuerzo [3].

El PFC es el momento adecuado para luchar contra el mito de que los proyectos reales son artesanía. Si el alumno termina su formación técnica y ya en la etapa final de la misma ve que lo estudiado en otras asignaturas sobre la calidad es sólo teoría, es de esperar que cederá con toda facilidad a las presiones del entorno profesional para desarrollar más rápido, "a costa de la calidad".

No resulta fácil resumir en pocas actividades cómo debe vigilarse la calidad; es una guerra que exige luchar en muchos frentes. Simplemente, debe garantizarse que el alumno comprende claramente que se espera de él que desarrolle software de calidad, *en su propio beneficio* (por puro egoísmo). Deben aplicarse todas las enseñanzas que se transmiten en el Plan de Estudios sobre cómo debe desarrollarse el software.

Una actividad que puede resultar muy beneficiosa en este sentido es la realización de revisiones técnicas formales (RTF), que nosotros proponemos según el modelo descrito en [4]. Además de mejorar la calidad del proyecto en cuestión, si estas RTF se realizan de manera cruzada entre miembros de distintos proyectos, pueden constituir un excelente ejercicio de análisis, diseño y codificación en un problema distinto del que ocupa a cada alumno en su propio PFC, aportándoles una experiencia extra y una visión más amplia del desarrollo.

Por supuesto, son aplicables las mismas reservas que en las RTF entre profesionales: hay que conseguir que los alumnos no consideren a los participantes como enemigos o censores, sino como personas que están ayudándole a terminar su trabajo antes y mejor. De hecho, como personas que están *haciéndole parte del trabajo*. Cada defecto señalado en una RTF ahorrará al alumno muchas horas de depuración, y así debe percibirlo.

Como inconveniente, cabe señalar que los alumnos de otros PFC probablemente no estén en buena situación para comprender un problema que les es ajeno. Esto exige del profesor una selección adecuada del material que se somete a RTF cruzada con otros alumnos, de manera que sea *digerible* por ellos. Otras partes críticas del sistema que no puedan ser revisadas por estos alumnos pueden someterse a RTFs en las que sólo

participen el alumno afectado y algunos profesores (al menos, lógicamente, el director del proyecto).

Prácticas recomendadas:

- En general, aplicar de manera decidida y rigurosa los fundamentos del desarrollo.
- Realizar revisiones técnicas formales, cruzadas con otros alumnos o sólo con profesores.

Problemas mitigados: VI, VII, V

8. Documentación

En realidad, este punto es simplemente otra faceta del problema de la calidad. La necesidad de una buena documentación es un caballo de batalla en los entornos profesionales, y también en los académicos.

Aquí merece la pena llamar la atención sobre la documentación relacionada con la gestión del proyecto en sí. Cuando se habla de "documentación", con frecuencia se piensa en documentación de análisis y diseño, que describe el programa, probablemente con el fin de facilitar su mantenimiento. Pero siendo este un tema importante, no es el único. Conviene reflexionar sobre la documentación del proyecto, que no se refiere de manera directa al programa sino a lo que rodea a su desarrollo.

Como ya se ha dicho, en muchos casos los problemas de un proyecto no son técnicos, sino de gestión. La falta de control del proyecto, la ausencia de planes precisos y ordenados o la falta de información sobre decisiones adoptadas previamente puede ser tan nociva como una mala descripción del código fuente. Existe un problema adicional con este tipo de documentos, que nuevamente va más allá del aspecto técnico: la documentación sobre el proyecto corre el riesgo de convertirse en burocracia, que el alumno percibe como inútil y que realiza cumpliendo un mero trámite.

La documentación sería motivo para un estudio pormenorizado por sí misma, pero aquí nos limitaremos a algunos apuntes relacionados con el tema que nos ocupa.

Utilidad. Es fundamental que la documentación sea para el alumno *una herramienta* y que la perciba como tal. El director del proyecto puede aprovechar algunas de las dificultades específicas que el PFC plantea al alumno (indefinición, incertidumbre, falta de control, dificultad...) para que dicho alumno encuentre en sus propios documentos el apoyo que necesita. Contra la indefinición, una buena lista de requisitos, que pueda consultar y actualizar con frecuencia; contra la incertidumbre, una buena planificación, y así sucesivamente.

Creemos que puede ayudar a esto el que el alumno mantenga una buena organización y accesibilidad de los documentos; imponemos como norma obligatoria el mantenimiento de una página web del proyecto, donde se encuentra toda la documentación (Ilustración 3).

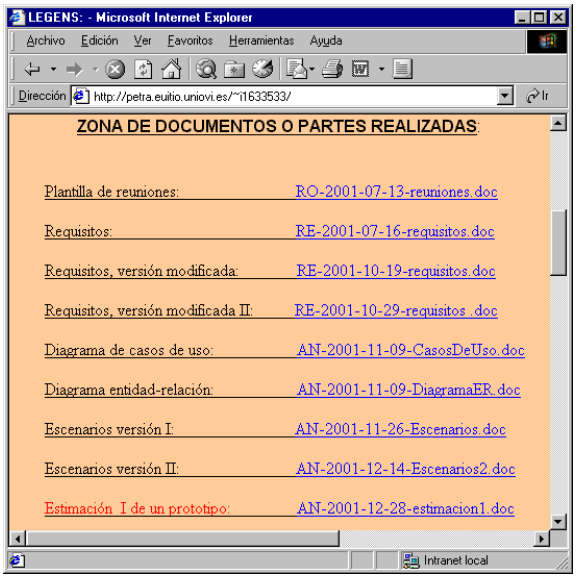


Ilustración 3. Página web con la documentación de un PFC

Burocracia. Hay que evitar a toda costa que el alumno perciba la documentación como un ejercicio de burocracia gratuita. Además de conseguir que la documentación le resulte útil y

aprecie esa utilidad, hay que procurar que se reduzca al mínimo posible.

El que la documentación no tiene por qué convertirse en burocracia queda patente si pensamos en los documentos realmente necesarios (aparte de la documentación de análisis/diseño habituales):

- Un documento de requisitos preciso (probablemente entre 1 y 5 páginas). Esto puede considerarse documentación de análisis, pero tiene un papel activo en la gestión del proyecto.
- Un solo documento de planificación y estimación (tipo Microsoft Project)
- Un documento de gestión de riesgos (1-2 páginas)
- Un "diario del proyecto" en el que se reflejen los sucesos importantes (y en este no tiene importancia una mayor longitud)
- Un historial de fechas de entregas parciales (las mismas consideraciones que en el apartado anterior)
- Otros documentos "históricos" (como actas de reuniones, RTFs, etc.)

Este simple conjunto de documentos permite realizar la mayor parte de las acciones sugeridas aquí, y parece claramente manejable.

Sincronización. El típico problema de que la documentación se adecue a la realidad, y cambie con ella, puede abordarse también adoptando los hábitos de trabajo adecuados. Si el alumno *piensa* sobre sus documentos, y se guía por ellos, cada nueva decisión quedará reflejada en primer lugar de forma natural en los documentos afectados, y posteriormente se llevará a la práctica. Esto puede ser más difícil de conseguir en la documentación que alude directamente al código, pero en los documentos del proyecto (requisitos, planificación, etc.) parece viable, puesto que la codificación es una tarea a la que el alumno está habituado y ha adquirido ciertos patrones de comportamiento, pero el desarrollo de un PFC es una experiencia nueva.

9. Conclusiones

En este artículo se realiza una reflexión sobre el especial papel que el Proyecto Fin de Carrera cumple en la formación de un Ingeniero Técnico en Informática de cara a su futura actividad profesional. Se identifican algunos de los problemas más graves del desarrollo de software en entornos profesionales, y se analiza su relación con algunos aspectos del desarrollo de un PFC; simultáneamente, se sugieren algunas prácticas recomendables con el fin de que el alumno adquiera hábitos que le conviertan en un profesional mejor preparado para luchar contra los problemas descritos.

En general, de lo que se trata es de que el alumno perciba que el desarrollo con arreglo a las directrices de la profesión que se transmiten en los estudios universitarios resulta *práctico* y redunda en su propio beneficio. Asimismo, se pretende que adquiera una clara conciencia de que el dominio de la técnica, por sí solo, no garantiza un buen rendimiento profesional; se necesitan ciertos hábitos de trabajo (y herramientas adicionales) que ofrezcan un marco adecuado a la labor técnica y la apoyen.

Referencias

[1] Frederick P. Brooks. *The Mythical Man-Month, Anniversary Edition: Essays on Software Engineering*. Addison-Wesley Pub Co; ISBN: 0201835959, Julio 1995.

[2] Jones, Capers. Assessment and Control of Software Risks. Englewood Cliffs, N. J.: Yourdon Press (1994).

[3] McConnell, Steve. *Desarrollo y gestión de proyectos informáticos*. McGraw-Hill, 1997.

[4] Software Formal Inspections Guidebook (NASA-GB-A302). Office of Safety and Mission Asurance, NASA (Agosto de 1993). Disponible en http://satc.gsfc.nasa.gov/fi.

Experiencia de realización de proyectos fin de carrera en el área de los sistemas multi-agente

Antonio Moreno, Aïda Valls
Dep. de Ingeniería Informática y Matemáticas
Escuela Técnica Superior de Ingeniería
Universidad Rovira i Virgili (URV)
Av. dels Països Catalans, 26, 43007-Tarragona
{amoreno,avalls}@etse.urv.es

Resumen

El *GruSMA* (Grupo de trabajo en Sistemas Multi-Agente) proporciona un entorno en el que alumnos de primer y segundo ciclo de Ingeniería Informática de la Universidad Rovira i Virgili (URV) pueden realizar Proyectos Fin de Carrera en el área de los Sistemas Multi-Agente. En esta ponencia describimos la motivación para su creación, su metodología de trabajo y los resultados obtenidos hasta el momento.

1. Introducción

Los autores de esta ponencia pertenecen al grupo de investigación en Inteligencia Artificial de la URV (Banzai, [3]). Parte de la actividad investigadora de este grupo se ha centrado en los últimos años en el tema de los *Sistemas Multi-Agente* (SMA); por eso nuestro objetivo principal era hacer que a los alumnos les pareciera atractivo el realizar el PFC en esta área. Para ello se les había de proporcionar un entorno en el que pudieran realizar este tipo de proyectos. Con este propósito en mente se ideó el *GruSMA* (Grupo de trabajo en Sistemas Multi-Agente, [9]). En esta ponencia, que extiende el trabajo presentado en [15], describimos las circunstancias de creación del grupo, su metodología de trabajo y los proyectos fin de carrera ya presentados. La ponencia acaba con una valoración de esta experiencia docente.

2. Creación de GruSMA

En el curso 98-99 ya se empezó a pensar en la posibilidad de crear este grupo. Lo primero que se hizo fue estudiar las diferentes herramientas de construcción de SMAs existentes en el mercado, para ver si había alguna adecuada (con suficiente calidad, ejecutable en PCs y, a ser posible, gratuita) para realizar los PFCs. Durante este curso un alumno hizo en su PFC un detallado estudio comparativo de diferentes entornos de construcción de SMAs ([10]). De este trabajo se obtuvieron dos conclusiones importantes:

- Hay dos grandes corrientes de estandarización del diseño e implementación de SMAs. En una, fundamentalmente utilizada en universidades americanas, los agentes se comunican utilizando un lenguaje llamado KQML (*Knowledge Query Manipulation Language*, [6]). La otra, con amplia base europea, está liderada por una organización llamada FIPA (*Foundation for Intelligent Physical Agents*, [7]) que ha definido los elementos básicos que deben componer un SMA, un lenguaje de comunicación entre agentes llamado FIPA-ACL (FIPA-Agent Communication Language), un lenguaje de representación de

conocimiento llamado FIPA-SL (FIPA-Semantic Language), un conjunto de protocolos de intercambio de mensajes, etc. Nosotros elegimos utilizar esta segunda opción, debido a la posibilidad (comentada más adelante) de utilizar buenas herramientas de construcción de sistemas multi-agente de libre distribución que siguen las especificaciones de la FIPA. En los últimos años esta segunda alternativa se está imponiendo poco a poco como un estándar de facto a nivel mundial.

- Existían ya (en verano de 1999) una gran colección de entornos de construcción de sistemas multi-agente lo suficientemente buenos y fiables como para plantearse la realización de PFCs en esta área. La mayoría de ellos son librerías en Java que proporcionan las clases necesarias para poder crear de forma relativamente sencilla un sistema multi-agente. La principal diferencia entre las diversas alternativas reside en las herramientas de soporte al programador que ofrecen.

Así pues, en septiembre de 1999 se creó el GruSMA.

3. Metodología de trabajo

La metodología de trabajo de GruSMA durante los cursos académicos 99-00, 00-01 y 01-02 ha sido la siguiente. En octubre, cuando se inicia el curso académico y los alumnos de cursos finales ya empiezan a pensar cómo harán el PFC, se hace publicidad de la posibilidad de realizar proyectos en el área de los Sistemas Multi-Agente. Esta publicidad se hace colgando carteles, realizando una sesión informativa y enviando mensajes a alumnos que hayan realizado asignaturas relacionadas con la Inteligencia Artificial durante el curso anterior. A finales de octubre ya se sabe cuántos alumnos están interesados en formar parte del grupo. En estos cursos todavía no ha sido necesario realizar ninguna selección: en el curso 99-00 hubo sólo 2 alumnos (ambos acabaron el PFC), en el curso 00-01 ya hubo 7 alumnos (4 de los cuales ya han presentado el PFC), y en el curso actual hay 10 alumnos realizando el PFC para presentarlo en junio o en septiembre de 2002. Estimamos que un grupo de estas características puede llegar a tener como máximo unos 10 alumnos, ya que sería muy difícil llevar más de 10 PFCs distintos entre los dos profesores que coordinan las actividades del grupo. Por tanto, suponemos que, a partir del curso 2002-2003, será necesario un proceso de selección previo.

En noviembre y diciembre se realizan las sesiones de trabajo del grupo (2 horas semanales durante 6-7 semanas). En estas sesiones los profesores responsables del grupo empiezan explicando lo que es un agente y lo que es un SMA. La explicación teórica se ha ilustrado este curso con la descripción y demostración práctica de los 2 PFCs realizados en el curso 99-00 ([18],[5]) y los 4 PFCs presentados en el curso 00-01 ([19], [12], [8], [13]). Después se describe con detalle el estándar de construcción de SMAs definido por la FIPA ([7]). Se hace especial énfasis en el lenguaje de comunicación entre agentes (FIPA-ACL), los protocolos de intercambio de mensajes y la definición e implementación de ontologías en FIPA-SL.

A continuación se dedican varias sesiones de trabajo a explicar el funcionamiento de un entorno concreto de construcción de SMAs, llamado JADE (*Java Agent Development Environment*, [4]). Este entorno está siendo desarrollado en el laboratorio italiano CSELT, y se distribuye de forma gratuita bajo las condiciones de la *GNU Lesser General Public License*. JADE proporciona una serie de clases Java que el usuario ha de utilizar para desarrollar sus propios agentes. Nos decidimos por este entorno de construcción de Sistemas Multi-Agente por las siguientes razones:

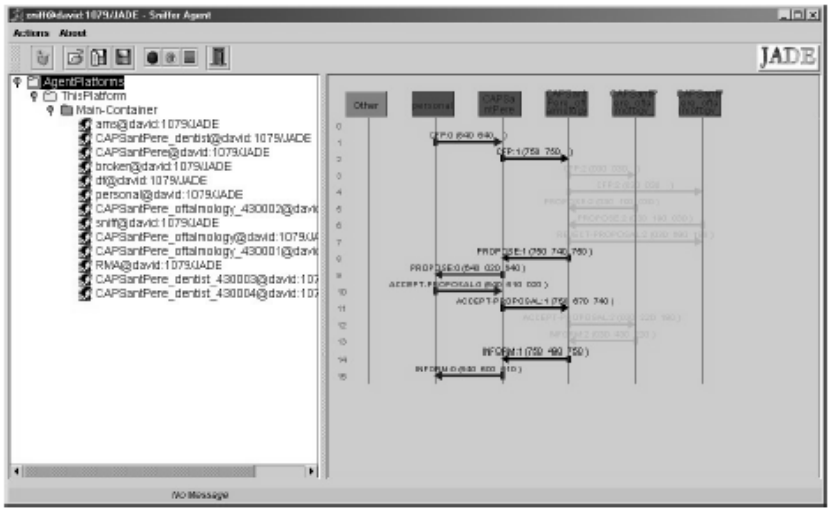


Figura 1-Interfície gráfica de JADE

- Las pruebas que realizamos sobre él nos mostraron que es un entorno fiable. Esta apreciación ha sido corroborada durante la realización de los PFCs.
- Es gratuito. Se puede bajar por Internet [11], instalar y utilizar muy fácilmente sobre cualquier ordenador que tenga Java 1.2.
- Sigue las especificaciones de la FIPA (estructura del SMA, lenguaje y protocolos de comunicación, etc.).
- Proporciona al usuario un entorno gráfico en el que se pueden hacer acciones como ver los agentes del sistema, añadir o eliminar agentes, o seguir los mensajes que éstos se van intercambiando (ver figura 1).

Existe una lista de distribución muy activa en la que los usuarios del sistema pueden enviar dudas, intercambiar experiencias o sugerir nuevas incorporaciones al sistema.

Los 2 PFCs del curso 99-00 se hicieron sobre la versión 1.3, y los 4 del curso 00-01 han utilizado la versión 2.01, adaptada a los cambios en las especificaciones de la FIPA realizados en el año 2000. Ahora acaba de aparecer la versión 2.5 de JADE, que permitirá realizar PFCs sobre entornos Java móviles (p.e. PDAs) el próximo curso.

4. Trabajo individual y en grupo

Otro de los objetivos de crear un *grupo* de trabajo es que los alumnos se encuentren en un entorno amigable, en el que se enfrenten en común a diferentes problemas y puedan ayudarse los unos a los otros. Este aspecto debe fomentarse, de manera que sea más atractivo para los alumnos pasar a formar parte de un colectivo con intereses similares que enfrentarse a la realización de un PFC de forma aislada. Este aspecto colectivo se ve reforzado, por un lado, por el encuentro semanal en las reuniones de trabajo. Por otro lado, los coordinadores de GruSMA buscamos problemas que sean de interés para todos los miembros, y hacemos que se enfrenten en común a ellos antes de ponerse a trabajar cada uno en su PFC en particular. Este tipo de problemas incluyen aspectos diversos de JADE como estudiar la comunicación de agentes ejecutándose en diferentes

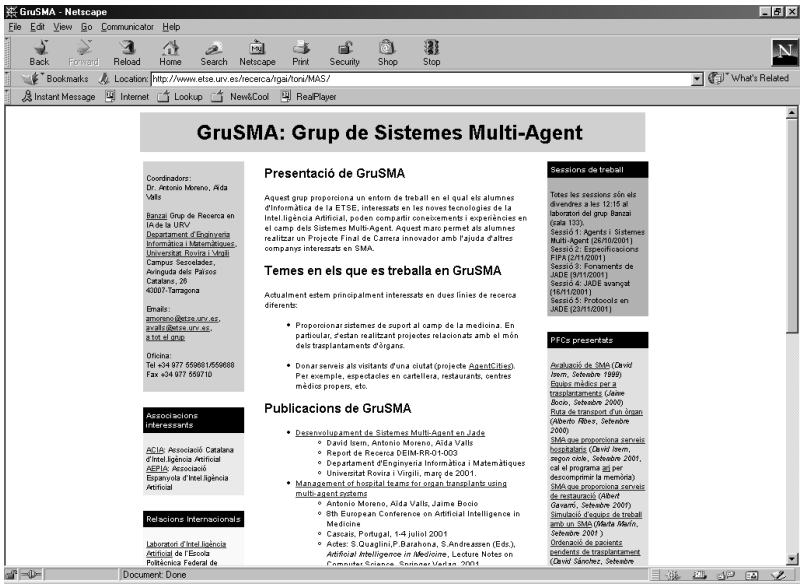


Figura 2-Página web de GruSMA (http://www.etse.urv.es/recerca/banzai/toni/MAS)

PCs, acceder a la información de una base de datos desde un agente, estudiar técnicas de coordinación de uso general, etc. Este tipo de problemas les son planteados a mediados de diciembre. Hacia finales de enero ya han estudiado estos problemas, y ponen en común las soluciones encontradas para que cada uno sepa resolver ese tipo de situaciones en su PFC particular. Hay que hacer notar, sin embargo, que la parte básica del trabajo se ha de desarrollar de manera individual. A principio de febrero, una vez han acabado los exámenes del primer cuatrimestre, ya se definen los PFCs individuales y cada alumno trabaja en su tema concreto. A estas alturas ya se conocen todos los miembros del grupo y saben que pueden contar con los demás para ayudarles en cualquier problema concreto que se puedan encontrar a la hora de utilizar JADE. Los PFCs finalizados se pueden presentar en junio o en septiembre.

Los alumnos disponen de toda la información que se maneja dentro de GruSMA en una página web ([9], ver figura 2). En esta página se puede encontrar todo el material de las sesiones de trabajo, el código fuente y ejecutable de JADE, los datos de los miembros del grupo, los PFCs propuestos y asignados, las memorias de los PFCs ya presentados, el calendario de las sesiones de trabajo, vínculos a las páginas de FIPA y de JADE, etc.

5. Proyectos final de carrera

Los coordinadores de GruSMA decidimos hacer que todos los PFCs desarrollados en el grupo fueran sobre un conjunto restringido de temas (manteniendo, en todo momento, un nivel alto de exigencia académica y una diferencia clara de calidad y trabajo personal entre los de primer ciclo y los de segundo ciclo). La motivación de esta decisión era doble. Por un lado, el hecho de trabajar varios alumnos en un mismo tema fomenta el espíritu colectivo del grupo. Por otro, se creaba la posibilidad de unir diferentes SMAs desarrollados en varios PFCs en un solo Sistema

Multi-Agente que solucionara algún problema complejo (en la sección 6.2 haremos un comentario a este respecto). Hasta ahora se han desarrollado 6 PFCs, básicamente centrados en dos áreas diferentes: el transplante de órganos y AgentCities.

5.1. Transplante de órganos

El principal problema que decidimos tratar fue la gestión de la coordinación de transplantes de órganos entre los hospitales de toda España. En el grupo de investigación en Inteligencia Artificial de la URV estamos especialmente interesados en la aplicación de SMAs en entornos médicos, y de ahí surgió la idea de usar este dominio ([2]). El problema es lo suficientemente complejo como para abarcar gran cantidad de PFCs individuales, que son del siguiente estilo:

- Crear el SMA interno de un hospital, con agentes que se comunican con otros hospitales, agentes que acceden a la base de datos local de pacientes en espera, agentes que se comunican con los médicos, agentes que reciben los datos de los órganos disponibles, etc.
- Crear el SMA de coordinación entre hospitales a nivel español, con un coordinador nacional, diversos coordinadores regionales y autonómicos, coordinadores locales, y un coordinador en cada hospital (PFC de segundo ciclo en curso).
- Tener un agente que se ocupe del tratamiento de los pacientes de emergencia 0, que son aquellos que ya han llegado a un estado crítico en el que necesitan un órgano en un período muy breve de tiempo (PFC de primer ciclo en curso).
- Crear agentes que sepan, dado un órgano disponible, aplicar técnicas de toma de decisiones multi-criterio para encontrar el paciente en espera que se adecúe más a sus características (PFC de primer ciclo ya presentado, [19]).
- Una vez determinado el paciente que recibirá un órgano, buscar medios de transporte que permitan llevar el órgano lo más rápido posible al hospital de ese paciente (PFC de primer ciclo ya presentado, [18]).
- Mientras se va transportando el órgano, tener un SMA en el hospital receptor que se encargue de buscar el equipo médico adecuado para realizar la operación y que gestione la reserva de quirófanos (PFC de primer ciclo ya presentado, [5]).
- Tener agentes que mantengan un registro histórico de las operaciones realizadas y puedan aplicar técnicas de minería de datos para obtener información útil sobre la asociación entre pacientes y órganos.

Este problema permite ser abordado de forma natural mediante la utilización de SMAs, al estar la información necesaria distribuída espacialmente en ordenadores de toda España (en hospitales, organizaciones de coordinación de transplantes, bases de datos de medios de transporte, etc.), y ser clara la necesidad de una comunicación y una coordinación entre los hospitales de toda España para asegurar una distribución rápida y justa de los órganos entre los pacientes en espera.

5.2. AgentCities

AgentCities.RTD ([1]) es un proyecto financiado por la Unión Europea que empezó sus actividades el 1 de julio de 2001. El objetivo principal de este proyecto es la creación de un entorno innovador en el que se pueda experimentar la composición dinámica de servicios heterogéneos online. Este entorno se construirá utilizando estándars y tecnología definidos por la FIPA, de forma que se permita la interoperación y comunicación de agentes inteligentes. Se desea que los miembros del proyecto sean capaces de construir todo un conjunto de plataformas que ofrezcan servicios basados en agentes. Este proyecto está

acompañado por una red europea, AgentCities.NET, que ofrece un cierto apoyo a las instituciones europeas que deseen formar parte de la experiencia de creación de este conjunto de servicios basados en agentes. Esta red empezó sus actividades el 1 de noviembre de 2001; actualmente ya tiene más de 30 miembros, de los cuales 8 son españoles. En la última reunión de esta red (Lausanne, 8/2/2002) hubo más de 90 asistentes que ya mostraron su interés en formar parte de esta iniciativa.

Un PFC de segundo ciclo desarrollado en este proyecto fue presentado en septiembre de 2001 ([12]), y contiene una serie de agentes que ofrecen al usuario servicios de tipo hospitalario. Por ejemplo, el usuario puede pedirle a su agente personal que le diga el hospital más cercano en el que le pueda visitar un oftalmólogo, e incluso puede pedir al sistema que le reserve hora con él, o consultar su historial médico. La arquitectura del sistema se muestra en la figura 3.

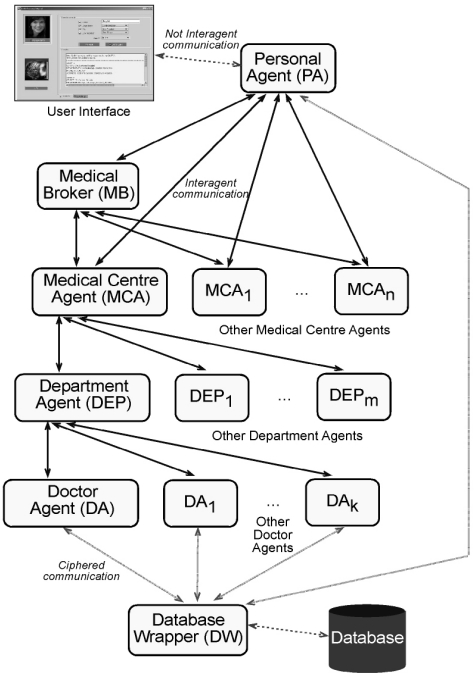


Figura 3. SMA que proporciona servicios médicos

En la misma línea, también se ha presentado este pasado septiembre un PFC en el que se ha diseñado e implementado un SMA que ofrece servicios relacionados con la restauración ([8]); el agente personal del usuario puede hacer consultas sobre los restaurantes de Tarragona en función de condiciones sobre un conjunto de atributos (tipo de comida, precio del menú, platos concretos, disponibilidad de aparcamiento, tarjetas de crédito aceptadas, etc.), y también puede gestionar la reserva de una mesa en un restaurante concreto.

6. Resultados y valoración de la experiencia docente

6.1 Resultados

En el curso 99-00 los 2 integrantes de GruSMA presentaron sus PFCs (de Ingeniería Técnica en Informática) en la convocatoria de septiembre ([18],[5]). Los dos proyectos estaban integrados en el dominio de la gestión del transplante de órganos. En el curso 00-01 hubo un incremento importante en el número de alumnos, y se empezaron 7 PFCs (5 de Ingeniería Técnica en Informática y 2 de segundo ciclo de Ingeniería Informática). De éstos, 4 ya fueron presentados en septiembre de 2001 (1 relacionado con el transplante de órganos [19], 2 relacionados con AgentCities [12], [8] y el último relativo a la simulación de equipos de trabajo utilizando SMAs [13]). Tres de estos PFCs corresponden a la Ingeniería Técnica, y uno al segundo ciclo. En el presente curso hay actualmente 10 alumnos trabajando en PFCs (3 en el dominio de los transplantes, 5 en diferentes ámbitos dentro del proyecto AgentCities - cines, información turística, búsqueda de trabajo, información médica a usuarios móviles - y 2 en la construcción de agentes que busquen información por Internet y la representen en una ontología dinámica).

6.2 Valoración de la experiencia docente

Algunos de los puntos positivos de la puesta en marcha de GruSMA son los siguientes:

- Constatar el interés que han mostrado los alumnos de Ingeniería Informática en realizar PFCs relacionados con la Inteligencia Artificial, y la creatividad desarrollada durante su diseño e implementación.
- Buen espíritu de trabajo en grupo durante las sesiones de trabajo.
- Los cinco alumnos que entregaron el PFC de la Ingeniería Técnica ahora están haciendo el segundo ciclo de Ingeniería Informática, y tienen la intención de aprovechar la formación y los conocimientos adquiridos en GruSMA para realizar el PFC del segundo ciclo en el mismo tema. Así, se ha establecido una cierta continuidad de los alumnos dentro del grupo.
- El alumno que ha hecho el PFC del segundo ciclo ha comenzado los estudios de doctorado dentro del programa de Inteligencia Artificial de la UPC, bajo la dirección de uno de los coordinadores de GruSMA.
- El trabajo realizado es de buena calidad. Una muestra de ello es la presentación de estos resultados en conferencias y revistas de carácter nacional e internacional (p.e. [17], [16], [20], [14]).
- Los coordinadores del grupo también han aprendido mucho de la experiencia de dirección de PFCs en SMAs (qué problemas se encuentran en el diseño e implementación de Sistemas Multi-Agente, cómo se solucionan, cuáles son las herramientas que proporciona JADE, etc.). Este conocimiento nos lleva a sugerir la posibilidad de tener una asignatura optativa de segundo ciclo centrada en el área de los sistemas multi-agente.

Entre los aspectos negativos podemos indicar los siguientes:

- Será difícil unir diferentes SMAs en uno sólo, como teníamos previsto en un principio, ya que las especificaciones de la FIPA (y las diferentes versiones de JADE) van evolucionando bastante deprisa (p.e. los PFCs desarrollados en la versión 1.3 de JADE no pueden ejecutarse directamente en la versión 2.01, que tampoco es compatible con la versión actual, la 2.5).
- El grupo de trabajo no dispone de excesivos recursos propios (ordenadores, impresoras). Así, sólo se pueden utilizar recursos del departamento de Ingeniería Informática y Matemáticas o de la Escuela Técnica Superior de Ingeniería. Esto limita en cierta forma la actividad del grupo.

En resumen, la valoración global de la puesta en marcha de GruSMA es totalmente positiva, y animamos firmemente a profesores interesados en temas de Inteligencia Artificial a explorar alternativas similares en otras universidades.

Referencias

[1] AgentCities. Información disponible en *http://www.agentcities.org.*.

[2] Arantza Aldea, Beatriz López, Antonio Moreno, David Riaño, Aïda Valls. *A Multi-Agent System for Organ Trasplant Co-ordination*. En *Artificial Intelligence in Medicine* (Eds: S.Quaglini, P.Barahona, S.Andreassen). Lecture Notes in Computer Science 2101, pp.413-416, Springer Verlag.

[3] Banzai: grupo de investigación en Inteligencia Artificial de la Univ. Rovira i Virgili (*http://www.etse.urv.es/recerca/banzai*).

[4] Fabio Bellifemine et al. *Developing multi-agent systems with a FIPA compliant agent framework*. Software Practice and Experience, No. 31, pp. 103-128, 2001.

[5] Jaime Bocio. *Diseño e implementación de un SMA para la gestión de equipos médicos*

para transplantes. PFC, Ing. Técnica en Informática de Sistemas, URV, Septiembre 2000.

[6] Tim Finin et al., *Specification of KQML Communication Language*, DARPA Knowledge Sharing Effort, 1993 (http://www.cs.umbc.edu).

[7] FIPA: Foundation for Intelligent Physical Agents (ver *http://www.fipa.org*).

[8] Albert Gavarró. *Agents que proporcionen serveis de restauració als visitants de Tarragona dins el projecte AgentCities*. PFC, Ing. Técnica en Informática de Sistemas, URV, Septiembre 2001.

[9] GruSMA: Grupo de trabajo en Sistemas Multi-Agente (información disponible en *http://www.etse.urv.es/recerca/banzai/toni/MAS*).

[10] David Isern. *Avaluació d'entorns de desenvolupament de Sistemes Multi-agent*. PFC, Ing. Técnica en Informática de Sistemas, URV, Septiembre 1999.

[11] JADE: Java Agent Development Framework (información disponible en *http://sharon.cselt.it/projects/jade*).

[12] David Isern. *Disseny i implementació d'un Sistema Multi-Agent per proporcionar serveis mèdics dins del projecte AgentCities*. PFC, segundo ciclo de Ing. en Informática, URV, Septiembre 2001.

[13] Marta Marín. *Un Sistema Multi-Agent d'ajuda a la composició de grups de treball per a tasques complexes*. PFC, Ing. Técnica en Informática de Sistemas, URV, Septiembre 2001.

[14] Antonio Moreno, David Isern. *A first step towards proving health-care agent-based services to mobile users*. Póster aceptado en la 1st International Joint Conference on Autonomous Agents and Multi-Agent Systems, AAMAS-2002. Bologna, Italia, julio de 2002.

[15] Antonio Moreno, Aïda Valls. *GruSMA: Grupo de Trabajo en Sistemas Multi-Agente*. Encuentro sobre docencia en Inteligencia Artificial, en el XI Congreso de la Asociación española para la Inteligencia Artificial, CAEPIA-2001. Gijón, noviembre de 2001.

[16] Antonio Moreno, Aïda Valls, Jaime Bocio. *A Multi-Agent System to Schedule Organ Transplant Operations*. *Inteligencia Artificial* (Revista Iberoamericana de Inteligencia Artificial) 13 , pp. 36-44, 2001.

[17] Antonio Moreno, Aïda Valls, Alberto Ribes. *Finding efficient organ transport routes using multi-agent systems*. 3rd International Workshop on Enterprise Networking and Computing in Healthcare Industry, HealthCom'2001. L'Aquila, Italia, 29 junio- 1 julio 2001.

[18] Alberto Ribes. *SMA d'ajuda a la gestió dels transplants d'òrgans entre hospitals: logística del transport d'òrgans*. PFC, Ing. Técnica en Informática de Sistemas, URV, Septiembre 2000.

[19] David Sánchez. *Un Sistema Multi-Agent d'ajuda a l'assignació d'òrgans en trasplantaments. El mètode ClusDM amb criteris lingüístics*. PFC, Ing. Técnica en Informática de Sistemas, URV, Septiembre 2001.

[20] Aïda Valls, Antonio Moreno, David Sánchez. *A multi-criteria decision aid agent applied to the selection of the best receiver in a transplant*. Actas de la 4th International Conference on Enterprise Information Systems, ICEIS-2002, pp 431-438. Ciudad Real, abril de 2002.

Acerca de la investigación

Javier Oliver
Dept. de Ingeniería del Software
Facultad de Ingeniería
Universidad de Deusto
48007 Bilbao
e-mail: oliver@eside.deusto.es

Resumen

Es opinión generalizada que el profesor universitario debe combinar docencia e investigación, pero ¿porqué no se realiza en la universidad española más investigación y de mayor calidad? Algunos de los motivos fueron ya expuestos en el siglo XIX por uno de nuestros más insignes investigadores: D. Santiago Ramón y Cajal. Lo que sigue es una reflexión crítica acerca de la investigación, basada en gran parte en las recomendaciones de D. Santiago, que no han perdido un ápice de su vigencia a pesar del tiempo transcurrido.

1. Breve semblanza de D. Santiago

Santiago Ramón y Cajal nación en Petilla de Aragón, provincia de Navarra, en 1852, aunque él siempre consideró Ayerbe, en Huesca, como su patria chica [1]. Tras una infancia difícil, en la que llegó a cumplir un arresto de cuatro días, y a trabajar de aprendiz de barbero y zapatero, descubrió al fin el gusto por el estudio, y cursó la carrera de Medicina en la Facultad de Zaragoza, licenciándose a los 21 años.

Fue profesor de anatomía, histología y anatomía patológica. Investigó acerca de la estructura del sistema nervioso, mejorando los métodos de tinción celular utilizados hasta entonces. En 1897 fue nombrado miembro de la Real Academia de Ciencias Exactas, Físicas y Naturales y su discurso de aceptación fue publicado bajo el título de "Reglas y consejos sobre la investigación biológica" [2], obra en la que se inspira este trabajo, y en la que se hacen abundantes consideraciones sobre la investigación científica, la mayoría de las cuales siguen siendo aplicables más de 100 años después de haber sido escritas.

En 1906 compartió con el científico italiano Camilo Golgi el premio Nobel de Medicina, por sus estudios sobre la estructura y composición del sistema nervioso.

2. Ataduras del investigador novel

Al comenzar la carrera investigadora el principiante se enfrenta a una serie de obstáculos:

2.1. Idolatría a los clásicos

Aunque el respeto excesivo a los trabajos de sus antecesores se debe en general a una mezcla de humildad y reconocimiento legítimo a la obra de otros científicos, puede llegar a ser un lastre insuperable, y a anular toda iniciativa y espíritu crítico. Dice D. Santiago:

> Defecto por defecto, preferible es la arrogancia al apocamiento: la osadía mide sus fuerzas y vence ó es vencida; pero la modestia excesiva huye de la batalla y se condena á vergonzosa inacción.

Siempre se debe mantener el espíritu crítico alerta. La lectura de cualquier autor, por famoso que sea, nos debe hacer cuestionar todo aquello sobre lo que exista alguna duda razonable.

2.2 Agotamiento de la ciencia

Se trata del falso convencimiento de que ya no queda nada por descubrir. Si bien es cierto que hay épocas en la historia de la ciencia en las que la coincidencia de factores propiciadores hace que tengan lugar numerosos avances científicos, en cualquier otro momento, quedan siempre y en todas las disciplinas pequeñas y grandes cuestiones que resolver.

Además, el calificar un problema científico de *grande* o *pequeño*, es algo muy relativo. A menudo, una observación aparentemente trivial se convierte con el tiempo en un concepto fundamental. Como nos recuerda D. Santiago:

> Perdido en un indigesto Tratado de Teología, *Christianismi Restitutio* [2], escribió Servet, como al desdén, tres líneas tocante á la circulación pulmonar, las cuales constituyen hoy su principal timbre de gloria.

Es recomendable comenzar la andadura científica abordando problemas más accesibles, dejando para más adelante los más complejos.

2.3. Exaltación de la investigación práctica

A menudo, el investigador principiante sobrevalora la investigación aplicada frente a la básica sin darse cuenta de que los descubrimientos científicos surgen frecuentemente como curiosidades sin utilidad inmediata y es la combinación de varias de esas *curiosidades* la que finalmente acaba teniendo aplicaciones industriales.

La descripción inicial de la cámara oscura apenas tuvo repercusión, y la posibilidad de obtener imágenes fotográficas sobre un papel bañado en nitrato argéntico no despertó apenas interés, porque la imagen no podía ser fijada. Pero en 1839, Daguerre sentó las bases de la fotografía actual basándose en éstas y otras *curiosidades científicas*. De nuevo nos habla D. Santiago:

> Poco importa que una verdad científica sea aprovechada por nuestros hijos o por nuestros nietos. Medrada andaría la causa del progreso si Galvani, si Volta, si Faraday, descubridores de los hechos fundamentales de la ciencia de la electricidad, hubieran menospreciado sus hallazgos por carecer entonces de aplicación industrial.

Esta cuestión está hoy día de rabiosa actualidad. Resulta muy difícil conseguir financiación para una investigación sin demostrar claramente *para qué sirve*. En ocasiones no hay que dejarse guiar solamente por la visión a corto plazo, y pensar que con el tiempo, la investigación básica puede llegar a ser rentable.

2.4. Insuficiente autoestima

Aunque es necesaria una cierta capacidad intelectual para la investigación científica, no es necesario tener una inteligencia superdotada, pero sí cualidades como la paciencia, la minuciosidad o la tenacidad. Éstos rasgos del carácter se pueden potenciar con el trabajo y verse reforzados con la consecución de algún éxito científico. Hoy en día, la investigación es una labor de equipo, y hace falta contar con gente brillante y con gente metódica, con creativos y con minuciosos...

Gran parte de las capacidades intelectuales y rasgos del carácter pueden tener cabida en un equipo de investigación. Lo que se considera genio, no es muchas veces más que una mayor velocidad en la concepción de las ideas y en la realización de los planes. Los entendimientos rápidos pueden ser necesarios en algunos campos como en la oratoria y el debate político, pero en la actividad científica, como en el arte, los resultados se juzgan por su valía, y no por el tiempo empleado en producirlos.

Por otro lado, la concentración del esfuerzo puede suplir una cierta limitación en la capacidad. La afirmación de que *el saber no ocupa lugar*, es muy discutible ya que los recursos son limitados, y el saber exige al menos de un cierto tiempo para ser adquirido. Opina D. Santiago que

> La lista de los aptos para la labor científica es mucho más larga de lo que se cree, y se compone, no sólo de los talentos estudiosos, de los fáciles, de los grandes ingenios codiciosos de reputación y ansiosos de enlazar su nombre á una obra grande, sino también de esos entendimientos modestos, conocidos con el dictado de *mañosos*, por la habilidad y tino con que realizan toda obra manual; de esos otros dotados de temperamento artístico y que sienten con vehemencia la belleza de las obras de la naturaleza; en fin, de los meramente curiosos, flemáticos, cachazudos, devotos de la religión de lo menudo y capaces de consagrar largas horas al examen del más insignificante fenómeno natural.

Si se tiene vocación, unas aptitudes entro de lo normal, y la suficiente perseverancia, es posible dedicarse a la carrera científica.

3. Enfermedades de la voluntad

Dice D. Santiago, con cierta dureza, que aquellos científicos que teniendo la capacidad, la formación y los medios para investigar, no llevan a cabo obra significativa alguna, son *enfermos de la voluntad*:

> Todos hemos visto profesores superiormente dotados, desbordantes de actividad y de cultura, en posesión de suficientes medios de trabajo, y que sin embargo no realizan obra personal ni escriben casi nunca. [...] dichos maestros son enfermos de la voluntad [...] sus discípulos y amigos tienen el derecho de considerarlos como anormales, y de proponerles, con el respeto y dulzura debidos, un tratamiento espiritual adecuado.

Llevado por su afán taxonómico, D. Santiago clasifica a los enfermos de la voluntad en los siguientes grupos:

3.1. Contempladores o diletantes

Les encanta admirar la estética de la naturaleza y con un culto casi fetichista, acumulan, dibujan, fotografían y examinan sus especímenes pero sin añadir nunca nada nuevo al conocimiento sobre los mismos:

> Todos nuestros lectores recordarán tipos y variedades interesantes de esta especie, tan simpática por su entusiasmo juvenil y verbo cálido y cautivador, como estéril para el progreso efectivo de la ciencia.

3.2. Bibliófilos y políglotas

La erudición no tiene en sí misma mucho valor si no es la antesala de una obra personal fecunda, y desde el punto de vista del avance de la ciencia, tiene más interés un nuevo libro para la biblioteca, que llevarla toda entera en la cabeza. Dice D. Santiago que:

> Los síntomas de esta dolencia son: tendencias enciclopedistas; dominio de muchos idiomas, algunos totalmente inútiles, abono exclusivo á Revistas poco conocidas; acaparamiento de cuantos libros novísimos aparecen en los escaparates de los libreros; lectura asidua de lo que importa saber, pero sobre todo de lo que interesa á muy pocos; pereza invencible para escribir y desvío del seminario y del laboratorio.

3.3. Megalófilos

Esta variedad morbosa se reconoce fácilmente por su afán de dar desde un principio con la piedra filosofal, con el gran descubrimiento que cambie el curso de la historia. No se dan cuenta de que hay que empezar por el principio, y poco a poco, marcarse objetivos más ambiciosos, a medida que aumenta la experiencia y los medios disponibles. D. Santiago los describe diciendo que:

> ... como por vía de milagro esperan estrenarse con una hazaña prodigiosa. Recordando acaso que Herz, Mayer, Schwann, Röntgen, Curie, iniciaron su vida científica con un gran descubrimiento, aspiran á ascender, desde el primer combate, de soldados á generales...

3.4. Organófilos

Son aquellos obsesionados con los útiles de trabajo. El medio se convierte en el fin, y se dedican a atesorar todo tipo de instrumentos, accesorios, bibliografía, pero sin utilizarlos nunca, no vaya a ser que se estropeen, se pierdan o se desgasten. Todos los materiales se guardan bajo llave, fuera del alcance de investigadores fogosos que pudieran hacer un mal uso de los mismos. Se lamenta D. Santiago de que:

> De los organófilos empedernidos no puede sacarse partido. Es enfermedad casi incurable, sobre todo si va asociada, como ocurre con frecuencia, á cierto estado moral poco confesable: á la preocupación egoísta y antipática de impedir que otros trabajen ya que ellos no saben o no quieren trabajar.

3.5.Descentrados

Son aquellos que han equivocado su vocación, y que dedican la mayor parte de su tiempo y sus esfuerzos a tareas que nada tienen que ver con su puesto en la sociedad. El trabajo lo ejercen siguiendo la ley del mínimo esfuerzo, sin ilusión y con el único fin de cobrar el sueldo consiguiendo de paso que otra persona más entregada no lo desempeñe. D. Santiago habla de generales nacidos para pacíficos burócratas, profesores de medicina que cultivan la literatura o la arqueología, ingenieros escribiendo melodramas... Los descentrados de cierta edad son casos irrecuperables pero los jóvenes, harían bien en cambiar a tiempo el rumbo

de su carrera, armonizando mejor su labor diaria y las inquietudes de su espíritu.

3.6. Teorizantes

Son individuos capaces, con iniciativa e imaginación pero con un marcado rechazo a los hechos concretos y al trabajo de laboratorio. Quieren tener visiones globales de los temas, y prefieren el libro al artículo especializado. Siempre se decantan por una teoría innovadora aunque inconsistente, frente a un modelo clásico pero sólido y comprobado.

Ante una cuestión científica sin resolver tienden siempre a elaborar una nueva teoría en lugar de aplicar el método experimental. Aunque las teorías en forma de hipótesis de trabajo son imprescindibles para el avance científico, el abuso de la tendencia teorizante no consigue hacer avanzar la ciencia. Las teorías, por sólidas que parezcan inicialmente, suelen ser periódicamente sustituidas por otras teorías. Son los hechos y las observaciones concretas, una vez sistematizados y correctamente interpretados los que hacen avanzar la ciencia. D. Santiago no perdona cuando afirma que:

> En el fondo, el teorizante es un perezoso disfrazado de diligente. Sin percatarse de ello, obedece a la ley del mínimo esfuerzo. Porque es más fácil forjar una teoría que descubrir un fenómeno.

4. Investigación y docencia

Por lúcidas y acertadas que sean las observaciones de D. Santiago, ¿qué aplicación tienen hoy día en nuestra universidad? ¿Podemos compatibilizar docencia universitaria e investigación en informática? De entre las carencias que a menudo se citan para justificar una pobre actividad investigadora destacan la falta de tiempo, la falta de medios, la falta de apoyo institucional... Pero, ¿qué es en realidad lo que necesitamos para investigar?

Indudablemente, hacen falta *medios* materiales, pero creo que éste es el factor menos decisivo. Muchos de los ilustres científicos que nos han precedido son ejemplos del principio de *economía de medios*: sacar el máximo partido a un mínimo de recursos.

Más importante es el *método*, el saber cómo hacer las cosas. Tener una buena metodología de trabajo exige una formación general (método experimental, estadística, técnicas de búsqueda de información...) y específica (la propia de cada área de conocimiento) que nunca se termina de adquirir: es una formación continua. En la universidad tenemos una posición de privilegio, que deberíamos aprovechar mejor, para disfrutar de una formación permanente.

La complejidad de algunos proyectos de investigación y las condiciones que para su financiación imponen determinadas instituciones exigen la cooperación de numerosos investigadores de distintos grupos y países. Hay que alcanzar una *masa crítica*. Aunque la comunidad académica ha fomentado tradicionalmente los contactos e intercambios entre sus miembros, es fundamental que mejoremos nuestras capacidades de comunicación, organización y gestión, encontrando de verdad las sinergias que el trabajo en grupo posibilita.

Pero en mi opinión, el elemento fundamental de la investigación es la *motivación*. Un profesor universitario puede tener objetivos muy diversos cuando investiga y publica: estabilidad laboral, mejora del salario, hacer curriculum, reconocimiento social, ascenso profesional... Aunque éstas y otras sean metas lícitas, creo que la motivación principal para investigar ha de ser la curiosidad por descubrir el porqué de las cosas, y para publicar, el afán lógico de contar a los demás lo que se ha descubierto.

5. Conclusión

La descripción de estas ataduras y enfermedades de la voluntad de los investigadores, dictadas con ironía y clarividencia por D. Santiago desde su tumba, pretende ser un espejo, a veces descarnado, en el que nos miremos y hagamos examen de conciencia los docentes universitarios.

En todo caso pueden servir como base para una reflexión crítica en la que cada uno intente responderse a la pregunta: ¿porqué no hago más investigación y de mayor calidad? Hoy como ayer si detectamos alguna de estas patologías en nuestro ánimo, quizá debamos recetarnos una estancia en algún centro de investigación de auténtico prestigio:

> El laboratorio del sabio es un sanatorio incomparable para los extravíos de la atención y los des-

> mayos de la voluntad. En él se desvanecen viejos prejuicios y se contraen sublimes contagios. Allí, al lado de un sabio laborioso y genial, recibirá nuestro abúlico el bautismo de sangre de la investigación; allí contemplará con envidia una ardiente emulación por arrancar secretos á lo desconocido; allí respirará el desdén sistemático hacia las vanas teorías y los discursos retóricos; allí, en fin –en extrañas tierras- sentirá renacer el santo patriotismo. Y, cuando lanzado en el camino del trabajo personal cuente en su haber algunos estimables descubrimientos, de regreso al país natal, medirá mejor sus admiraciones y mirará con desdén, casi con lástima, á sus antiguos ídolos.

Quizá debamos volver a la motivación genuina que toda investigación debe tener, que no es otra que la curiosidad infantil de querer averiguar el porqué de las cosas. Si lo descubrimos, lógicamente se lo queremos contar a aquellos que están interesados en los mismos asuntos que nosotros, y éste debe ser el motivo de toda publicación. Si nos dejamos guiar por otros intereses más mundanos, nos estaremos alejando de la esencia de la investigación.

Referencias

[1] Enciclopedia Espasa, Tomo 49, p. 568, Editorial Espasa Calpe S.A., 1923.

[2] Santiago Ramón y Cajal. *Reglas y consejos sobre investigación biológica.* Imprenta y librería de Nicolás Moya, 3ª Edición, 1913.

[3] Miguel Servet. *Christianismi Restitutio*. Fundación Universitaria Española, 1980.

Demostraciones

Un conjunto de herramientas didácticas sencillas para un curso introductorio sobre modelado y evaluación de computadores

Xavier Molero, Vicente Santonja
Departament d'Informàtica de Sistemes i Computadors
Universitat Politécnica de València
Camí de Vera, s/n. 46022 València
e-mail: {xmolero|visan}@disca.upv.es

Resumen

El presente artículo describe de manera concisa un conjunto de pequeñas herramientas didácticas implementadas mediante programas de fácil uso. Estos programas tratan sobre diversos aspectos de carácter básico en materias sobre modelado y evaluación de sistemas informáticos. El principal objetivo perseguido con el diseño de las herramientas expuestas en este trabajo es que sean fácilmente utilizables por los alumnos que cursan asignaturas relacionadas con esta temática.

1. Introducción

La evaluación de prestaciones o rendimiento (*performance*) de los computadores tiene una gran influencia en el diseño, desarrollo, configuración y sintonización de los sistemas informáticos. El marco que actualmente ofrece el plan de estudios de nuestra Escuela Universitaria de Informática para el tratamiento de la temática relacionada con la evaluación de prestaciones viene representado por la asignatura Evaluación de Sistemas Informáticos (ESI).

La asignatura ESI tiene carácter optativo y se imparte en el primer cuatrimestre del tercer curso. La distribución de créditos es de 3 para las sesiones de teoría y 3 para las de laboratorio, lo que le confiere un aspecto muy práctico. La parte teórica se trata en cuatro temas:

- Introducción a la evaluación del rendimiento
- Monitorización de sistemas informáticos
- Comparación del rendimiento de sistemas: referenciación (*benchmarking*)
- Técnicas analíticas: análisis operacional. Detección de cuellos de botella

Las sesiones de laboratorio inciden en los aspectos más prácticos de la parte teórica: análisis de la configuración de un computador mediante el programa Sandra [6], herramientas básicas de monitorización de sistemas Unix [4,7], resumen y comparación de rendimientos [2,3], diseño de programas de prueba (*benchmarks*) [2,4] y modelado mediante redes de colas de espera [1,2] con el lenguaje de especificación de modelos QNAP2 [5].

2. La necesidad de herramientas prácticas

Dentro del apartado práctico expuesto en el punto anterior, y tras una dilatada experiencia en la impartición de la asignatura ESI, se ha puesto de manifiesto la gran utilidad de disponer de un conjunto reducido de pequeñas herramientas con las cuales el alumno pueda aplicar algunas de las técnicas más usuales que aparecen en el campo de la evaluación de prestaciones.

El hecho más importante no es que estas herramientas tengan una interfaz visual destacada, sino que lo realmente interesante es poder resolver pequeños problemas mediante herramientas diseñadas *ad hoc* y que sean fácilmente portables de una plataforma a otra. Ello ha hecho que se escoja C (en su versión ANSI) como lenguaje de implementación, y que los parámetros se especifiquen en la línea de órdenes del sistema operativo. Por otro lado, se ha incluido una

pequeña ayuda de manejo de cada herramienta, la cual se puede obtener sin más que ejecutar el programa sin ningún parámetro.

3. Breve descripción de las herramientas

A continuación se presenta brevemente un conjunto de cuatro programas de manejo muy sencillo para su aplicación en algunos de los temas básicos de la evaluación de los sistemas informáticos.

Herramienta `amdahl`. Calcula, empleando la ley de Amdahl [3], la aceleración global de un sistema (*speedup*) después de sustituir un componente del mismo por uno *k* veces más rápido, el cual se utiliza durante una fracción de tiempo *f*. Esta aceleración se calcula mediante la fórmula:

$$A = \frac{1}{1 - f + \frac{f}{k}}$$

Asimismo, también se calcula, utilizando la misma expresión, el aumento del coste del sistema a partir del coste total *C* del mismo, el del componente que se reemplaza *Cr* y el del componente nuevo *Cn*. La Figura 1 muestra un ejemplo sencillo de uso donde se puede ver el valor que toma cada parámetro de entrada al programa.

```
amdahl   f  k       C     Cr     Cn

amdahl 0.8  3  150000 56000 90000

      Aceleración del sistema: 2.14
      Incremento del coste:    1.23
```

Figura 1: Ejemplo de uso del programa `amdahl`

Herramienta `zerotest`. Esta herramienta resulta muy útil para comparar el rendimiento de dos computadores [2,3]. Dados un total de *n* programas de prueba, este programa analiza los tiempos de ejecución de dichos programas en dos computadores, y calcula la significación estadística de las diferencias observadas. Si el intervalo de confianza calculado con un nivel de confianza del 95% para estas diferencias incluye el cero entonces no hay una diferencia significativa de rendimientos. Un ejemplo de aplicación de esta herramienta para el caso de 4 programas de prueba se muestra en la Figura 2.

```
zerotest n    x1   x2   x3   x4   y1   y2   y3   y4

zerotest 4  23.3 11.8 14.8 87.2 26.9 14.1 13.7 98.6

 Media aritmética diferencias:      -4.05
 Desviacion típica diferencias:      5.29
 Intervalo de confianza diferencias: [-12.46,4.36]
 Análisis: no hay diferencias significativas
```

Figura 2: Ejemplo de uso del programa `zerotest`

Herramienta `merrill`. Esta herramienta calcula el índice de Merrill de un gráfico de Kiviatt [2]. La forma de este tipo de gráficos es la de un polígono donde la longitud de sus lados está determinado por el valor de 8 índices x_i de rendimiento, la mitad buenos y la otra mitad malos, dispuestos alternativamente. Los índices pueden representar la utilización de determinados componentes del sistema (como procesador, entrada/salida, red, etcétera). El índice de Merrill varía entre 0 y 100 y se calcula según la fórmula:

$$Q = \sqrt{\frac{1}{2n}\sum_{i=1}^{n}(x_{2i-1} + x_{2i+1})(100 - x_{2i})}$$

Cuanto mayor sea el valor de este índice *Q* mejor será el rendimiento del sistema informático, ya que el gráfico de Kiviatt asociado se parecerá más a una estrella. Para comparar el rendimiento de dos computadores basta comparar sus índices de Merrill teniendo en cuenta que sólo es significativa la parte entera de este índice. La Figura 3 muestra un ejemplo de resolución.

```
merrill  x1 x2 x3 x4 x5 x6 x7 x8

merrill 100 60 40  0 40  0 40 60

  Cuantificación de Merrill (0..100): 58.31
```

Figura 3: Ejemplo de uso del programa `merrill`

Herramienta `solvenet`. Esta herramienta es, con mucho, la más compleja de todas las presentadas hasta ahora. El objetivo fundamental es resolver modelos sencillos de redes de colas de espera [1,2] mediante la aplicación del algoritmo del valor medio o los algoritmos para redes de colas abiertas. La herramienta está pensada como alternativa al programa QNAP2 en los casos de redes más simples. Para facilitar la lectura de resultados se ha optado por utilizar también una interfaz similar a la empleada por QNAP2, además de aportar información adicional sobre cuellos de botella y asíntotas del rendimiento que resultan muy útiles para el análisis de las prestaciones del sistema. La Figura 4 muestra cómo se especifican los parámetros de entrada.

```
solvenet [0|1] [lambda| N Z] tcpu nio Sio1 Vio1...Sion Vion

  red:     0 (abierta) 1 (cerrada)
  lambda:  tasa de llegada(sólo para redes abiertas)
  N:       número de trabajos (sólo para redes cerradas)
  Z:       tiempo de reflexión(sólo para redes cerradas)
  tcpu:    tiempo de servicio de la CPU
  nio:     número de dispositivos de I/O
  Sio:     tiempo de servicio del dispositivo de I/O nº i
  Vio:     razón de visita o probabilidad de encaminamiento
           del dispositivo de I/O nº i
```

Figura 4: Parámetros de entrada al programa `solvenet`

Para ilustrar el funcionamiento de este programa, se ha considerado la red de colas mostrada en la Figura 5. En esta figura también se especifican los valores de los tiempos de servicio S_i de las estaciones así como la razón de visita V_i.

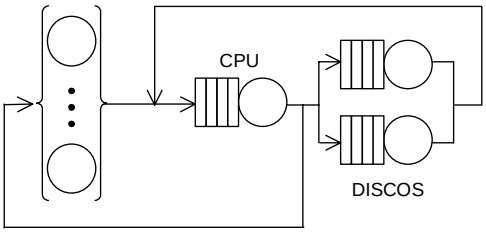


Figura 5: Ejemplo de red de colas cerrada

La información aportada por el programa `solvenet` se refleja en la Figura 6 de la página siguiente. Como se puede apreciar, el programa aporta tanto información referida a cada estación de servicio del modelo como del sistema en su conjunto. La información de cada estación, expresada en valores medios, incluye la utilización, el número de clientes en toda la estación así como en la cola de espera, el tiempo de respuesta, la productividad y la demanda de servicio.

En cuanto a la información referida al sistema informático completo, el programa calcula, a diferencia del programa QNAP2 que sólo da información por estación de servicio, el número de trabajos en el sistema central y en reflexión, el tiempo de respuesta y su valor mínimo, la productividad y su valor máximo. Por otro lado, también indica los límites asintóticos optimistas de la productividad y del tiempo de respuesta del sistema en función del número de trabajos.

4. Conclusión

En este trabajo se ha presentado un conjunto de herramientas sencillas diseñadas *ad hoc* y destinadas fundamentalmente a ser utilizadas en un curso básico de modelado y evaluación de computadores.

El hecho de estar diseñadas en lenguaje C y disponer de parámetros indicados en la línea de órdenes permiten que los alumnos puedan utilizarlas fácilmente. En la actualidad se está trabajando en otra serie de herramientas adicionales para tratar aspectos relacionados con la monitorización de sistemas Unix.

Referencias

[1] J. Cady and B. Howarth. *Computer systems performance, management and capacity planning.* Prentice may, 1990.

[2] R. Jain. *The art of computer system performance analysis. Techniques for experimental design, measurement, simulation and modeling*. John Wiley & Sons, 1991.

[3] D.J. Lilja. *Measuring computer performance. A practitioner's guide.* Cambridge University Press, 2000.

[4] M. Loukides. *System performance tuning*. O'Reilly & Associates, Inc., 1992
[5] QNAP2. *Reference manual* (Tomes I and II). Simulog, 1996.
[6] Sandra: *The System ANalyser, Diagnostic and Reporting Assistant program*. Versión de uso gratuito disponible en la dirección web www.sisoftware.co.uk/sandra
[7] Varios autores. *Unix performance tuning*. R&D Books, 1997.

```
QUEUEING NETWORK: CLOSED
Mean Value Analysis Algorithm

*******************************************************
*   NAME    *   UTIL    *  CUST NB * RESPONSE *  THRUPUT *
*******************************************************
*           *           *          *          *         *
* CPU       *    0.7534*    2.1342*    0.1416*   15.0687*
*           *           *          *          *         *
* I/O  1    *    0.2917*    0.3996*    0.0411*    9.7217*
*           *           *          *          *         *
* I/O  2    *    0.2430*    0.3141*    0.0646*    4.8609*
*           *           *          *          *         *
*******************************************************

*******************************************************
*   NAME    *  VISIT    *  SERVICE *  DEMAND  *  CUST NQ *
*******************************************************
*           *           *          *          *         *
* CPU       *   31.0000*    0.0500*    1.5500*    1.3808*
*           *           *          *          *         *
* I/O  1    *   20.0000*    0.0300*    0.6000*    0.1079*
*           *           *          *          *         *
* I/O  2    *   10.0000*    0.0500*    0.5000*    0.0710*
*           *           *          *          *         *
*******************************************************

      ***********************************
      *         SYSTEM VARIABLES          *
      ***********************************
      *                         *         *
      * WORKING CUSTOMERS       *    2.8479*
      * THINKING CUSTOMERS      *   12.1521*
      * ALL CUSTOMERS           *        15*
      * SATURATION POINT        *        18*
      *                         *         *
      * RESPONSE TIME           *    5.8588*
      * MINIMUM RESPONSE TIME   *    2.6500*
      *                         *         *
      * THROUGHPUT              *    0.4861*
      * MAXIMUM THROUGHPUT      *    0.6452*
      *                         *         *
      ***********************************
```

Figura 6. Resultados generados por el programa `solvenet` para el caso de red de colas de la Figura 5.

Una aplicación didáctica para el diseño y simulación de redes de colas

Vicente Santonja, Xavier Molero, Miguel Caballer
Departament d'Informàtica de Sistemes i Computadors
Universitat Politècnica de València
Camí de Vera, s/n. 46022 València
e-mail: {visan | xmolero}@disca.upv.es

Resumen

El programa WinNet 3.0 es una herramienta didáctica para el diseño de redes de colas y su simulación. Permite diseñar de una forma sencilla y rápida redes de colas cerradas, abiertas y mixtas; monoclase o multiclase. La red diseñada puede ser analizada mediante simulación. También permite traducir el diseño gráfico al lenguaje de especificación de modelos QNAP2. De esta forma WinNet puede utilizarse como entrada gráfica para QNAP2, el cual implementa múltiples algoritmos de resolución analítica de redes de colas, así como potentes mecanismos de simulación.

1. Introducción

Las asignaturas relacionadas con la evaluación de prestaciones de los sistemas informáticos tienen en la teoría de colas una de sus herramientas fundamentales. El funcionamiento de diversos sistemas informáticos desde el punto de vista de sus prestaciones puede ser modelado adecuadamente mediante redes de colas abiertas, cerrada o mixtas [1]. Existen diversos algoritmos que permiten resolver analíticamente estos modelos [2]. Es más, cuando la red no tiene solución analítica, esta técnica de modelado puede servir de base para la construcción de modelos de simulación. Sin embargo, la especificación de una red de colas es un proceso complejo, ya que deben definirse múltiples variables de entrada: caracterización del proceso de llegadas, distribución de los tiempos de servicio, probabilidades de encaminamiento, políticas de planificación de las colas, etc. Para facilitar esta parte del proceso de modelado se ha desarrollado la herramienta WinNet que funciona bajo el entorno Windows. Mediante una sencilla interfaz gráfica, el usuario dibuja la red de colas y, posteriormente, de forma asistida, va introduciendo los parámetros de cada una de las estaciones. A partir de este diseño, WinNet permite obtener una descripción del modelo en el formato QNAP2 [3]. Posteriormente puede utilizarse QNAP2 para la resolución analítica o la simulación de la red de colas. Con el fin de hacer de WinNet una herramienta más útil desde el punto de vista educativo, el programa incorpora un simulador con el cual se puede obtener índices de prestaciones de una forma rápida. De esta forma los alumnos pueden evaluar el impacto sobre la prestaciones de diversas alternativas de diseño o ajustes sobre la configuración de un sistema.

2. Descripción del programa

La Figura 1 muestra las partes principales que componen el entorno gráfico de WinNet 3.0

Mediante la barra de herramientas se accede fácilmente a las tareas más comunes de la aplicación: crear un nuevo documento, imprimir el documento actual, dibujar una estación de servicio, interconectar dos estaciones, aumentar o disminuir el nivel de zoom, etc.

La barra de estado aparece en la parte inferior de la ventana de WinNet. En ella se describen las acciones que realizan los elementos de menú mientras se usan las teclas de flecha para desplazarse por los menús.

El área de dibujo es donde se representa gráficamente la red de colas. WinNet es una aplicación multidocumento, es decir, permite tener abiertos

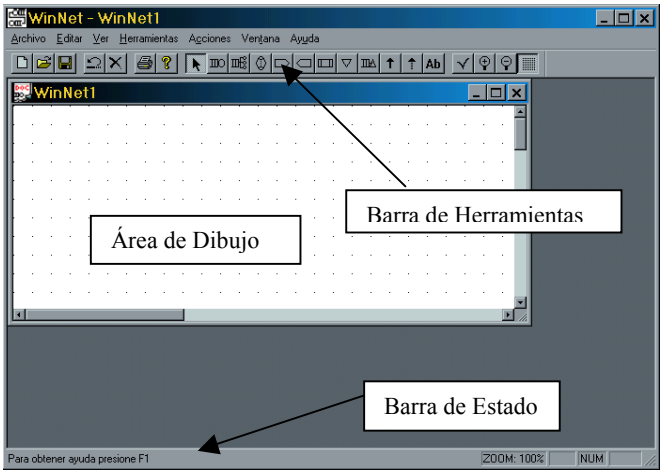


Figura 1: Aspecto general de la aplicación Winnet

varios documentos a la vez, y trabajar con ellos de forma independiente.

3. Tipos de estaciones

La red de colas estará formada por un conjunto de estaciones elegidas de entre los tipos siguientes:

- *Servidor simple*: Es una estación de servicio con un único servidor. Las estaciones de este tipo requieren diversos parámetros. A continuación se mencionan los más importantes. La política de planificación indica cómo se elige el siguiente cliente que entrará en el servidor de entre los que están esperando en cola, cuando el servidor queda libre. Puede ser una de las siguientes: FIFO, LIFO, PS, QUANTUM o PRIOR. Para cada clase de trabajos que atiende esta estación se especifica el número inicial de clientes en la estación y la distribución del tiempo de servicio. La aplicación ofrece la posibilidad de elegir entre seis distribuciones diferentes: exponencial, hiperexponencial, Erlang, determinista, uniforme entera y uniforme real. Dependiendo de la distribución elegida, la definición del tiempo de servicio deberá completarse especificando uno o dos parámetros adicionales (por ejemplo, valor medio y coeficiente de variación). Para cada clase se debe indicar, además, las probabilidades de encaminamiento que indican las rutas que siguen los clientes dentro de la red. Para que, en todo momento, la información de encaminamiento sea compatible con la topología de la red dibujada, sólo se permitirá utilizar como estaciones destino aquéllas con las que la estación actual está enlazada.
- *Servidor múltiple*. Modela un recurso del cual existen múltiples copias idénticas. Requiere los mismos parámetros que el anterior más el número de servidores.
- *Servidor infinito*. Modela un recurso del cual existen suficientes copias para que nunca se formen colas de clientes en espera de utilizarlo, o bien, un recurso que por sus características, puede ser compartido simultáneamente por un número ilimitado de clientes. Este tipo de estaciones también recibe el nombre de *estación tipo retardo*. Requiere los mismos parámetros que un servidor simple, salvo que en este caso la política de planificación de la cola no es aplicable.
- *Fuente*. Modela un punto de entrada de clientes en la red de colas. Sólo hay que especificar la distribución del tiempo entre llegadas y las probabilidades de encaminamiento.
- *Sumidero*. Modela un punto de salida la red de colas.
- *Recurso*. Junto con los semáforos, permite modelar la compartición simultánea de recursos. También se usa para modelar la exclusión mutua. Debe indicarse el número de copias del recurso disponibles.
- *Semáforos*. Modelan la solicitud y liberación de los recursos modelados según la estructura anterior.

4. Diseño de una red sencilla

Como ejemplo del uso de WinNet, en esta sección mostraremos la secuencia de pasos a seguir para crear una red sencilla. Se trata de una red cerrada monoclase formada por cuatro estaciones: los terminales, la CPU, y dos discos. El resultado final que obtendremos es el que se muestra en la Figura 2.

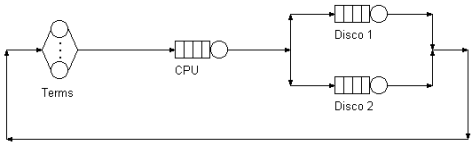


Figura 2. Red de colas ejemplo.

En primer lugar dibujaremos las estaciones. Empezamos por los terminales. Éstos se pueden modelar mediante una estación del tipo *Servidor Infinito*, por lo que debemos escoger ese tipo de estación en la barra de herramientas. Una vez elegido lo situamos en la zona del área de dibujo.

Seguidamente incorporamos al diseño el resto de estaciones. Al ser todas del tipo *Servidor Simple*, elegimos este tipo en la barra de herramientas y pinchamos tres veces en la zona de dibujo para situar correctamente las tres estaciones restantes. Una vez hemos añadido todas las estaciones, le asignamos un nombre a cada una de ellas, los cual ayudará a la hora de enlazarlas. Para ello simplemente hay que hacer doble clic sobre cada estación y rellenar el campo nombre de su ventana de parámetros. La red en este momento se encuentra así:

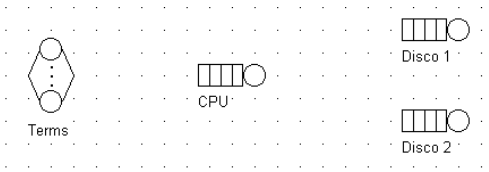


Figura 3. Colocación de las estaciones.

Ahora procedemos a crear los enlaces entre las estaciones. Para ello elegimos la opción *Enlace* de la barra de herramientas. Pinchamos en la estación de los terminales con el botón derecho del ratón, y después en la estación CPU, entonces aparecerá un enlace entre las dos estaciones. Después creamos los enlaces entre la CPU y los discos y, por último, cerramos la red uniendo los discos con los terminales. Para ello pinchamos en el Disco 1, en los puntos intermedios, y finalmente en los terminales. Para el Disco 2 no hace falta realizar todos los puntos intermedios, sólo tenemos que unir el Disco 2 con el primer punto en común con el enlace que viene del Disco 1. Automáticamente el enlace es finalizado por el programa hasta llegar a los terminales (ver Figura 4).

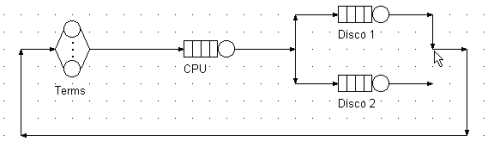


Figura 4. Enlaces entre estaciones.

Una vez realizados los enlaces ya podemos parametrizar las estaciones; para ello, hacemos doble clic sobre cada una de las estaciones y introducimos la información acerca de las distribuciones del tiempo de servicio, probabilidades de encaminamiento, políticas de planificación de la cola, etc.

La red de colas ya está diseñada. Para comprobar que no hayamos cometido ningún error vamos al menú *Acciones* y elegimos la opción *Chequear*. Si hemos seguido los pasos adecuadamente, aparecerá un mensaje indicando que la red es correcta. En caso contrario, aparecerá un listado con los errores cometidos.

Una vez revisada ya podemos obtener resultados, ya sea mediante la simulación o generando el archivo QNAP2 equivalente. Para generar el archivo QNAP2 debemos ir al menú *Acciones* y elegir la opción *Generar QNAP*, y elegir el archivo de destino. Con ese archivo podemos ejecutar la aplicación QNAP2 y generar los índices de prestaciones que deseemos. El archivo generado se muestra en la Figura 5.

WinNet también permite obtener resultados mediante simulación. Para ello, elegimos la opción *Simulación* del menú *Acciones*, introducimos el fichero de destino, y el tiempo de simulación. Una vez realizada la simulación aparece una ventana que muestra los índices de prestaciones

más importantes de las estaciones de la red (ver la Figura 6): utilización, longitud media de la cola, productividad, tiempo medio de espera en cola, tiempo medio de respuesta, etc.

El ejemplo que acabamos de presentar es intencionadamente sencillo. Se ha elegido para mostrar el proceso básico de modelado mediante WinNet. Sin embargo, la aplicación permite diseñar modelos mucho más complejos, que incluyan diversas clases de trabajos y redes mixtas (abiertas para unas clases y cerradas para otras).

```
/DECLARE/
    QUEUE Terms, CPU, Disco_1, Disco_2;

/STATION/
    NAME = Terms;
    TYPE = SERVER, INFINITE;
    INIT = 15;
    SERVICE = EXP(15.00);
    TRANSIT = CPU, 1.00;

/STATION/
    NAME = CPU;
    TYPE = SERVER, SINGLE;
    INIT = 0;
    SCHED = FIFO;
    SERVICE = EXP(0.50);
    TRANSIT = Disco_1, 0.70, Disco_2, 0.30;

/STATION/
    NAME = Disco_1;
    TYPE = SERVER, SINGLE;
    INIT = 0;
    SCHED = FIFO;
    SERVICE = EXP(2.00);
    TRANSIT = Terms, 1.00;

/STATION/
    NAME = Disco_2;
    TYPE = SERVER, SINGLE;
    INIT = 0;
    SCHED = FIFO;
    SERVICE = EXP(2.50);
    TRANSIT = Terms, 1.00;

/EXEC/
    SOLVE;

/END/
```

Figura 5. Programa QNAP2 generado automáticamente.

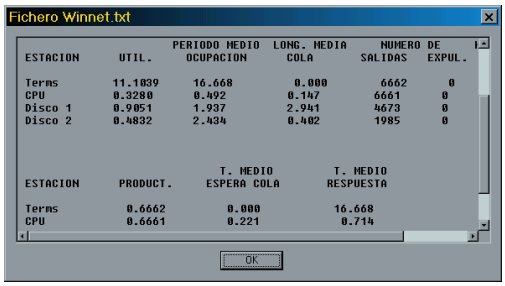


Figura 6. Resultados obtenidos por simulación.

5. Conclusión

La herramienta WinNet permite la descripción y simulación de redes de colas. Todo ello mediante un entorno gráfico y de fácil uso. Puede ser una herramienta interesante para las prácticas de asignaturas relacionadas con la evaluación de prestaciones de sistemas informáticos.

La incorporación de algoritmos de análisis dentro de la aplicación es uno de nuestros objetivos inmediatos.

Referencias

[1] G. Bolch, S. Greiner, H. de Meer y K. Trivedi. *Queueing Networks and Markov Chains*. John Wiley & Sons, 1998.

[2] F. Baskett, K. Chandy, R. Muntz y F. Palacios. Open, Closes, and Mixed Network of Quues with Different Classes of Customers. *Journal of the ACM*, 22(2):248-260, April 1975.

[3] Simulog. Disponible en http://www.simulog.fr

ECODE: Entorno Integrado de Desarrollo para CODE-2

Antonio Martínez Alberto Prieto Héctor Pomares Pedro Castillo

Departamento de Arquitectura y Tecnología de Computadores
E.T.S.I. Informática
Universidad de Granada
E-18071 Granada. e-mail: aprieto@ugr.es

Resumen

En este artículo, se describe el funcionamiento y las características más sobresalientes de ECODE, un entorno integrado de desarrollo (IDE) para el Computador Didáctico Elemental CODE-2. ECODE permite editar, ensamblar y depurar programas escritos en lenguaje ensamblador de CODE-2, al modo tradicional, además de emular la ejecución de programas en lenguaje máquina. El entorno puede trabajar en dos modos: modo real y modo ampliado, con el que se visualizan y monitorizan los contenidos de cualquier elemento interno del computador, pudiendo en cualquier instante interactuar con ellos, modificándolos directamente.

1. Introducción

CODE-2, es la segunda versión de un Computador Didáctico Elemental diseñado en el Departamento de Arquitectura y Tecnología de Computadores de la Universidad de Granada [1,2]. Se trata de un computador de tipo von Newmann, que contiene las unidades típicas de este modelo: entradas, salidas, unidad de control, unidad de procesamiento de datos y memoria para datos e instrucciones. La filosofía de concepción y diseño de este computador, es eminentemente pedagógica, y está completamente descrito a nivel de lenguaje máquina, lenguaje ensamblador y micromáquina en [2]. La longitud de palabra de CODE-2 es de 16 bits, y dispone de tan sólo 16 instrucciones máquina. Los elementos a los que se tiene acceso directamente desde el nivel de descripción de lenguaje máquina son:

- Un banco de 16 registros (r0 a rF).
- Unidad Aritmético Lógica (ALU), que realiza la operaciones que más adelante detallaremos. La ALU tiene asociados biestables indicadores, que funcionan de forma usual, de acuerdo con los resultados obtenidos en ella: biestable de cero (Z), de signo (S), de acarreo (C) y de desbordamiento (V).
- Memoria Principal: de 2^{16} =64Kpalabras de 16 bits (128 Kbytes), direccionable por palabras.
- Puertos de entrada y salida: Se admiten 256 puertos de entrada (IP00...IPFF) y otros 256 puertos de salida (OP00...OPFF).

El lenguaje máquina de CODE-2 se compone de 16 instrucciones, todas ellas de 16 bits, y son las siguientes:

- *Transferencia de datos*: Carga en un registro de un contenido de memoria (*LD*), almacenamiento del contenido de un registro en memoria (*ST*), carga inmediata en registro (byte alto, *LLI*, o byte bajo, *LHI*), entrada (*IN*) y salida (*OUT*).
- *Aritmético-Lógicas*: Suma (*ADDS*), resta (*SUBS*), NAND (*NAND*), y desplazamientos lógicos y aritméticos (*SHL*, *SHR*, *SHRA*).
- *Bifurcaciones* incondicionales y condicionales: salto (*B-*), llamada a subrutina (*CALL-*), retorno (*RET*) y
- *Parar* (*HALT*).

A continuación se presenta la primera versión de ECODE (Entorno CODE), un conjunto integrado de herramientas para desarrollar programas en lenguaje ensamblador (definido según el estándar IEEE 694 [4]) o máquina, depurarlos, y analizar con detalle el comportamiento dinámico de CODE-2 al ejecutar programas en código máquina.

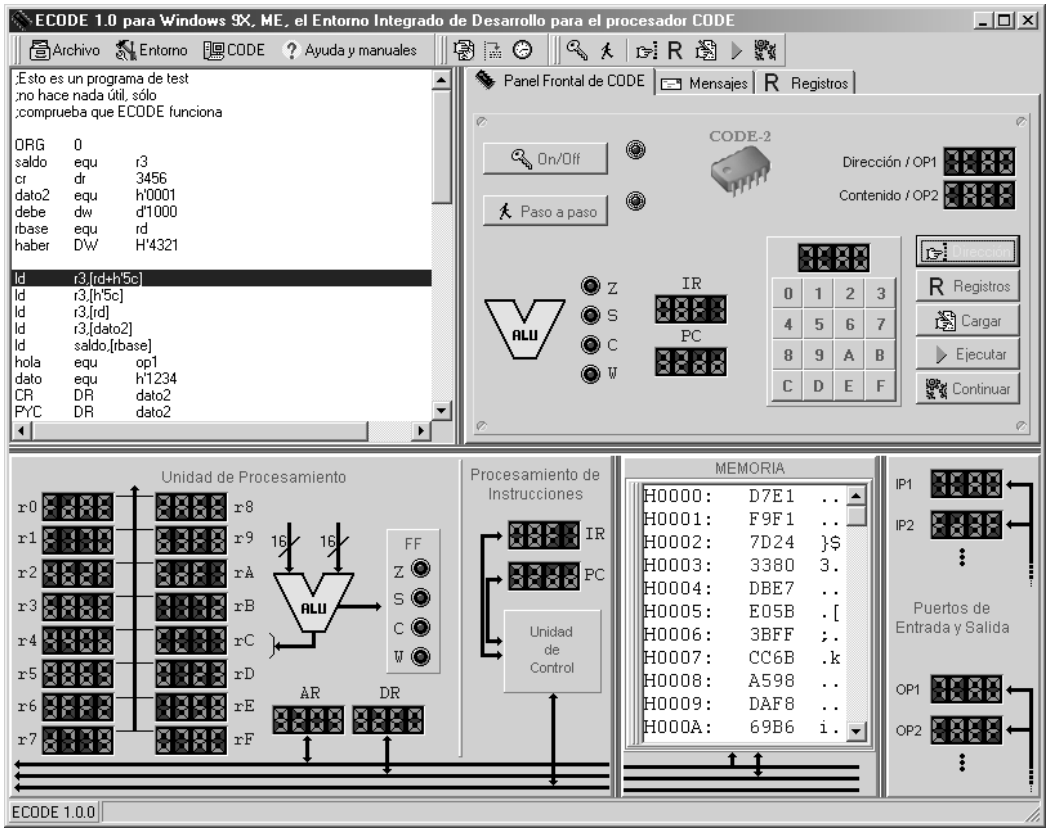


Figura 1. Imagen de la interfaz gráfica de ECODE.

2. Especificaciones de diseño de ECODE

ECODE es un Entorno Integrado de Desarrollo (IDE) para el computador CODE-2 que aúna diversas herramientas y programas diseñados con el propósito de conformar un entorno didáctico para el aprendizaje del funcionamiento de un computador básico.

A tal efecto, se definen en ECODE dos modos de funcionamiento, el modo real y el modo ampliado [3]:

- En el *modo real* se emula exactamente el comportamiento de CODE-2; es decir, el alumno sólo puede interactuar con la herramienta, de forma similar a como lo haría con un prototipo físico del computador. Concretamente, sólo se tiene acceso en este modo al panel frontal de CODE-2 descrito en [2] que dispone de teclado (puerto *IP1*) para introducción de información en hexadecimal, dos puertos de salida (*OP1*, *OP2*), teclado de órdenes (cargar, ejecutar, examinar/cargar registros, etc.), y visualizadores del contenido hexadecimal de los registros IR y PC y de los biestables indicadores.
- El *modo ampliado* incrementa notablemente las posibilidades del modo real. Se implementa por medio de utilidades y programas que facilitan el acceso directo a cualquier componente de CODE-2: posiciones de memoria, registros, biestables indicadores y puertos de entrada y salida. Entre los programas del modo ampliado, se incluye un módulo de desensamblado, que se comentará más adelante, y que por su propia naturaleza, puede ser considerada como un sub-modo del

modo ampliado, ya que dentro de él cambia el comportamiento de muchos objetos gráficos del entorno ECODE.

En la Figura 1 se incluye una imagen que muestra el aspecto del entorno funcionando en modo ampliado. Se ha tenido especial cuidado, en el diseño de la interfaz de usuario para hacerla lo más didáctica y cómoda posible. La disposición de los componentes gráficos ha sido especialmente pensada para lograr la mayor coherencia posible entre el entorno y la descripción real (física) de CODE-2. También se ha diseñado de forma que el puntero del ratón cambia de aspecto al superponerse a un objeto, indicando este hecho que el usuario puede interactuar directamente con dicho objeto. Además, al descansar unos instantes el puntero del ratón sobre un objeto gráficos relevante, se despliega automáticamente un rótulo indicando su significado.

El entorno está programado en Object Pascal, usando el entorno de Borland Delphi 5.

3. Características de ECODE

La Figura 1 muestra la interfaz gráfica de ECODE (versión 1.0.0 para Windows). La imagen se compone de cinco zonas o sub-ventanas, y que de arriba abajo, y de izquierda a derecha son:

1. Barra de menús, ubicada en la parte superior de la imagen y contiene las siguientes opciones :

- *Archivo*: Desde esta opción es posible abrir un archivo en ensamblador para editarlo, o para traducirlo y guardar el resultado en un archivo o cargarlo directamente en la memoria de CODE-2 para emular su ejecución.
- *Entorno:* con esta opción se pueden seleccionar, entre otras utilidades, los distintos modos de funcionamiento del entorno: real, ampliado y desensamblador, así como acceder al convertidor visual de archivos.
- *CODE*: esta opción permite interactuar directamente con el emulador de CODE-2.
- *Ayuda y manuales*: con esta opción se obtiene información referente al uso de ECODE, al funcionamiento de CODE-2 y un manual de ensamblador de CODE-2. También se incluyen ejemplos de programas y plantillas en ensamblador para facilitar el trabajo del usuario y la compresión de los conceptos que se pretenden transmitir al alumno.
- *Botones rápidos:* Permiten acceder de forma más ágil, a las rutinas más usadas, estén o no en el menú principal, y son los siguientes: Ver/Ocultar la ampliación del panel frontal, Desensamblar, Variar la velocidad de ejecución, Encender/Apagar CODE-2, Modo paso a paso, Dirección de memoria, Registros, Cargar, Ejecutar, y Continuar.

2. Sub-ventana de texto: visualiza el archivo del programa en desarrollo, en lenguaje ensamblado o máquina.

3. ***Panel frontal*** de CODE-2, que representa al prototipo físico. A esta imagen se le ha añadido en su parte superior tres solapas, para visualizar, alternativamente el panel frontal, los mensajes emitidos por el entorno a lo largo de la sesión, o el contenido de los registros.

4. Esquema completo de CODE-2, donde se incluyen, además de los elementos que aparecen en el panel frontal, los contenidos de todos los registros, de la memoria y de los puertos IP1 e IP2 de entrada, y OP1 y OP2 de salida. Todos los contenidos puede modificarse por medio de un editor hexadecimal. Además se permite arrastrar, ampliar y soltar donde el usuario desee la sub-ventana que representa la memoria.

5. Barra de estado, informa de la situación actual del programa.

Cuando ECODE funciona en modo real sólo aparece el panel frontal, y cuando funciona en modo ampliado se muestra completamente la imagen de la Figura 1. Por otra parte, el sistema ECODE contiene los siguientes módulos, integrados unos con otros:

- *Emulador*: simula el comportamiento de CODE-2.
- *Ensamblador*: traduce archivos de ensamblador de CODE-2 a su lenguaje máquina. La salida puede ser presentada en los formatos .HEX (hexadecimal) o .EHC (Easy Hexadecimal Code) descritos en [3].
- *Editor hexadecimal:* permite, por ejemplo, cargar directamente un programa en la memoria sin más que situarse con el ratón en la sub-ventana de memoria .
- *Desensamblador*: para desensamblar automáticamente un programa cargado previamente en la memoria de CODE-2. El resultado es

simulable posteriormente igual que si hubiera sido el resultado del ensamblaje normal de un programa.

- *Cargador*: Los programas pueden ser almacenados automáticamente en la memoria, después de haber sido traducidos a código máquina por el ensamblador, y de acuerdo con las directivas del programa fuente que especifiquen las posiciones de carga en memoria.
- *Módulo de información*: genera y almacena los mensajes de ECODE, tanto los de error como los de estado (modo de funcionamiento), y se pueden visualizar seleccionando la solapa *Mensajes.*
- *Convertidor Visual de Archivos*: gestiona un cuadro de diálogo con el que el usuario puede realizar la conversión entre los formatos .ASM, .HEX, .EHC, con la excepción de que no podemos pasar de .ASM a cualquier otro.

Algunas de las características del entorno, que enfatizan su fin pedagógico son:

- Visualización de los contenidos de todos los registros y biestables.
- Durante la ejecución de un programa se resalta la posición de memoria donde se encuentra la instrucción en ejecución.
- Interacción directa con los registros y puertos de entrada/salida. Se pueden editar cada uno de estos elementos, seleccionándolos con el ratón.
- Visualización del comportamiento dinámico de todos los contenidos; para lo cual se puede utilizar el modo de ejecución *paso a paso* (CODE-2 entra en modo de espera inmediatamente después de la ejecución de cada instrucción) o regular la velocidad de ejecución utilizando uno de los botones rápidos.

4. Utilización del entorno

Nada más iniciar ECODE, queda listo para empezar a simular. Implícitamente arranca en modo ampliado, y se visualiza la imagen de la Figura 1. Ahora, se puede elegir una de las opciones del menú *Archivo*, si se desea cargar un programa en código máquina, traducir un programa en ensamblador y guardarlo como código máquina, o editar un programa fuente, traducirlo, depurarlo y cargarlo en la memoria de CODE-2 para su ejecución simulada. Cuando se ejecuta un programa las sub-ventanas de memoria y de texto visualizan la información de forma coherente, resaltando los cambios que se van produciendo en la memoria y en el contador de programa.

En cualquier momento, aún con un programa cargado, podremos editar la memoria, los registros y los puertos, esta "edición en caliente" es muy útil para depurar programas o comprobar de forma sencilla el funcionamiento de cualquier instrucción.

4. Conclusiones

Hemos presentado en esta comunicación las características más notables de ECODE, un entorno integrado de desarrollo (IDE) orientado a la enseñanza, para un Computador Didáctico Elemental (CODE-2). El sistema puede obtenerse en la dirección web:

http://atc.ugr.es/intro_info_mcgraw.htlm

Referencias

[1] Página web principal del proyecto CODE-2 http://atc.ugr.es/intro_info_mcgraw

[2] A. Prieto, A. Lloris, J.C. Torres. *Introducción a la Informática*, 3ª Edc. Cap 6-7, McGraw-Hill 2002.

[3] A.Prieto, F.J.Pelayo, F.Gómez, J.Ortega, A.Cañas, A.Martínesz, F.J.Fernández. *Un computador didáctico elemental*.Actas de la VIII Jornadas de Enseñanza Universitaria de la Informática (JENUI'2002). Cáceres 10-12 Julio 2002. *(en este mismo libro)*

[4] A. Martínez. *Diseño de un Entorno Integrado de Desarrollo con emulador y compilador cruzado para el computador CODE-2*, Proyecto fin de carrera de Ingeniero Electrónico, Dpto. Arquitectura y Tecnología de Computadores, Universidad de Granada. Octubre 2001.

[5] *IEEE Standard for Microprocessor Assembly Language.* IEEE Std. 694-1985. The Institute of Electrical and Electronics Engineers, Inc. New York., June 30, 1985.

Simulador de dispositivos de entrada/salida programables

Manuel Prieto, Antonio J.Vicente, José. A. Vargas
Departamento de Automática
Universidad de Alcalá
28871 Alcalá de Henares
e-mail: mpm@aut.uah.es

Resumen

La ponencia presenta un entorno de simulación de la entrada/salida de sistemas basados en la familia i80x86. Está enfocados principalmente a servir como apoyo a los alumnos de la Universidad de Alcalá en las asignaturas de Sistemas Electrónicos Digitales y Laboratorio de Sistemas Electrónicos Digitales en la titulación de Ingeniería Técnica Telemática, y por extensión, a otras asignaturas afines de otras ingenierías tales como Estructura de Computadores.

1. Motivación

Es una problemática muy común en la enseñanza de sistemas digitales, ya sea de electrónica digital o de sistemas basados en microprocesador, la de ofrecer al alumno cuantas herramientas y recursos hardware estén a disposición del profesor para el correcto seguimiento y compresión de la materia en cuestión. Es evidente que la mejor forma de enseñar, y de que los alumnos comprendan los sistemas digitales, es trabajando con los dispositivos digitales reales que se enseñan en el aula. De este modo, es muy común encontrar en laboratorios de asignaturas de esta índole (Electrónica Digital, Estructura de Computadores, Sistemas Electrónicos Digitales, etc.) todo el equipamiento necesario, como por ejemplo placas de desarrollo, entrenadores, fuentes de alimentación, osciloscopios, etc. Otra opción, es recurrir a la simulación, con las ventajas y restricciones que ello conlleva. Como ventajas destacaremos que la simulación es una herramienta muy potente, puesto que permite evaluar un sistema previo a su montaje y estimar su comportamiento. Así mismo, otra ventaja que reporta la simulación es su coste, ya que resulta, sin lugar a dudas, menos costosa que equipar un laboratorio con material electrónico. No obstante, la principal desventaja de la simulación es la de que no deja de ser un complemento. Si el sistema simulado no llega a probarse físicamente, el objetivo que se persigue queda, en términos coloquiales, un poco "cojo". La situación ideal es aquella que incorpora parte de simulación y parte práctica real.

El hecho de que al laboratorio se le haya dado un enfoque puramente práctico, aún cuando es muy didáctico, puede suponer un gran impedimento para el alumnado en el desarrollo de las prácticas. Para "investigar" sobre los conceptos explicados en clase o la puesta a punto de las prácticas es imprescindible asistir al laboratorio, puesto que los alumnos no tienen los recursos económicos para adquirir el equipamiento necesario.

En el caso de que el laboratorio sea exclusivamente de simulación, el problema anterior queda subsanado siempre y cuando el simulador en cuestión no precise de una licencia que impida distribuir libremente el software entre el alumnado. En cualquiera de los casos, el alumno nunca llegaría a trabajar con los sistemas reales, que es, en definitiva, con lo que tendrá que enfrentarse en su vida laboral.

Como ya se ha resaltado, el equilibrio idóneo se obtiene de la combinación de simulación y práctica. Sin embargo hay que buscar que haya una correspondencia directa entre lo que se simula y lo que se prueba. En el campo de la enseñanza de electrónica digital y de sistemas basados en microprocesador existe una amplia gama de simuladores de muy diversa índole, en cuanto a sus capacidades y coste. Lo más frecuente es que ciertos simuladores destaquen por unos aspectos

pero que carezcan de otros, es decir, que es muy difícil encontrar un simulador que reúna todas las condiciones que se precisan. En el peor de los casos, no llega siquiera a encontrarse.

Para el caso que nos ocupa, relativo a la enseñanza de sistemas digitales basados en microprocesador, lo ideal sería disponer de un entorno de simulación lo más transparente posible para el alumno. Un entorno en el que trabajase con instrucciones reales, ensambladores reales y depuradores reales y que además que la simulación no tuviese límites. Una de las ventajas que tiene la arquitectura i80x86 frente a otras, es que no es necesario emplear emuladores o entrenadores; cualquier PC es suficiente. Prácticamente todos los alumnos disponen de un PC en casa, con lo que pueden trabajar desde ella. Sin embargo no se puede decir lo mismo cuando las prácticas dejan de ser de mera programación en ensamblador y se pasa a trabajar con dispositivos de entrada/salida (E/S).

Con este ánimo, estamos trabajando en el diseño de simuladores de sistemas periféricos que presentamos a continuación.

2. Objetivos

Tomando como base la asignatura Sistemas Electrónicos Digitales, los requisitos que deben cumplir los simuladores a desarrollar son los siguientes:

1. *Transparente al alumno*. El alumno debe tener la sensación de que está trabajando en todo momento con el dispositivo real y debe tener así mismo la certeza de que cuando lo pruebe sobre estos dispositivos obtendrá los mismos resultados. En el ensamblado, enlazado y posterior depuración de su programa debe emplear las mismas herramientas que emplearía para la aplicación real. Nuestros alumnos emplean el programa ensamblador y depurador MacroAssembler [2] y Codeview de Microsoft [3] respectivamente.
2. *Lo más reales posibles*. Los simuladores deben ajustarse lo más posible a los dispositivos que simulan. No deben carecer de ninguna de sus características.
3. *Didácticos*. Que a su vez los simuladores aporten información al alumno que trabaja con él y que le oriente en caso de que se equivoque en su manejo.
4. *Facilidad de trasladarlo a la realidad*. Que una vez realizada la simulación, el alumno, con unos cambios mínimos, pueda trasladar el código desarrollado al sistema real. Es decir, que no se vean obligados a re-escribir el código.
5. *Gráficos*. Debe ser un simulador gráfico de colores donde se distingan claramente las acciones tomadas por los alumnos en su programa.
6. *Reducido tamaño*. Lo ideal es que no resulten simuladores de gran tamaño. Se persigue que todos los ficheros necesarios para la simulación quepan en un disquete.
7. *Libre distribución*. Los simuladores deben ser gratuitos y abiertos. Al alumno no sólo se le facilitará el ejecutable, sino también su código fuente y una descripción del funcionamiento del mismo.
8. *Ampliables*. ¿Por qué restringir el simulador a una asignatura concreta en la que se estudian unos dispositivos concretos?. Tanto para el profesor como para el alumno es muy interesante que los simuladores sean ampliables.
9. *Multiplataforma*. Se persigue desarrollar simuladores que sean multiplataforma o que, en su defecto, la conversión del simulador de una plataforma a otra sea lo más sencilla posible.

3. Desarrollo de los simuladores

Buscando el cumplimiento de los requisitos expuestos en el apartado anterior, se ha pensado en el desarrollo de simuladores modulares, es decir, para cada dispositivo de E/S a simular se desarrollará un módulo diferente. De este modo, la capacidad de ampliación está asegurada. Estos módulos de simulación se programarán en ensamblador, lenguaje con el que trabajan los alumnos en las asignaturas y que conocen perfectamente. En estos programas se cargará por un lado la pantalla gráfica de simulación (VGA 640x480) y por otro lado se definirán los procedimientos necesarios. El alumno desde su programa llamará a esos procedimientos externos definidos en el módulo de simulación (figura 1).

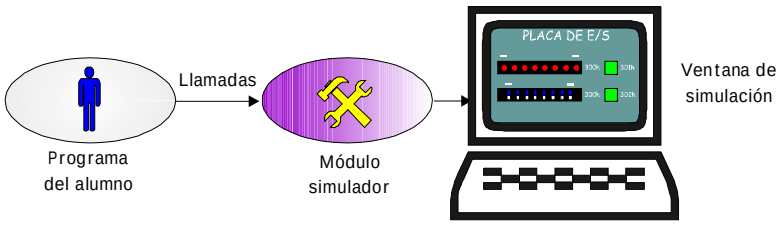


Figura 1. Diagrama de funcionamiento

La arquitectura i80x86, a diferencia de otros microprocesadores, tiene los espacios de direcciones de memoria y entrada/salida separados. Para procesos de E/S existen únicamente dos instrucciones: IN para entrada de puerto y OUT para salida a puerto. Para conseguir la máxima transparencia de cara al alumno, se ha pensado en definir dos instrucciones, llamémoslas virtuales, de entrada/salida que se asemejen lo más posible a las instrucciones reales. Estas instrucciones virtuales son en definitiva dos macros con las que el alumno interactuará con el simulador en cuestión.

Desde el punto de vista del alumno, lo único que cambia respecto a trabajar con un sistema real son las instrucciones de entrada/salida que empleen. Para el trabajo en simulación utilizarán las instrucciones virtuales, mientras que, cuando deseen trasladar su programa a una aplicación real, lo único que deberán hacer es sustituir esas instrucciones virtuales por las reales (operación de buscar y reemplazar en cualquier editor de texto).

Las dos instrucciones virtuales se han llamado INM y OUTM y su formato se corresponde exactamente con el de las originales. En la figura 2, puede apreciarse claramente lo sencillo que le resulta al alumno pasar del entorno de simulación al real.

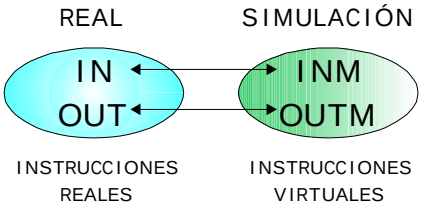


Figura 2. Simulación vs. realidad

Para operar con el simulador, el alumno sólo deberá añadir una pequeña cabecera en su programa y una vez ensamblado, enlazarlo con el fichero objeto del simulador.

Todos los simuladores desarrollados tienen asignadas unas direcciones de entrada/salida "virtuales" predeterminadas. Con fines didácticos, se ha dotado a los simuladores de capacidad de depuración. De este modo, en caso de que el usuario realice una acción no permitida, bien a propósito o bien por error (como por ejemplo acceder a una dirección errónea), se le informará de ello en pantalla, indicando la causa del error.

Por último, sobre la pantalla del simulador en cuestión se ha incluido el ratón, mediante el cual el alumno podrá interactuar sobre el simulador (cambio de estado de microinterruptores, pulsación de botones, etc.)

4. Simuladores

Hasta el momento, se han desarrollado tres simuladores que se ajustan a los temas tratados en Sistemas Electrónicos Digitales:

- Placa de Entrada/Salida. La ventana de simulación de la placa de E/S se muestra en la figura 3.

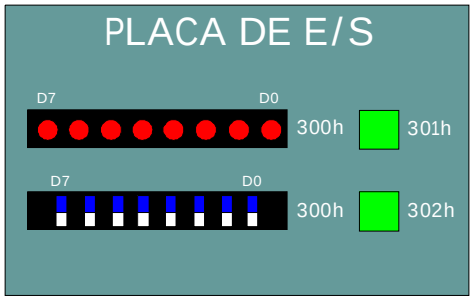


Figura 3. Ventana de la placa de E/S

En ella pueden distinguirse los elementos más característicos como son los micro-

interruptores y los LEDs para simular la entrada y la salida respectivamente.
Tal y como se ha resaltado previamente, las direcciones virtuales de la placa de E/S son fijas, aunque pueden fácilmente cambiarse, reensamblando el programa del simulador. Para el módulo desarrollado se han escogido las direcciones virtuales comprendidas entre la 300h y la 302h. Estas direcciones coinciden con las de la placa de E/S real que existe y que manejan los alumnos en las prácticas de laboratorio.

- Interfaz Paralelo Programable PPI 8255A. La ventana principal de simulación del dispositivo PPI 8255A [1] se muestra en la figura 4.

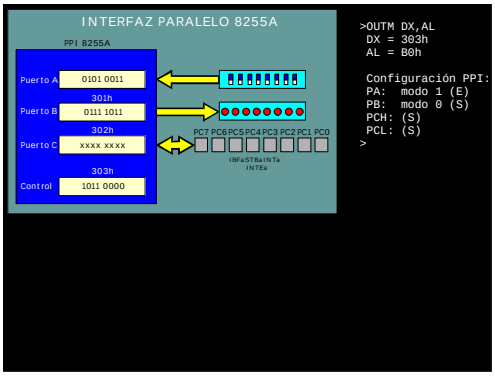


Figura 4. Ventana del simulador de la PPI

Pueden distinguirse claramente los puertos de la PPI, los registros internos y las líneas de control. Se trata de una ventana dinámica. Dependiendo de la configuración de la PPI, las líneas de control se reconfiguran adecuadamente. Así mismo en la parte izquierda de la ventana, el alumno dispone de un histórico de acciones. Si el alumno comete un error, se le informa de ello.

- Interfaz Serie USART 8251A. Como complemento al tema 6 de Sistemas Electrónicos Digitales, se ha desarrollado un simulador de la interfaz serie USART 8251A [1]. La ventana principal de simulación se muestra en la figura 5. Al igual que en el caso de la PPI, se muestran todos sus registros internos y las señales de interés.

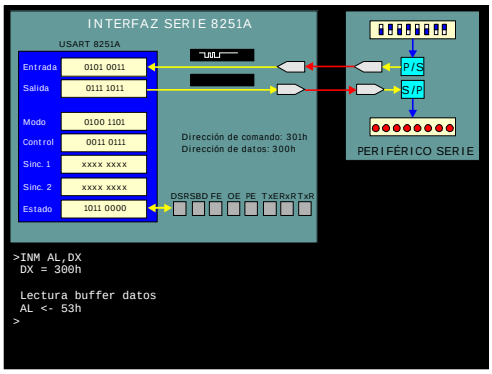


Figura 5. Ventana del simulador de la USART

5. Conclusión

Se han presentado tres simuladores de dispositivos periféricos para la familia i80x86. La idea del desarrollo de estos simuladores ha surgido de la experiencia docente y de las sugerencias de los alumnos de Sistemas Electrónicos Digitales. Pensamos que estos simuladores ayudarán a dichos alumnos en el próximo curso académico a comprender mejor la asignatura y servirán como complemento a su laboratorio. Se han desarrollado persiguiendo la máxima versatilidad de los mismos, cumpliendo los requisitos expuestos en el apartado dos. Nuestro propósito es continuar desarrollando nuevos simuladores (PIC 8259, DMA 8237, timer 8254, etc.) que contribuyan a la formación de los alumnos en los sistemas microprocesador. La filosofía en su desarrollo, permite la posibilidad de crear nuevos y potentes simuladores que incluyan una combinación de interfaces programables y periféricos (DACs, ADCs, etc.), siempre persiguiendo simular fielmente la entrada/salida.

Referencias

[1] Intel. *Microprocessor and Peripheral Handbook*, Volumen I y II. 1989.
[2] Microsoft. *Macro Assembler. Programmer's Guide*. 1987
[3] Microsoft. *Codeview and Utilites*. 1987

Una ayuda a la aplicación de técnicas heurísticas de optimización

Jose Antonio Lozano Alonso, Francisco Javier Echarte Ayerra
Dept. de Ciencias de la Computación e Inteligencia Artificial
Universidad del País Vasco/Euskal Herriko Unibertsitatea
e-mails: lozano@si.ehu.es laudix@infonegocio.com

Resumen

El presente artículo presenta una herramienta de ayuda a la implementación y experimentación con técnicas heurísticas de optimización. En concreto, las técnicas de optimización a las que da soporte son: el algoritmo de enfriamiento estadístico, los algoritmos genéticos y la búsqueda tabú.

Al mismo tiempo que se exponen las principales características de la herramienta, se ilustrará el uso que se hace de la misma en la asignatura "Optimización. Técnicas Avanzadas".

1. Introducción, Situación y Motivación

La asignatura "Optimización. Técnicas Avanzadas" [1] es una asignatura optativa de segundo ciclo, dentro del plan de estudios de Ingeniero en Informática en la Universidad del País Vasco. Dicha asignatura consta en el plan de estudios con una carga lectiva de seis créditos (cuatro teóricos y dos prácticos). El número de alumnos matriculados en la asignatura oscila entre veinte y treinta y cinco. El primer coautor de la presente ponencia viene impartiendo esta asignatura durante los cuatro últimos años.

Los contenidos teóricos de la asignatura se centran en la resolución de problemas de optimización combinatoria [3] mediante técnicas heurísticas. Las más destacables son: búsqueda local, enfriamiento estadístico (*simulated annealing* (a veces citado como: temple, recocido simulado, o enfriamiento simulado)), búsqueda tabú y algoritmos genéticos [2].

La parte práctica de la asignatura consiste en la resolución de un problema de optimización combinatoria mediante el uso de los algoritmos vistos en clase y, variaciones de los mismos específicas para el problema concreto a resolver. Éstos son problemas clásicos de optimización combinatoria como: el problema del agente viajero, el problema de la mochila, el problema de la partición del grafo, etc.

Una de las dificultades a la que se enfrentaban los alumnos al realizar la parte práctica de la asignatura era que, la implementación de los algoritmos presentados en las clases teóricas no es sencilla. Esto implicaba que una mayor parte del tiempo lo invertían en la implementación de uno de los algoritmos, quedando muy poco tiempo para la comparación de las soluciones obtenidas con diferentes conjuntos de parámetros y de componentes para dicho algoritmo, y por supuesto no pudiendo llevar a cabo una comparación experimental con diferentes métodos.

Dado que los algoritmos son conceptualmente fáciles de entender, la implementación en este caso, no aportaba ningún conocimiento extra a los alumnos.

Una solución a esta situación es la utilización de paquetes de software estándar que tuviesen implementados los algoritmos. Sin embargo, no existe actualmente ninguna herramienta que implemente todos los algoritmos, y menos en lengua española. Esto suponía que los alumnos debían familiarizarse con más de un paquete, lo que implica nuevamente un gran esfuerzo.

Para solucionar dicha dificultad hemos elaborado una herramienta que supone una ayuda a la aplicación de estos algoritmos: recoge todos los algoritmos vistos en clase bajo un mismo entorno; tiene implementados un conjunto de componentes y de opciones que permiten la comparación experimental de diferentes algoritmos; y además es abierta en el sentido de que es posible añadir nuevos componentes al sistema, de forma sencilla.

El resto del artículo está organizado de la siguiente manera: la sección 2 describe en líneas generales la herramienta. La sección 3 trata sobre las posibilidades de aumentar el VOpt, dedicándose la sección 4 a explicar el entorno. La sección 5 apunta las limitaciones del sistema y se termina apuntando las conclusiones en la sección final.

2. VOpt: Una herramienta de ayuda a la aplicación de técnicas heurísticas de optimización

2.1 Aspectos generales y metodología de utilización básica

VOpt (Visual Optimization) [4] es una herramienta de ayuda a la aplicación y experimentación con técnicas heurísticas de optimización. Dicha herramienta se ha centrado en tres algoritmos heurísticos (los más utilizados atendiendo a la literatura del área): el algoritmo de enfriamiento estadístico, los algoritmos genéticos y la búsqueda tabú.

La herramienta codifica los componentes fundamentales de estas tres técnicas de optimización. De esta forma, el estudiante, una vez elegido el algoritmo a utilizar, decide dentro del conjunto de componentes disponibles aquellos que más le interesen. Esto permite que el usuario concentre su atención en la implementación de la función de coste o función objetivo del problema de optimización a resolver.

Una vez codificado el procedimiento que implementa la función de coste y elegidos los componentes del algoritmo, el sistema genera código C. Si se dispone de un compilador de C en el ordenador es posible compilarlo desde el entorno de VOpt, en caso contrario el código C generado se puede trasladar a cualquier otra máquina que disponga de un compilador y compilarlo en dicha máquina.

Si la compilación se realizó dentro del entorno, el sistema devuelve los errores de compilación si es que estos existían (los asociados a la codificación de la función objetivo). Finalmente, se dispondrá de un fichero ejecutable. Dicho fichero puede ser ejecutado desde el sistema y los datos de salida son almacenados en un archivo que gestiona VOpt.

Dado que estos algoritmos utilizan de alguna forma la aleatoriedad, los resultados variarán de una ejecución a otra, por lo que resulta interesante realizar varias ejecuciones si es que se quiere obtener alguna información de valor. Por esta razón, el sistema ofrece también algunas funciones para calcular estadísticas sencillas utilizando los ficheros de salida.

2.2 Componentes básicos de los métodos de optimización

Un elemento común a todos los algoritmos de optimización heurísticos es la codificación de la solución. Cada solución a un problema de optimización dado debe codificarse de una forma concreta que permita almacenarlo en el ordenador.

La herramienta VOpt dispone por defecto de las siguientes codificaciones: lista binaria, lista de reales, lista de enteros y permutaciones. Cada una de estas codificaciones dispone de dos inicializaciones diferentes, una aleatoria y otra a partir de un fichero de datos.

El algoritmo de enfriamiento estadístico dispone del siguiente conjunto de componentes: sistema de vecinos, longitud de las cadenas de Markov, función de reducción de la temperatura y criterio de terminación. Además de los componentes anteriores el sistema proporciona al usuario la posibilidad de establecer el valor inicial del parámetro temperatura.

En relación a los algoritmos genéticos los componentes configurables son: elección de la población inicial, operadores de selección, cruce y mutación y el criterio de parada. Los parámetros a los que se debe asignar valor son: probabilidad de cruce, probabilidad de mutación, tamaño de la población y longitud del individuo.

La búsqueda tabú dispone de los siguientes componentes: sistema de vecinos, criterio de parada, criterios de aspiración. Los parámetros que deben ser establecidos son: longitud de la lista tabú, tiempo que un atributo permanece tabú, longitud de la intensificación y de la diversificación.

3. VOpt: Un sistema abierto

Una de las características a destacar del sistema es que es una herramienta abierta. Es decir, es posible incorporar de forma sencilla nuevos componentes.
Esta característica es fundamental si trabajamos con algoritmos de optimización. Cada problema tiene sus propias características, y por lo tanto puede ocurrir que los componentes que ofrece por defecto el sistema no sean los más adecuados. Por ejemplo es posible que necesitemos una codificación diferente.

Para crear nuevos componentes el sistema dispone de un conjunto de ayudas. Por ejemplo, en el caso de una nueva codificación, una vez definida la misma es necesario codificar también las funciones de creación, inicialización, copia y destrucción. A la hora de crear las funciones anteriores, una de las ayudas que proporciona el sistema es la cabecera de dicha función.

Sin embargo no sólo es posible crear componentes, sino también modificar los ya existentes. Por ejemplo, pueden existir situaciones donde sea interesante que un algoritmo genético utilice sólo mutación y no cruce. Cada algoritmo viene dado por una plantilla, luego en este caso hay que construir una nueva plantilla. La construcción de las plantillas se facilita mediante la utilización de directivas, que hacen que algunos componentes de la plantilla sean obligatorios mientras que otros sean opcionales.

4. VOpt: Un entorno amigable

Otra característica a destacar de la herramienta VOpt es su entorno. Proporciona un interfaz gráfico con menús y ventanas de tipo Windows.

El acceso a las diferentes funcionalidades se puede realizar de diferentes formas permitiendo al usuario utilizar la que más le convenga en cada momento.

El sistema tiene integrado una ayuda donde además de contener información acerca de cada una de sus funcionalidades y componentes, proporciona una introducción a los métodos de optimización que se implementan. Además cada componente tiene asociada una descripción que aparece en la pantalla cuando se pincha con el ratón en dicho elemento, de esta forma el usuario puede conocer que representa el componente que se va a añadir al algoritmo que se quiere construir.

La figura 1 muestra un posible estado del interfaz. La información que refleja el entorno nos indica que estamos codificando un algoritmo de enfriamiento estadístico para un problema de minimización. El usuario se encuentra eligiendo una codificación para las soluciones, teniendo seleccionada (TlistaBinaria) una lista de ceros y unos. Obsérvese también como se muestra el código C de dicha componente.

5. VOpt: Puntos débiles

VOpt también tiene algunos puntos que son susceptibles de mejora. El principal problema es que en la actualidad, únicamente se ejecuta de forma correcta bajo Windows NT. El sistema fue desarrollado por el segundo coautor de este escrito, como proyecto fin de carrera en el año 1999, y desde entonces la tecnología ha superado el sistema operativo Windows NT. Por supuesto no se dispone de ningún soporte técnico, ni está prevista ninguna actualización.

En relación con las capacidades de la herramienta en sí, hay aspectos que podrían ser mejorados. Por ejemplo, las herramientas de que se dispone para analizar los resultados de diferentes ejecuciones son muy básicas, y en la mayoría de los casos hay que utilizar otros programas auxiliares (Excel, SPSS, etc.).

Otro punto débil es el control de las ejecuciones y de bs datos que se generan. Por ejemplo, el establecimiento de los parámetros para estos algoritmos se realiza en la mayoría de los casos tras experimentar con ciertos conjuntos de ellos. Una persona experta en la aplicación de estos algoritmos es capaz de descubrir fallos u obtener mejores conjuntos de parámetros simplemente observando una traza de la ejecución del algoritmo (por ejemplo en el caso del algoritmo de enfriamiento estadístico, la información que debería proporcionar la traza es: valor del parámetro temperatura, solución actual, solución vecina elegida, número de aceptados con el valor de t fijo y mejor solución conseguida hasta el momento). Desafortunadamente el sistema actual no permite la obtención de este tipo de trazas.

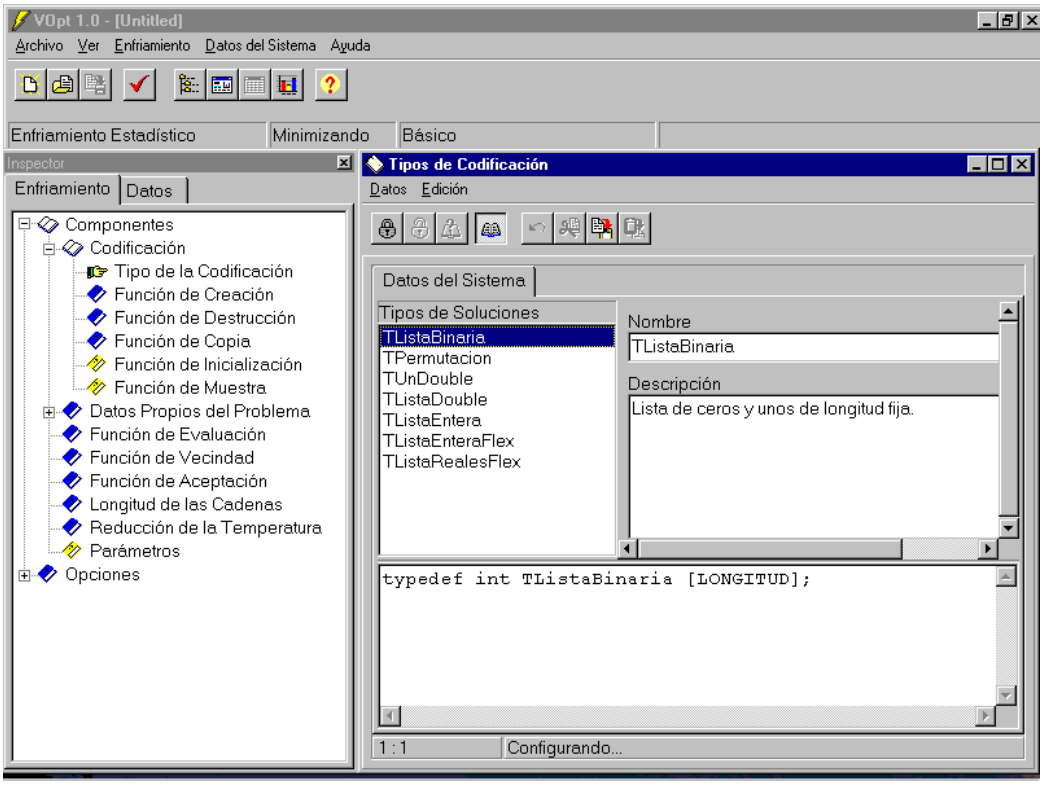


Figura 1. Ejemplo de uso del entorno VOpt

6. Conclusiones

En este escrito hemos presentado el software VOpt. Este software constituye una ayuda en la aplicación y experimentación con algoritmos heurísticos de optimización. El entorno proporciona la oportunidad de que el usuario se centre en la implementación de la función objetivo obviando los detalles de la implementación del algoritmo de optimización. Al mismo tiempo el sistema aúna bajo un mismo entorno tres técnicas que optimización hasta ahora dispersas en paquetes diferentes. Finalmente destacar que el sistema es un sistema abierto que permite añadir nuevos componentes de forma sencilla.

La utilización de VOpt en la asignatura ha sido muy satisfactoria. Los alumnos han realizado experimentos con varias técnicas de optimización, adquiriendo mas conocimiento experimental acerca de su funcionamiento.

Agradecimientos

Los autores desean agradecer las sugerencias de la profesora Susana Rodríguez.

Referencias

[1] *Guia Docente de la Facultad de Informática 2001-2002.* Universidad del País Vasco. San Sebastián.

[2] C. Reeves. *Modern heuristic techniques for combinatorial optimization.* Blackwell scientific publications, 1993.

[3] C. Papadimitriou y K. Steiglitz. *Combinatorial optimization: Algorithms and complexity.* Prentice-Hall, 1982.

[4] F. Echarte. *Entorno de ayuda a la aplicación de técnicas de optimización.* Proyecto fin de carrera.. Fac. Informática UPV-EHU, 1999.

ENTORNO MULTIMEDIA DE APOYO A LA DOCENCIA

José Poveda, Julian Gutierrez
Departamento de Informática
Universitat de València
46100 Valencia
e-mail: jose.f.poveda@uv.es

Resumen

Un aspecto muy importante en la enseñaza es el seguimiento y evaluación continuada del alumno a lo largo del periodo en el cual se imparte la asignatura. Por desgracia, en aquellos estudios universitarios que presentan masificación de asignaturas, cubrir estos objetivos con el grado que sería deseable no es siempre posible.

En el presente artículo exponemos, la concepción, especificación, funcionamiento y resultados de una experiencia piloto con un sistema tutor para el seguimiento y evaluación continuada del alumnado. El sistema tutor, concebido como un entorno multimedia de apoyo a la docencia bajo una arquitectura cliente-servidor es una materialización particular de los genéricamente llamados Sistemas Tutores Inteligentes.

El sistema tutor, ha sido probado durante el curso académico 00-01, 01-02, en la asignatura de Informática II de la Diplomatura de Biblioteconomía y Documentación de la Universitat de València.

1. Introducción

A lo largo de nuestra experiencia docente hemos detectado ciertos aspectos que cada vez nos parecen más importantes para el correcto planteamiento y funcionamiento de cualquier asignatura como son el saber:

1. ¿Cómo aprenden los alumnos?
2. ¿Cuándo aprenden?
3. ¿Sobre qué contenidos específicos encuentran problemas?
4. ¿Qué nivel previo tiene la clase?
5. ¿Cómo evoluciona el nivel de un alumno concreto, a lo largo del curso?
6. Dado un nivel, ¿qué tanto por ciento de la clase superaría dicho nivel en una evaluación final y oficial de la asignatura?.

En este sentido, las preguntas que nos planteamos son muchas y de gran interés para el correcto planteamiento de una asignatura. El conocimiento de estos interrogantes, establecería una interacción fundamental entre el profesor y la clase. Esta comunicación bidireccional favorecería de forma global el proceso de enseñanza-aprendizaje.

A lo largo de nuestra experiencia docente se ha observado que aquellos alumnos que repetidamente encuentran dificultades en la comprensión de la materia, si éstas no son subsanadas, conducen a un irremediable abandono del la asignatura por parte del alumno.

Motivados por esta estrecha relación entre las dificultades que encuentran los alumnos y el abandono de la asignatura, se cree necesaria la elaboración de algún sistema que nos informe de la marcha del estudiante y de sus posibles deficiencias. Por otro lado, el sistema informa al propio estudiante de su nivel de conocimientos en la asignatura, de las deficiencias detectadas y de los conocimientos que debe reforzar para la subsanación de tales deficiencias.

Este sistema tutor ha sido implementado en una plataforma web, que toma las entradas de los test de evaluación efectuados de forma personalizada a cada alumno de forma que el sistema experto informa al alumno de su estado de conocimientos sobre el tema del que ha sido evaluado o, de la

asignatura en general, para el caso de un control de evaluación continua de contenidos.

El presente artículo se ha dividido de la siguiente manera: en primer lugar intentamos, analizar las bases del proceso enseñanza-aprendizaje, en segundo lugar, estudiamos con detalle el sistema tutor inteligente. Una vez vistas las diferentes partes del sistema, definiremos los objetivos funcionales del sistema experto.

Aunque el sistema ha sido probado para la asignatura de primer curso, Informática II de la Diplomatura en Biblioteconomía y Documentación, cuyos contenidos corresponden a fundamentos de la informática y la programación, el sistema ha sido diseñado e implementado para aceptar los contenidos de cualquier asignatura con un formato establecido en XML. Para finalizar, mostraremos los resultados obtenidos en la prueba piloto de dicho sistema, así como las conclusiones que se extraen de dichos resultados.

2. El proceso enseñanza-aprendizaje

Tradicionalmente, el papel que desempeña el personal docente con respecto a los alumnos durante el proceso de aprendizaje, se limita a una evaluación de los conocimientos adquiridos, que muchas veces se realiza por medio de un simple examen. La universidad viene utilizando un sistema que se basa en la "desasistencia liberal" [BLO71], en donde el estudiante carece de un sistema de control y de tutorización sistemática.
Erróneamente se ha supuesto que la ayuda pedagógica que debe recibir el estudiante está en relación con su edad o madurez, y que por tanto, lo principal para el profesor es el dominio de la materia y no tanto de la pedagogía.

Las clases teóricas han sido y continúan siendo la principal actividad de la Universidad. Se desarrollan normalmente como clases magistrales, teniendo como principal fin, presentar sistemáticamente con rigor científico y claridad expositiva, el contenido de la asignatura. Esta tradición viene dada por el profundo arraigo que tiene este esquema, y por la dificultad de crear cambios sustanciales en el mismo. Esta dificultad es debida a la masificación de las carreras universitarias, donde un gran número de alumnos, no presentan ni la motivación, ni la preparación adecuada [MAR97]. Este comportamiento desanima al profesor, que por otra parte y, debido a la necesidad de reconocimiento, se centra más en aspectos investigadores que en las cuestiones docentes.

Debido a estos problemas, ha sido necesario un análisis profundo para tratar de solventar las deficiencias en el sistema educativo y detectar la forma en que los cambios sociales y tecnológicos contribuyen a la generación de nuevas necesidades en la sociedad.

Los estudios realizados sobre los procesos de aprendizaje revelan la necesidad de insistir en los aspectos cualitativos de la educación y en su carácter continuo e interdisciplinario. Por tanto, el profesor no debe limitarse a la transmisión de conocimientos, sino que debe de facilitar la labor de aprendizaje por parte del alumno, creando por ejemplo nuevos hábitos de estudio, facilitando el trabajo individual y la colaboración en equipo, el uso de la red informática, así como las capacidades fundamentales de expresar ideas y argumentar ideas.

En este sentido, las tecnologías de la información y la comunicación (TIC's) en general e INTERNET en particular, son el marco adecuado para el desarrollo de un sistema que nos permita el seguimiento y evaluación personalizada de cada uno de los alumnos pertenecientes a un grupo independientemente del grado de masificación en las aulas.

3. Los sistemas tutores inteligentes

Un sistema tutor inteligente [PAP98] es un sistema capaz de ayudar al estudiante en el aprendizaje de diversos conocimientos.

Se podría decir también que se trata de un sistema semi-presencial de conocimiento, debido a que el profesor no está presente y es el propio sistema el que guía al alumno en la asimilación de los diferentes conceptos.
Cualquier sistema tutor inteligente está formado por cinco módulos bien diferenciados [PAP98], como se muestra en la figura 1.

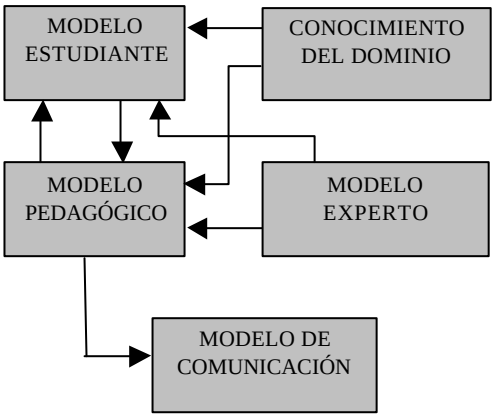


Figura 1: Módulos de un sistema tutor inteligente

Las tareas que desarrolla cada uno de estos módulos las podemos resumir como sigue:

1.- Modelo del estudiante: recoge datos históricos sobre los resultados y características del alumno.

2.- Modelo pedagógico: es el módulo capaz de pronosticar la evolución del alumno y su correspondiente solución (en este módulo se integra nuestro Sistema Experto).

3.- Modelo de comunicación: encargado de establecer el interfaz o comunicación entre el estudiante y el sistema.

4.- Conocimiento del dominio: módulo que recoge toda la información sobre la asignatura (temario, problemas, ejercicios, esquemas, exámenes, etc.).

5.- Modelo experto: Es capaz de discernir las respuestas del alumno hacia el sistema y verificar si son correctas o no. Además debe definir un perfil en el nivel de conocimientos del alumno.

3.1 Diseño del sistema

Las características de implementación de los diferentes módulos del sistema tutor inteligente son las siguientes:

1.- Modelo del estudiante. Los datos relativos al alumno y sus registros de evaluación son almacenados en la base de datos del sistema implementada con MySQL sobre plataforma Linux. Esta decisión se debe a que el sistema operativo y la base de datos seleccionadas son de libre distribución con la ventaja de que el alumno puede encontrar el software y una completa documentación de forma gratuita.

2.- Modelo pedagógico. Este módulo básicamente está compuesto por un sistema experto basado en reglas e implementado en código C++ con conexión a la base de datos a través de ODBC, que permite el acceso a los datos específicos de cada estudiante dado de alta en el sistema. El sistema experto genera test de evaluación a tres niveles, basados en la taxonomía de Bloom, y adaptados al perfil del estudiante.

3.- Modelo de comunicación. La interfaz del sistema ha sido implementada en página web con DHTML y php para el acceso a la base de datos del sistema.

4.- Conocimiento del dominio. El sistema presenta un entorno multimedia con material docente para el estudio de la asignatura en página web. Por otra parte y de forma activa, el sistema tutor evalúa al alumno sobre los objetivos docentes definidos en cada tema a partir de preguntas test clasificadas a tres niveles y almacenadas en la bases de datos del sistema. La entrada de los datos modela los siguientes aspectos para cada pregunta:

1. Enunciado de la pregunta
2. Conceptos que cubre
3. Opciones posibles
4. Respuesta correcta
5. Nivel de la pregunta

5.- Modelo experto de corrección. La corrección de los test se realiza directamente por acceso a la base de datos a través de php. Posteriormente el sistema experto analiza los resultados y emite un informe.

4. Evaluación del sistema

El sistema ha sido utilizado en la Diplomatura de Biblioteconomía y Documentación de la Universita de València, para la asignatura de primer curso de la diplomatura, Informática II.
Hay que decir que no se trata de una carrera técnica, si bien aborda algunos aspectos tecnológicos, y tiene una gran componente en

informática. Los alumnos que acceden a la titulación, lo hacen principalmente con un perfil muy poco tecnológico, por lo que los conocimientos en informática son mínimos o nulos. Esto hace, especialmente dificultoso la consecución de los objetivos propuestos si el planteamiento se hiciera de forma análoga a como se haría en una carrera técnica. Es por ello, que el desarrollo de un sistema que apoye, refuerce y oriente al alumno en el proceso de enseñanza, de forma adicional a la docencia impartida en las aulas es especialmente crítico en este marco de trabajo, para poder alcanzar unos objetivos docentes adecuados.
El sistema tutor, ha sido probado durante el curso académico 00-01, 01-02, en la asignatura de Informática II de la Diplomatura en Biblioteconomía y Documentación de la Universitat de València.

5. Resultados obtenidos

Tras la implantación del sistema tutor, la motivación por la asignatura que se eligió como asignatura piloto y la integración de los alumnos en la misma ha aumentado de forma considerable en los últimos años. En el curso académico 99-00 el porcentaje de alumnos no presentados en primera convocatoria en la asignatura Informática II de la Diplomatura en Biblioteconomía y Documentación era del 60% y del 40% de los presentados, el número de aprobados era de un 30%. Después de dos años de uso del sistema el número de no presentados a disminuido considerablemente a un 20% y de los presentados el número de aprobados es de un 60% en primera convocatoria.

6. Conclusión

Se ha presentado un sistema multimedia para el seguimiento y evaluación continuada del alumno. El sistema ha sido pensado para su uso simultáneo y complementario al esquema clásico de clase magistral.

El sistema ha sido probado durante los cursos académicos 00-01, 01-02 con alumnos de primer curso de la diplomatura en Biblioteconomía y Documentación de la Universitat de València los cuales cursaban la asignatura de Informática II. Tras un análisis de los resultados obtenidos en estos dos años de evaluación, llegamos a las siguientes conclusiones:

El sistema no solamente constituye un perfecto complemento al sistema tradicional de enseñanza sino que orienta de forma personalizada al alumno en la evolución de sus estudios por medio del sistema experto que analiza los errores conceptuales de fondo que el alumno comete en los test de evaluación. No se trata pues de un mero repositorio de documentación a la cual puede acceder el alumno para recoger material de la asignatura ni tampoco de un corrector de exámenes automatizado, sino que el sistema presenta un lado reactivo en la interacción con el usuario que tiene en cuenta la evolución en el proceso de aprendizaje del alumno.

Por otra parte, el profesor tiene criterios adicionales para la evaluación del alumno, ya que el sistema proporciona la trayectoria de cada uno de los alumnos en su proceso de aprendizaje. El profesor, por tanto, puede disponer de parámetros como: el tiempo dedicado por el alumno a cada tema, los resultados parciales obtenidos, así como del nivel global de asimilación de la materia.

7. Referencias

[DEN98] Denis Zambrano, J., et al..*Tecnologías de la información en la educación.* Anaya Multimedia. 1998.

[PAP98] Paproch, Kenneth. *Distance Learning: The ultimate Guide*, London, Sage Publications, 1998.

[DOM99] Doménech Betoret, F. .Proceso de enseñanza/aprendizaje universitario.Universitas, Universidad Jaume I. 1999.

[BLO71] Bloom B. Taxonomía de los objetivos de la educación. Ateneo. Buenos Aires. 1971.

[MAR97] Martín G., Morales F., Cerverón V. Relación entre los resultados de la prueba de acceso a la universidad y el rendimiento académico en el primer curso universitario. Universitat de València. 1999.

Depuración de Programas Haskell a partir de Especificaciones PRE/POST en Haskell

J. M. Burgos, J. Galve, J. García, M. Sutil
Dept. Lenguajes y Sistemas Informáticos (LSIIS)
Universidad Politécnica de Madrid
28660 Boadilla del Monte - Madrid
e-mail: {jmburgos, jgalve, juliog, msutil}@fi.upm.es

Resumen[1]

En la siguiente propuesta[2] se presenta una versión inicial de la herramienta de depuración declarativa llamada *H4D2*[3], la cual forma parte de un proyecto de investigación aplicado a la innovación educativa [8].

El objetivo principal de este trabajo es ofrecer herramientas metodológicas e instrumentales que faciliten el uso de metodos formales en un primer curso de programación en Haskell [5].

1. Un depurador de programas a partir de las Especificaciones PRE/POST

1.1. Estructura del Depurador *H4D2*

La herramienta *H4D2* debe permitir al operar en los modos *depuración / no_depuración*. Para hacer esto posible, la implementación incluye una constante debug que actúa a modo de *flag* para depurar o no. Para pasar de un modo a otro, se debe cambiar el valor de esta constante (True/False).

```
-- Flag de depuracion
debug :: Bool
debug = True
```

```
-- Mensaje de error en PRECONDICION
error_pre :: String -> String
error pre nombreFuncion =
 "ERROR EN PRECONDICION: " ++ nombreFuncion

-- Mensaje de error en POSTCONDICION
error_post :: String -> String
error_post nombreFuncion =
"ERROR EN POSTCONDICION: " ++ nombreFuncion
```

El modo en que opera el depurador declarativo H4D2 sigue la secuencia lógica definida por la estructura de las especificaciones PRE/POST. Así, dada una función Haskell con la estructura descrita en la sección 3.1, la ejecución de la depuración seguirá la siguiente secuencia de evaluaciones:

DEPURADOR (pre, funcion, post) (x) =

Paso 1) *EVALUAR* pre (x)
Paso 2) resultado = *EVALUAR* funcion (x)
Paso 3) *EVALUAR* post (x, resultado)

Para la implementación, se ha optado por la definición de una función de orden superior depurar que actúa sobre las funciones pre, post y funcion. Se ha añadido, así mismo, un parámetro de tipo cadena de texto que representa el nombre de la función que se quiere depurar. Este último parámetro lo utilizaremos más adelante para asociar los mensajes de depuración a la función que se evalúa.

[1] Una versión completa de este trabajo puede encontrarse en [9].

[2] Este trabajo ha sido parcialmente financiado por el proyecto español TIC 01-2798.

[3] H*askell for* D*eclarative* D*ebugging*

```
-- Interprete de depuración
depurar :: (b -> Bool, b -> a, b -> a ->
                    Bool, String) -> b -> a
depurar (pre, funcion, post, nombre) (x) =
 eval PRE (pre (x))
   (eval_POST ((funcion x) (post x) nombre)
    nombre
```

Obsérvese, que la secuencia de evaluaciones se implementa mediante composición de funciones (eval_PRE, eval_POST). La anterior secuencia de evaluaciones puede interrumpirse si bien la precondición o la postcondición no se cumplen.

```
-- Función que evalúa la PRECONDICIÓN
eval_PRE :: Bool -> a -> String -> a
eval_PRE pre funcion nombre =
        if debug then
           if pre then funcion
           else error (error_en_pre nombre)
        else funcion

-- Función que evalua la POSTCONDICION
eval_POST :: a -> (a->Bool) -> String -> a
eval_POST funcion post nombre =
     if debug then
       if post funcion then funcion
       else error (error en post nombre)
     else funcion
```

Como puede verse, utilizamos el valor de la constante debug para evaluar en modo depuración (evaluar la precondición y la postcondición) o modo normal. Finalmente, caso de que se incumpla alguna de las condiciones, debe dispararse una excepción y finalizar la ejecución. Esto lo realizan las siguientes funciones error_pre y error_post, las cuales están parametrizadas con el nombre de la función que estamos depurando.

2. Ejemplos de Depuración

A continuación se presentan 2 ejemplos representativos del uso de la técnica propuesta para depurar programas Haskell. Los ejemplos se presentan ya con la conversión de especificación a depuración.

2.1. Ejemplo 1: *"Número Primo"*

"Determinar si un número $n \in \mathbb{Z}^+$ es primo".

Para resolver el problema definimos la función auxiliar `hay_divisor`, la cual habíamos especificado y se ha obtenido la siguiente versión:

```
import Debug

-- PROBLEMA: Determinar si en [2, x] hay
--           algún divisor de n

hay_divisor :: (Int, Int) -> Bool
hay_divisor = depurar (pre,fun,post,name)

where
--   name :: String
     name = "hay_divisor"
--   pre :: (Int,Int) -> Bool
     pre (x, n) = n > 0
--   funcion :: (Int,Int) -> Bool
     fun (1, n) = False
     fun (x, n) = (n `mod` x) ||
                  hay_divisor (x-1, n)
--   post :: (Int,Int) -> Bool -> Bool
     post (x,n) res = res ==
     (existe (ini,sig,min) (\y->True)
                           div_inf)(x,n)
--   ini :: (Int,Int) -> (Int,Int,Int)
     ini (x, n) = (2, x, n)

--   sig :: (Int,Int,Int) -> (Int,Int,Int)
     sig (inf,sup,n) = (inf+1,sup, n)

--   min :: (Int,Int,Int) -> Bool
     minimal (inf,sup,_) = (inf > sup)

--   div_inf :: (Int,Int,Int) -> Bool
     div_inf (inf,sup,n) = divide (inf,n)
```

Ahora ya, describimos la versión de depuración del problema de calcular si un número es primo o no, como sigue:

```
-- PROBLEMA: Determinar si un número n es
--           primo.

es_primo :: Int -> Bool
es_primo = depurar (pre, fun, post, name)
  where
-- name :: String
   name= "es_primo"
-- pre :: Int -> Bool
   pre n = n > 0
-- fun :: Int -> Bool
   fun n = (n == 1) || (n == 2) ||
           not (hay_divisor (n `div` 2, n))
-- post :: Int -> Bool -> Bool
   post n res =
res ==
       n==1
    || n==2
    ||(todos (ini,sig,min)(\x -> True)
                           no_divide n)
-- ini :: Int -> (Int, Int)
   ini n = (n-1, n)
-- sig :: (Int, Int) -> (Int, Int)
```

```
   sig (x, n) = (x-1, n)
-- min :: (Int, Int) -> Bool
   min (x, n) = (x == 1)
-- no divide :: (Int, Int) -> Bool
   no divide (x, n) = (n `mod` x /= 0)
```

2.2. Ejemplo 2: "Operaciones con Dígitos"

a) *"Determinar los anillos de un dígito"*

```
-- PROBLEMA : Determinar el número de
              anillos de un dígito
anillos_digito :: Int -> Int
anillos_digito = depurar(pre,fun,post,name)
where
--  name :: String
    name = "anillos_digito"
--  pre :: Int -> Bool
    pre (n) = n >= 0
--  fun :: Int -> Int
    fun (n)  = if (n == 8) then 2
               else if (n == 0 ||
                        n == 6 ||
                        n == 9) then 1
               else 0
-- post :: Int -> Int -> Bool
   post n resultado = resultado == anillos
      where anillos  | (n == 8)       = 2
                     | (n == 0) ||
                       (n == 6) ||
                       (n == 9)       = 1
                     | otherwise      = 0
```

b) *"Determinar los anillos de un número $n \in \mathbb{N}$"*

```
-- PROBLEMA : Determinar los anillos de
              un número Natural n.
anillos_numero :: Int -> Int
anillos_numero = depurar(pre,fun,post,name)
  where
--  nombre :: String
    nombre = "anillos_numero"
--  pre :: Int -> Bool
    pre (n) = n >= 0
--  fun :: Int -> Int
    fun n = if (n < 10) then
              anillos_digito (n)
            else
              anillos_digito (n `mod` 10) +
              anillos_numero (n `div` 10)
--  post :: Int -> Int -> Bool
    post n resultado =
    resultado ==
    suma (inicial, siguiente, minimal)
         (\x -> True) anillos_digito_i (n)
--  ini :: Int -> (Int, Int)
    ini (n) = (n, numDigitos(n))
--  sig :: (Int, Int) -> (Int, Int)
    sig (n, i) = (n, i-1)
--  min (Int,Int) -> Bool
    min (n, i) = i == 0
```

3. Conclusiones y Trabajos Futuros

En este trabajo hemos presentado una propuesta de depuración de programas Haskell a partir de las especificaciones PRE/POST descritas en Haskell.

En el futuro, planeamos mejorar la herramienta H4D2 con más facilidades de depuración y elementos que hagan más amigable la depuración (trazas paso a paso, evaluación de elementos del programa, etc..).

Referencias

[1] Burgos, J.M., Galve, J., García, J. and Sutil M.: Una Taxonomía de Problemas para la Enseñanza de la Programación, *Actas del VII INFOREDU'2000*, Mayo 2000, La Habana (Cuba).

[2] Burgos, J.M., Galve, J., García, J. and Sutil M.: Diseño de Soluciones Abstractas mediante el Refinamiento de Especificaciones. VII Jornadas sobre la Enseñanza Universitaria de la Informática (JENUI'01).

[3] J.M. Burgos, J.Galve y J. García. From Problems to Programs: A Pattern Language to Go from Problem Requirements to Solution Schemas in Elementary Programming. Proceedings of the 5th EuroPLoP´2000.

[4] Schmidt, D.R. Towards a classification approach to design. In *Proceedings of the Fifth International Conference on Algebraic Methodology and Software Technology*, AMAST'96 (1996), vol. LNCS 1101, Springer-Verlag, pp. 62-84.

[5] http://www.haskell.org/

[6] S. Peyton Jones and J. Hughes (editors). Standard libraries for the Haskell 98 programming language, February 1999.

[7] E.Y.Shapiro. Algorithmic Program Debugging. MIT Press, Cambridge, MA, 1983.

[8] Grupo de Investigación de Programación Declarativa Aplicada LSIIS. *Herramientas metodológicas e instrumentales para la enseñanza de la Programación*. Proyecto Fundación General de la UPM (FG-UPM-43700000190), 2000-2001. Grupo de Investigación de Programación Declarativa Aplicada LSIIS. *Herramientas*

metodológicas e instrumentales para la enseñanza de la Programación. Proyecto Fundación General de la UPM (FG-UPM-43700000190), 2000-2001.

[9] http://abraham.ls.fi.upm.es/h4d2/jenui02.pdf

Sistema de servicios web de apoyo a la docencia y gestión de una asignatura

Antonio Cañas Antonio F. Díaz Alberto Prieto
Dpto. de Arquitectura y Tecnología de Computadores
Universidad de Granada
Escuela Técnica Superior de Ingeniería Informática
e-mail: acanas@atc.ugr.es e-mail: afdiaz@atc.ugr.es e-mail: aprieto@ugr.es

Resumen

Además de las clásicas secciones estáticas que suele ofrecer la página web de una asignatura –programas, horarios, etc.–, es interesante la incorporación de servicios dinámicos que faciliten algunas tareas docentes y de gestión. Este trabajo presenta un sistema de servicios web dinámicos que puede incorporarse de manera sencilla a las páginas de varias asignaturas. El sistema integra el acceso identificado para alumnos y profesores de la asignatura, la consulta individual de calificaciones, la descarga de documentos, la ficha electrónica –incluyendo fotografía–, la orla de la clase, un foro de discusión y la autoevaluación del alumno mediante un test.

1. Introducción

La incorporación de servicios web automáticos de apoyo a la docencia y a la gestión de datos de los estudiantes es interesante ya que facilita al profesor la realización de tareas docentes y de gestión, pero sobre todo porque propicia la autonomía y el autoaprendizaje del estudiante. En los últimos años, varios grupos han desarrollando sistemas que ofrecen tales servicios, enfocados bien a la docencia en una asignatura concreta o en varias (p. ej. [1], [2], [8], [10], [11]), o bien a la gestión de todas las de una titulación o centro ([6], [7]). En el Dpto. de Arquitectura y Tecnología de Computadores de la Univ. de Granada hemos diseñado una herramienta enfocada a la docencia y gestión de cualquier asignatura, que está en constante evolución y venimos utilizando durante los tres últimos cursos en varias asignaturas de este dpto. En este trabajo se presenta su uso en *Estructura de los computadores I* y *II* [5] (Fig. 1).

2. Servicios dinámicos ofrecidos

Inicialmente, el sistema permite el acceso identificado de alumnos y profesores, que han de

Figura 1. Página de las asignaturas EC I y EC II, que incluye también los servicios estáticos de tablón de avisos, tutorías, horarios, programas, bibliografía, criterios de evaluación, FAQ y enlaces [4].

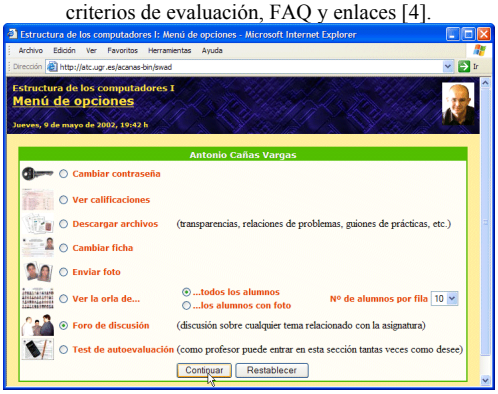


Figura 2. Página principal del sistema.

introducir su DNI y contraseña. El sistema consulta una base de datos, registra el acceso en un archivo log, y presenta la página principal (Fig. 2) con los siguientes servicios:

- *Cambio de contraseña*.
- *Consulta individual de calificaciones*. El profesor guarda las calificaciones en una tabla HTML (generada p. ej. con la hoja de cálculo Excel) que puede tener cualquier número de filas de encabezado y de pie, además de una fila para cada alumno. El número y formato de las columnas es indiferente, con la salvedad de que la primera ha de contener el DNI. En pantalla aparece el encabezado, la fila correspondiente al alumno y el pie de la tabla (Fig. 3).
- *Descarga de archivos*. Se presenta recursivamente el contenido del directorio especificado por el profesor (Fig. 4). El interés radica en restringir el acceso al ámbito de la clase, y ahorrar al profesor el trabajo de añadir un enlace a cada archivo.
- *Ficha del alumno* (común para todas las asignaturas). Se presenta un formulario inspirado en la clásica ficha en papel (Fig. 5). Tras enviarla se muestra otra pantalla similar para confirmar los datos recibidos.
- *Envío de fotografía*. Disponer de una fotografía del alumno en formato digital permite p. ej. su inclusión automática (mediante una macro) en la hoja de cálculo utilizada en la evaluación. La fotografía se envía mediante un formulario (Fig. 6) y la herramienta comprueba su formato y tamaño. En el futuro se pretende rechazar automáticamente fotografías de baja calidad.
- *Orla de la clase* (Fig. 7). Se puede seleccionar el número de columnas y la posibilidad de que aparezcan todos los alumnos o sólo los que disponen de fotografía. Bajo las fotografías aparecen el nombre y el lugar de origen. Puede accederse a la página web de cada alumno pulsando sobre él.
- *Foro de discusión*. La herramienta crea un foro para cada asignatura, estructurado en dos niveles: temas de discusión (Fig. 8), y contenido de los mensajes dentro de un tema. En ambos se muestra un formulario para

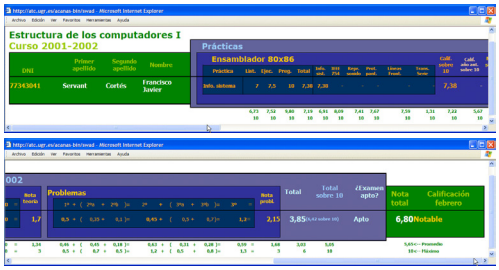


Figura 3. Calificaciones de un alumno (se muestra arriba la parte izda. y abajo la dcha.). Se incluye la nota obtenida en cada criterio evaluado en las prácticas y en cada apartado de cada pregunta del examen.

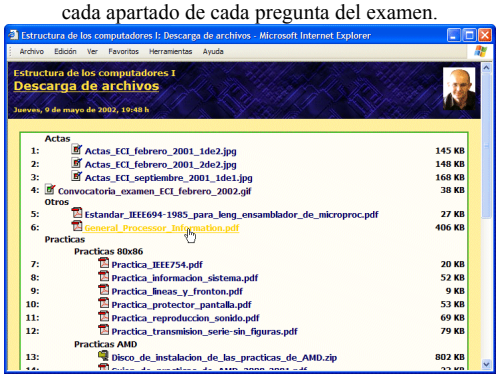


Figura 4. Descarga de archivos.

Figura 5. Ficha (datos del profesor en el ej.).

Figura 6. Envío de fotografía.

enviar un nuevo mensaje, que se firma automáticamente.

- *Test de autoevaluación*. Este servicio (el más valorado por los estudiantes) genera y evalúa un test con preguntas extraídas aleatoriamente de un archivo de texto. Las respuestas pueden ser V/F (Fig. 9), o bien una elección entre varias opciones.

Figura 7. Orla de la clase.

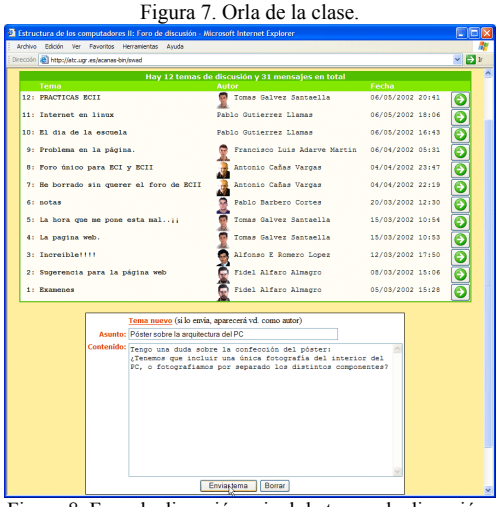


Figura 8. Foro de discusión: nivel de temas de discusión.

Figura 9. Test de autoevaluación.

3. Implementación del sistema

La herramienta es un único programa que utiliza la especificación CGI [9] en un servidor Apache [13] sobre Linux. Para facilitar la reutilización de funciones y el mantenimiento del código, integra las acciones que normalmente llevarían a cabo los siguientes 16 CGI: identificación, presentación del menú principal (Fig. 2), presentación del cambio de contraseña, actualización de contraseña, presentación de calificaciones (Fig. 3), listado de archivos (Fig. 4), presentación de ficha (Fig. 5), actualización de ficha, presentación del envío de fotografía (Fig. 6), recepción de la fotografía, presentación de la orla (Fig. 7), presentación del nivel 1 del foro (Fig. 8), presentación del nivel 2, recepción de un mensaje del foro, generación del test (Fig. 9) y evaluación del test.

Aunque el lenguaje más utilizado para programar mediante CGI es Perl, hemos optado por C, junto con un sistema de gestión de bases de datos (SGDB) PostgreSQL [12] (puede consultarse la comparación llevada a cabo en [6] entre distintos lenguajes, SGBD y herramientas para implementar servicios web dinámicos).

4. Evaluación del sistema

Se ha evaluado el interés de los alumnos en cada uno de los servicios estáticos y dinámicos, y la frecuencia de visitas a cada servicio. Las valoraciones respectivas se muestran en las Figuras 10 y 11. Además se han realizado las siguientes estadísticas (Fig. 12): el 76% de los alumnos dispone de acceso a Internet, mayoritariamente con conexión de banda ancha (Tabla 1); el 59% dispone de escáner y el 96% tiene acceso a uno; el 88% ha preferido rellenar la ficha electrónica frente al sistema tradicional aunque haya sido sólo para una única asignatura, y si la ficha electrónica fuese común para todas las asignaturas, el 97% preferiría este sistema. La Tabla 2 muestra los porcentajes medios de acceso para un estudiante desde distintos lugares.

5. Conclusión

Se ha presentado una herramienta que ofrece diversos servicios web dinámicos de apoyo a la

docencia y gestión de varias asignaturas. El sistema ha sido evaluado por los estudiantes, y se ha mostrado que los servicios dinámicos son mejor valorados en general que los estáticos y que la aplicación de las tecnologías basadas en Internet para la docencia y la gestión de fichas y fotografías de estudiantes es ya factible, al menos en las titulaciones informáticas.

Nuestra intención es seguir ampliando los servicios dinámicos para su uso en todas las asignaturas de la E. T. S. I. Inf. de Granada, centro que cuenta con unos 2000 alumnos y unas 180 asignaturas. Posiblemente, los servicios globales de gestión se incorporarán al portal web de dicha escuela [3].

Referencias

[1] M. C. Aranda y otros. *Valoración del marco docente de la informática en la Ing. Técn. Industrial.* JENUI 2001.

[2] R. Barchino y otros. *EDVI: un sistema de apoyo a la enseñanza presencial basado en Internet.* JENUI 2001.

[3] J. L. Bernier. *Portal de la E. T. S. I. Inf. de Granada.* 1996-2002. www-etsi2.ugr.es

[4] A. Cañas. *Enlaces sobre Arquitectura de computadores.* 1997-2002. atc.ugr.es/~acanas/arquitectura.html

[5] A. Cañas. *Página de estructura de los computadores I y II.* 1996-2002. atc.ugr.es/~acanas/docencia/ec

[6] D. Gayo y otros. *Desarrollo del portal web de la E. U. I. T. Inf. de Oviedo.* JENUI 2001.

[7] E. A. Martel y otros. *Sistema de gestión de asignaturas en entorno web.* JENUI 2001.

[8] R. Mas y otros. *Aplicaciones de Internet a la enseñanza: un sistema de autoevaluación.* JENUI 2001.

[9] Matt's Script Archive, Inc. *The CGI Resource Index.* 1997-2002. cgi.resourceindex.com

[10] J. J. Merelo y otros. *Integración de una asignatura en Internet: el caso de Diseño y Evaluación de Configuraciones.* JENUI 2001.

[11] R. Pérez y otros. *Incorporación de las nuevas tecnologías en la enseñanza.* JENUI 2001.

[12] PostgreSQL Global Development Group. *PostgreSQL.* 1997-2002. www.postgresql.org

[13] The Apache Software Foundation. *Apache.* 1999-2002. www.apache.org

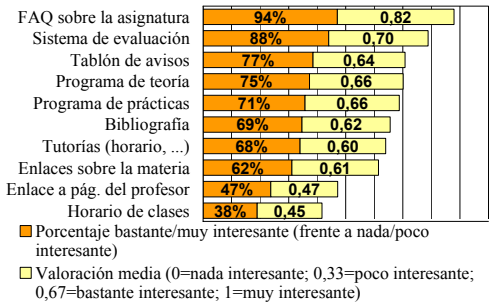


Figura 10. Valoración de los servicios estáticos.

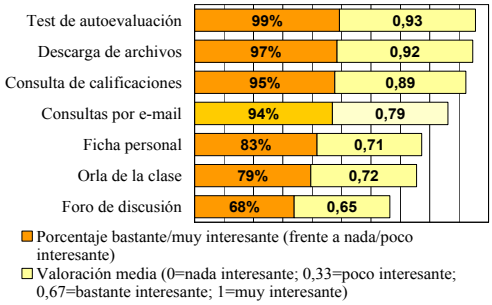


Figura 11. Valoración de los servicios dinámicos.

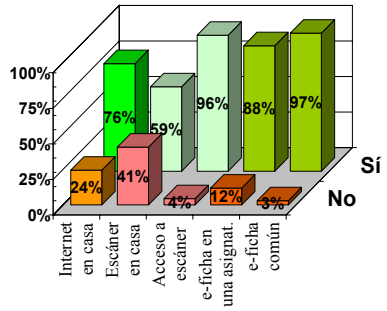


Figura 12. Otras estadísticas de la encuesta.

Módem (mayoritariamente 56 Kbps)	41%
Cable (128 - 512 Kbps)	32%
ADSL (128 - 256 Kbps)	14%
RDSI (64 - 128 Kbps)	14%

Tabla 1. Tipos de conexión a Internet desde casa.

Con Internet en casa		Sin Internet en casa	
Su casa	81%	Universidad	52%
Universidad	13%		
Col. mayor	3%	Cibercafé	22%
Otra casa	2%	Otra casa	21%
Cibercafé	1%	Col. mayor	5%

Tabla 2. Porcentajes de acceso desde distintos lugares.