ACTAS DEL SIMPOSIO NACIONAL DE DOCENCIA EN LA INFORMÁTICA (SiNDI'2005)

ORGANIZADO POR:
Asociación de Enseñantes Universitarios de la Informática (AENUI)
Congreso Español de Informática (CEDI)
Departamento de Arquitectura y Tecnología de Computadores
Universidad de Granada

ENTIDADES COLABORADORAS:
Vicerrectorado de Planificación, Calidad y Evaluación Docente UGR
Vicerrectorado de Investigación de la Universidad de Granada
Junta de Andalucía
Ministerio de Educación y Ciencia

COMITÉ DE PROGRAMA

PRESIDENTE

Joe Miró Juliá	(Universitat de les Illes Balears)

Mancia Anguita López	(Universidad de Granada)
Juan José Escribano Otero	(Universidad Europea de Madrid)
Javier Fernández Baldomero	(Universidad de Granada)
José Miró Nicolau	(Universitat de les Illes Balears)
Francisco de Sande González	(Universidad de La Laguna)

Añade el hombre conocimientos a conocimientos: nunca el saber es bastante.
Si tanto es uno más hombre cuanto más sabe, el más noble empleo será el aprender.

Padre Baltasar Gracián y Morales

Enseñar no debe parecerse a llenar una botella de agua, sino más bien a ayudar a crecer una flor a su manera.

Noam Chomsky

Presentación

Estas son las Actas del Simposio Nacional de Docencia en la Informática (SiNDI 2005), celebrado en Granada los días 15 y 16 de septiembre de 2005 en el Palacio de Exposiciones y Congresos de Granada, como parte del I Congreso Español de Informática (CEDI 2005). Desde la concepción del CEDI como multi-congreso en el ámbito de la Ingeniería Informática, se consideró ineludible la necesidad de contemplar la temática de la enseñanza universitaria de la Informática. En contacto con la Asociación de Enseñantes Universitarios de Informática (AENUI) organizadores de las Jornadas de Enseñanza Universitaria de la Informática (JENUI), se decidió incluir como parte del CEDI 2005 un muestrario de los trabajos presentados en dichas Jornadas, que sirviera también como carta de presentación de la Asociación y sus objetivos ante los asistentes al CEDI.

La preocupación por mejorar los métodos, materiales y enfoques docentes no es nueva en la universidad, como tampoco lo es la percepción de que la enseñanza universitaria está en continua crisis y constante adaptación. Pero en el caso de la informática, cuyos contenidos, técnicas y métodos están sufriendo una revolución permanente, y sobre la cual se depositan tantas expectativas de la *sociedad del conocimiento y de las TIC*, los cambios en su docencia son sin duda más rápidos y profundos que en otras disciplinas universitarias. Y sin embargo, en contraste con los centenares de congresos existentes sobre investigación en informática, existen poquísimas reuniones dedicadas a su docencia. De ahí la importancia de la AENUI como asociación y de las JENUI, y ahora el SiNDI, como congreso-reunión.

De las casi 200 ponencias de las últimas JENUI, los miembros del comité de programa hemos escogido 21 que nos parecen las más adecuadas y representativas. Todas son excelentes y hay muchas otras igualmente buenas que no hemos podido incluir. Hemos intentado que quedaran representadas las temáticas habituales de las Jornadas, tanto las genéricas de la docencia (*Métodos Pedagógicos Innovadores*, *Organización Curricular y Planes de Estudio, etc.*) como las específicas de las áreas involucradas en la misma (*Arquitectura*, *Informática Teórica*, *Ingeniería del Software*, *Inteligencia Artificial*, *Robótica*, *Sistemas Operativos, etc.*). Inevitablemente, muchas ponencias de excelente calidad han quedado excluidas. Las seleccionadas han sido sometidas a un proceso adicional de revisión y actualización por parte de los autores para adaptarlas a este simposio. Es de destacar que las 21 ponencias proceden de 16 universidades, mostrando que la preocupación por la mejora de la docencia está muy repartida por nuestras universidades. Estas 21 ponencias vienen precedidas por una ponencia institucional escrita por el Coordinador de AENUI y que sirve para enmarcar y dar el tono que esperamos tenga el Simposio.

Repasemos brevemente el contenido de estas actas. Las primeras dos sesiones de las cinco que componen el SiNDI tratan sobre *Innovación, Calidad y Evaluación Docente*. Las dos últimas sesiones están dedicadas a las temáticas específicas de las áreas, bajo los epígrafes *Arquitectura y S.O.*, y *Software, Programación y Fundamentos Teóricos*. En medio se ha incluido una sesión central titulada *Adaptación al EEES y actividades docentes relacionadas,* ya que el proceso de integración en Europa es posiblemente el factor que más nos preocupa y el que va a tener repercusiones más profundas a corto y medio plazo, tanto en los contenidos como en los métodos docentes.

Desde el comité de programa del SiNDI animamos a todos los profesores de informática interesados a ponerse en contacto con AENUI y a considerar la posibilidad de presentar alguna contribución para las próximas jornadas (JENUI 2006), a celebrar en la Universidad de Deusto (http://jenui2006.deusto.es). Damos la bienvenida a todos, deseando que sea un simposio agradable y provechoso tanto profesional como personalmente.

Granada, 13 de septiembre de 2005
Comité de Programa

CONTENIDOS

CEDI 2005

Ponencia Institucional

El placer de educar

Joe Miró Julià
Coordinador de AENUI (Asociación de Enseñantes Universitarios de la Informática)
Dept.de Matemàtiques i Informàtica
Universitat de les Illes Balears
Campus UIB
07122 Palma de Mallorca
joe.miro@uib.es

Hagamos números. ¿Cuántos artículos de tu tema de investigación has hojeado en el último año? Posiblemente centenares. ¿Cuántos has leído a fondo? Docenas, quizá. ¿Cuántos artículos de docencia has leído a fondo o simplemente hojeado? Si son unos pocos, estás por encima de la media. Y ahora seamos sinceros. ¿Qué influencia ha tenido en la sociedad tu investigación?¿A cuántos has cambiado su vida, aunque sólo sea profesionalmente? En mi caso, dudo que a nadie. En cambio sé de varios jovenes (o ya no tan jóvenes) que sí han visto cambiada su vida en nuestras aulas tanto por mis compañeros como por mí. El señor Spock, con su lógica vulcaniana, consideraría 'fascinante' que dediquemos tan poco tiempo a aquello en donde nuestra influencia es mayor, donde podemos dejar más huella.

Quizá con esto en mente, en 1994 Pedro Blesa, profesor de la Universidad Politécnica de Valencia creó una reunión de docencia que acabó convirtiéndose en las Jornadas sobre la Enseñanza Universitaria de la Informática (Jenui). Las Jenui se han convertido en un referente para la docencia de la informática en nuestras universidades. Y de esta semilla nació la Asociación de Enseñantes Universitarios de la Informática (AENUI). Somos una asociación aún joven y en crecimiento, pero creemos que llevamos a cabo una labor importante y cada vez más ambiciosa. Nuestros miembros son profesores que sienten una especial inquietud por su actividad docente y buscamos entre todos mejorar la enseñanza de la informática en nuestra universidad a través de actividades varias. Una de nuestras últimas realizaciones ha sido la organización de este simposio, que quisiera iniciar con algunas reflexiones que he ido haciendo con los años sobre por qué vale la pena dedicar tiempo a la docencia.

Vale la pena dedicarse a la docencia universitaria porque un buen profesor influye en la sociedad, en su entorno, en sus alumnos de una forma vital y única. A través del contacto con nuestros alumnos podemos hacer nuestra contribución a la mejora de nuestra sociedad mejor que con la mayoría de caminos a nuestro alcance. Estamos en una posición única que debemos aprovechar. Y no es sólo altruismo. Poco hay más satisfactorio que encontrarte con algún antiguo alumno que te agradezca la ayuda que le diste para que se convirtiera en alguien mejor.

¿Y qué es un buen profesor? Tengo una definición que descubrí a los pocos años de empezar a enseñar: El buen profesor es aquel que se preocupa por sus alumnos. Reconozco que no es muy original, y que probablemente mi inspiración provenga de la célebre cita evangélica del Buen Pastor, pero meditarla me ha ayudado a entender mejor mi labor como educador.

Esta definición mía ha dado a menudo lugar a malentendidos, tal vez convenga empezar por aclarar lo que *no* significa. Preocuparme por mis alumnos no significa ser paternalista y protector y llevarles entre algodones pa-

ra evitarles todo mal inmediato. Preocuparme por mis alumnos no significa inmiscuirme en su vida privada y querer influir en sus decisiones familiares y morales. Preocuparme por ellos en absoluto significa ser su 'colega' o bajar el nivel de exigencia. Todo esto no es. Veamos lo que sí es.

Preocuparte por tus alumnos implica que antes de enseñarles cosas prefieres educarlos. Educar proviene del latín *e-ducere*. *Ducere* significa guiar, conducir, mientras que *e* es un prefijo que significa a través de, a lo largo de. O sea que educar es guiar, conducir a tus alumnos a lo largo de su proceso de maduración y dejarlos al final en un lugar mejor, mejor situados de lo que estaban. Un educador deja a sus alumnos con más conocimientos y habilidades que ellos consideran útiles, pero sobre todo con más seguridad en sí mismos, una mejor visión del mundo. Un educador potencia a sus alumnos. Uno que simplemente enseña cosas deja a sus estudiantes con un bagaje que ellos consideran inútil y que con mucho gusto olvidan en dos semanas. En suma, acaban en el mismo lugar donde estaban.

Como principio, seguramente estamos todos de acuerdo, pero los principios, sin darles forma concreta, sirven de poco. Veamos algunas actitudes y actividades concretas que he comprobado que ayudan a mejorar mi labor de educador.

Conoce a tus alumnos por nombre. Puede parecer una tontería,pero es muy distinto dar una clase a un grupo de personas desconocidas —aunque las hayas visto mil veces y reconozcas todas las caras— que a una clase de las que te sabes los nombres de todos tus alumnos. Y bien pensado, es lógico: ¿cómo puedes preocuparte de seres anónimos (es decir, sin nombre)? Un profesor que se preocupa por sus alumnos aprende sus nombres, por muchos que sean. Curiosamente, tambien pasa la viceversa: un profesor que se sabe los nombres de sus alumnos no puede sino preocuparse por ellos. Y esta percepción no sólo la tiene el profesor, sino que también, y muy claramente, los alumnos.

No tengas prisa. Interacciona con tus alumnos. El profesor entra rápidamente en el aula, va directo a la pizarra y empieza a escribir y hablar muy rápidamente. Mira poco a sus estudiantes, tan poco que es muy difícil hacerle una pregunta, y cuando consiguen hacerla, responde rápidamente en dos frases y vuelve a la carga, a su explicación interrumpida, a rellenar la pizarra. En cuanto acaba, coge sus cosas y sale apresuradamente, a recuperar el tiempo perdido.

Probablemente conozcáis a más de uno que se acerca poco o mucho a este estereotipo. Un profesor así a veces provoca rechazo, a veces admiración —¡un señor tan importante!— pero pocas veces influye en sus alumnos. ¿Cómo ha de hacerlo, si no tiene tiempo?

No tengas prisa. Interacciona con tus alumnos. Entérate de lo que les preocupa, de lo que les mueve. Podrás enseñarles y educarles mejor si sabes qué es lo que les ilusiona y les inquieta.

Tus alumnos no son vasijas que llenar. Algunos profesores están excesivamente preocupados por la materia. Se obsesionan en comprimir enormes temarios en sus asignaturas. Es fácil identificar a estos profesores en los cambios de planes de estudio: son los que ante un menor número de créditos no cambian el temario, sino que aceleran el ritmo de sus explicaciones.

Este tipo de profesor parece que tiene la misión de asegurar la pervivencia de la materia que imparte. Sus estudiantes se convierten en simples vasijas donde preservar los sagrados contenidos para la siguiente generación. Cuando intento cuestionarles la amplitud de temario recibo una contestación de este estilo: "¿Cómo pueden salir de mi asignatura sin saber Álgebras de Lie?" Se me suelen ocurrir dos respuestas que por prudencia nunca pronuncio: (a) Por la puerta; (b) ¿Qué es un Álgebra de Lie? (Nunca falla, siempre es algo de lo que no he oído hablar en mi vida).

El objetivo de la educación no es preservar una materia ('mi' materia). No sabemos qué es lo que les pedirá la vida a nuestros alumnos y, sobre todo en informática, cualquier materia que les enseñemos probablemente la usen muy poco tiempo. Debemos potenciarlos para que puedan desenvolverse por su cuenta en

este mundo cambiante. Así durante toda su vida laboral podrán identificar la materia que necesitan ('su' materia) y aprenderla y usarla.

No hay enseñanza sin aprendizaje. Mi padre, profesor de larga experiencia y éxito, a menudo dice "Yo nunca he enseñado nada. Yo presento las cosas lo mejor posible y son mis alumnos los que aprenden". Nos arrogamos un protagonismo excesivo en el proceso de enseñanza. Hemos de dar un mayor protagonismo y responsabilidad a nuestros alumnos.

Si nos preocupamos por nuestros alumnos querremos que al final del camino sean más responsables y autosuficientes. Esto no lo vamos a conseguir si nos apropiamos de todo el protagonismo en el aula, ya que al apropiarnos del protagonismo, nos apropiamos ineludiblemente de buena parte de la actividad. Debemos hacerles cómplices del proceso de su aprendizaje y hacerles así responsables del mismo.

Y aunque no lo parezca, es un método con poco riesgo. Las ocasiones en las que he hecho caso a mis alumnos y hemos errado, han sido especialmente provechosas: aprendieron mucho más que si hubiera ejercido mi autoridad para evitar el error.

Cree en ellos. Tristemente, a menudo oímos frases como "No sé qué quieren que haga con estos alumnos. Son vagos, no se motivan, vienen muy mal preparados. En mis tiempos..." Yo no estoy de acuerdo.

Lo decimos porque siempre una generación se ha creído mejor preparada que la siguiente. Es famosa la frase de Sócrates: "Los jóvenes de hoy sólo amáis el lujo. Tenéis manías y despreciáis la autoridad. Respondéis a vuestros padres, tenéis malas costumbres y tiranizáis a vuestros maestros". Otra frase, menos famosa, pero que me gusta más se encontró en una tableta de arcilla babilónica de hace más de 3000 años: "Esta juventud actual está podrida hasta el fondo de su corazón. Los jóvenes de hoy son unos perezosos, unos malhechores que jamás serán como la juventud en otros tiempos. La juventud actual no será capaz de asegurar el mantenimiento de nuestra cultura". Comparar un recuerdo idealizado de nuestra juventud, con la realidad de esta juventud es injusto.

Lo decimos porque nosotros probablemente estábamos algo más motivados y éramos más trabajadores. Pero nosotros, los que somos profesores de universidad, éramos lo mejor de nuestras promociones, y nuestras promociones estaban compuestas por lo más selecto, los mejores estudiantes de la sociedad. Compararnos a nosotros con el alumno medio de una universidad más masificada es injusto.

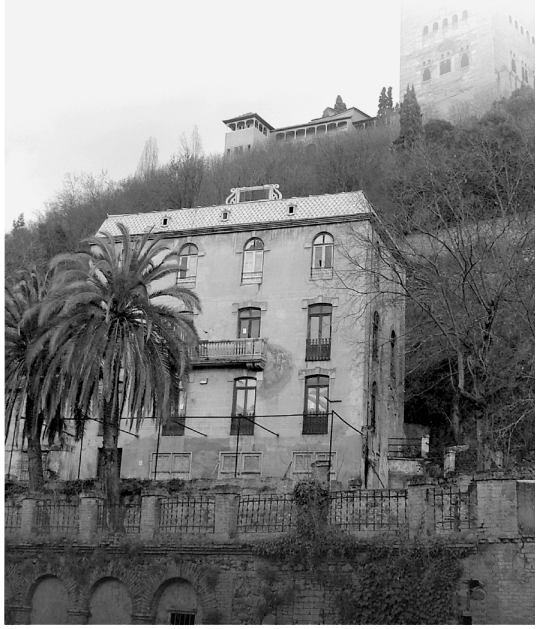
Lo decimos porque vivimos en una sociedad que cada vez vive más rápido, con cada vez más cosas que hacer. Yo tenía 5 asignaturas y 20 horas de clase a la semana, y ahora tienen 6 ó 7 asignaturas y 25 horas de clase a la semana. Y muchas más distracciones y necesidades impuestas. Exigir que nuestros alumnos se centren en las asignaturas como podíamos hacerlo nosostros es injusto.

Es posible que estéis pensando que tengo bajas expectativas para mis alumnos. No es cierto, las tengo altas, creo en ellos. Pero sé que muchas veces, a pesar de su buena voluntad, ellos no van a cumplirlas, porque no pueden. Creo en mis alumnos, y alguna vez he tenido la oportunidad de encontrarme con una clase quizá un poco mejor que la media, y, no sé cómo, he conseguido hacer sobresalir mi asignatura por encima del ruido y de las distracciones. Y esas veces mis alumnos han sido notablemente trabajadores, inteligentes, ilusionados, atentos, con iniciativa. Yo creo en mis alumnos y a veces consigo grandes resultados. Resultados imposibles si no creyera en ellos.

En resumen, conociéndoles, tomándote el tiempo de interaccionar con ellos, no obsesionándote con tu materia, haciéndoles responsables de su propia educación y creyendo en ellos, es como te preocupas por tus alumnos. Si haces esto influirás decisivamente —y positivamente— en su vida y, a través de ellos, en tu entorno y en tu sociedad. Te darás cuenta de la primera razón por la que es un gozo dedicarse a la docencia. Y también te darás cuenta de la segunda razón: ellos influirán en ti.

CEDI 2005

Innovación, Calidad y Evaluación Docente I



Relación entre el rendimiento de dos asignaturas de segundo curso y las asignaturas de primer curso en Ingenierías Técnicas de Informática de la UPV

Luisa R. Zúnica, Rosa Alcover
Dept. de Estadística e Investigación Operativa Aplicadas y Calidad
Facultad de Informática
Universidad Politécnica de Valencia
46022 Valencia
{lrzunica,ralcover}@eio.upv.es

Jorge Más
Dept. de Física Aplicada
Escuela Técnica Superior de Informática Aplicada
Universidad Politécnica de Valencia
46022 Valencia
jmas@fis.upv.es

José M. Valiente, José V. Benlloch
Dept. de Informática de Sistemas y Computadores
Facultad de Informática, Esc. Técnica Sup. de Informática Aplicada
Universidad Politécnica de Valencia
46022 Valencia
{jvalient,jbenlloc}@disca.upv.es

Pedro Blesa
Dept. de Sistemas Informáticos y Computación
Escuela Técnica Superior de Informática Aplicada
Universidad Politécnica de Valencia
46022 Valencia
pblesa@dsic.upv.es

Resumen

En un trabajo anterior, los autores estudiaron la relación entre el rendimiento de una asignatura tipo, de segundo curso, con respecto al rendimiento de las asignaturas del curso anterior, con el fin de detectar si existía una relación más fuerte con la/s asignatura/s posibles prerrequisitos. No se detectó que este efecto fuera significativo, pero en el análisis estadístico realizado se encontraron dos factores relevantes, que se pueden interpretar como el rendimiento global del alumno, y su "destreza" en asignaturas propiamente informáticas. En este trabajo se ha extendido el estudio, considerando dos asignaturas de segundo curso y ampliando los alumnos investigados a las dos ingenierías técnicas de informática. La estructura factorial obtenida es, en parte, similar a la hallada en nuestro anterior trabajo, pero han aparecido nuevos factores claramente interpretables y con efectos significativos sobre las asignaturas estudiadas.

1. Motivación y objetivos

Como se decía en nuestro anterior trabajo [1], han proliferado en la universidad española diversos estudios sobre rendimiento académico, consecuencia de la creciente percepción que tenemos los docentes de que "algo" no funciona en la universidad. El recurso fácil de achacar todas las culpas al alumnado se entremezcla con otras voces más autocríticas que ven deficiencias tanto en los centros como en el profesorado y en los planes de estudio vigentes. Consecuencia, en buena medida, de estas preocupaciones, ha sido la contrarreforma de los planes de estudio. Ya con la idea de acometer esta reforma con cierto conocimiento de causa, el Consejo de Universidades recomendó la realización de este tipo de estudios para evaluar los planes de estudio vigentes [2].

En particular, en la Universidad Politécnica de Valencia (UPV) se puso en marcha, hace ya una docena de años, el Proyecto de Innovación Educativa (PIE), con la intención de transformar las técnicas docentes. Acogidos a este proyecto, un grupo de profesores de la UPV relacionados con la Escuela Técnica Superior de Informática Aplicada o ETSIA (antigua Escuela Universitaria de Informática) iniciamos un proyecto [3] con el fin de elaborar una herramienta informática que permitiera extraer, de manera cómoda, sencilla, y sin necesidad de conocimientos informáticos avanzados, datos de rendimiento académico de la abundante base de datos de que dispone la universidad [4].

Mediante esta aplicación se han elaborado algunos estudios relativos a diferentes aspectos que pensamos tienen influencia sobre el rendimiento académico, factores tales como la nota de acceso de los alumnos a la universidad,

procedencia de los alumnos (COU, FP, etc.), organización de los estudios y otros [5,6]. En la mayoría de ellos hemos estudiado, de manera especial, la problemática existente en primer curso, que nos parece especialmente grave. Pero también tenemos la sensación de que en segundo y tercer curso, donde el fracaso académico también es elevado, la problemática no es la misma que en primero, y pueden afectar factores distintos y de manera diferente.

Basándonos en el anterior trabajo, en éste queremos ampliar la visión del problema y hemos escogido dos asignaturas similares, Sistemas Operativos I (SO1) y Sistemas Operativos II (SO2), ambas en segundo curso, pero en semestres consecutivos, y que son comunes a las titulaciones de Ingeniero Técnico en Informática de Gestión (ITIG) e Ingeniero Técnico en Informática de Sistemas (ITIS), en la ETSIA de la UPV. Nuestro objetivo es ver si los factores encontrados anteriormente se vuelven a reproducir, y si se encuentran diferencias entre las dos asignaturas. Las asignaturas son del plan de 1993, no pudiéndose hacer estudios del plan vigente (2001), ya que hemos utilizado una historia de cinco años.

2. Estudio realizado

El primer problema que se presenta al intentar hacer estudios de rendimiento académico es el de definir de forma clara qué es el rendimiento académico. En este sentido, y a partir de otros estudios sobre la cuestión [7], definimos el rendimiento académico de un alumno en una asignatura en un curso, como un parámetro comprendido entre 0 y 100 en el que se tiene en cuenta la calificación obtenida, la convocatoria en la que se aprobó la asignatura, y los años transcurridos desde la primera matriculación del alumno en la asignatura, según la fórmula:

$$R_{ij} = 0.4^{n_{ij}-1} k_{ij} * 40 \qquad (1)$$

donde R_{ij} es el rendimiento del alumno i en la asignatura j, n_{ij} es el año en el que el alumno i aprobó la asignatura j (referido al año en el que el alumno se matriculó en la asignatura j por primera vez), y k_{ij} es un coeficiente que depende tanto de la calificación obtenida por el alumno i en la asignatura j, como de la convocatoria en la que se aprobó, pudiendo tomar los valores siguientes:

Calificación	Convocat. ordinaria	Convocat. extraordinaria
<5	0	0
≥5 y <7	1,5	1
≥7 y <9	2	1,5
≥9	2,5	2

A partir de esta definición, se pueden definir rendimientos referidos a distintos ámbitos; para este estudio hemos utilizado el rendimiento de una asignatura j durante un período de tiempo P, entendido como el promedio de los rendimientos, en la asignatura j, de todos los alumnos que durante el período P estuvieron matriculados en dicha asignatura:

$$R_j(P) = \frac{\sum_{i \in j(P)} 0.4^{n_{ij}-1} k_{ij}}{\sum_{i \in j(P)} i} * 40 \qquad (2)$$

El segundo problema que se nos presenta para llevar a cabo el estudio pretendido, es el de definir de forma clara la población a estudiar. En el caso de estudios sobre primer curso es generalizada la técnica de elegir una cohorte de alumnos que ingresan en una titulación un curso determinado. Pero el hacer un estudio sobre segundo curso plantea el problema de definir quiénes son alumnos de segundo curso, máxime en un centro donde no existen incompatibilidades de matrícula entre asignaturas, y es común que muchos alumnos cursen, de forma simultánea, asignaturas de varios cursos.

Por ello, el procedimiento seguido fue el siguiente:

- Elegimos dos asignaturas de segundo curso, una de primero y otra de segundo semestre, ambas obligatorias para las dos titulaciones que se imparten en la Escuela, que nos permitieran abarcar una mayoría amplia de alumnos y situaciones. Dichas asignaturas fueron SO1 y SO2.
- Seleccionamos todos aquellos alumnos que durante el curso 2000-2001

estuvieron matriculados en dichas asignaturas, calculando el rendimiento obtenido por cada alumno en dicho curso y en cada asignatura.

- Buscamos, en los cinco cursos anteriores al 2000-2001, para cada uno de los alumnos señalados, el rendimiento obtenido en cada una de las asignaturas de primer curso (0 en caso de que la asignatura no hubiera sido aprobada o su valor, en caso contrario).
- Como se explicará en la siguiente sección, realizamos un estudio estadístico con toda la información obtenida a excepción de la correspondiente a las asignaturas que tienen carácter de obligatorias en una titulación, pero son optativas en la otra: SIO, ADO, AM2, ES1 y FFI (en cursiva en las tablas 1 y 2).
- De este modo, el total de alumnos estudiados, considerando las titulaciones de ITIG e ITIS, fue de 577 para SO1 y de 675 para SO2.

Tabla 1. Asignaturas obligatorias de primer curso del título ITIG

Sem	Cód	Asignatura	Cr
1A	AD1	Algoritmos y Estructuras de Datos I	6
1A	AM1	Análisis Matemático I	6
1A	FCO	Fundamentos de Computadores	9
1A	IPR	Introducción a la Programación	3
1A	MAD	Matemática Discreta	6
1A	*SIO*	*Sistemas de Información de las Organizaciones*	6
1B	*ADO*	*Administración de Organizaciones y Sistemas de Información*	6
1B	ALG	Álgebra	6
1B	AD2	Algoritmos y Estructuras de Datos II	6
1B	EC1	Estructura de Computadores I	6
1B	INT	Inglés Técnico	6

Tabla 2. Asignaturas obligatorias de primer curso del título ITIS

Sem	Cód	Asignatura	Cr
1A	ALG	Álgebra	6
1A	AD1	Algoritmos y Estructuras de Datos I	6
1A	AM1	Análisis Matemático I	6
1A	FCO	Fundamentos de Computadores	9
1A	IPR	Introducción a la Programación	3
1A	MAD	Matemática Discreta	6
1B	AD2	Algoritmos y Estructuras de Datos II	6
1B	*AM2*	*Análisis Matemático II*	6
1B	*ES1*	*Estadística I*	6
1B	EC1	Estructura de Computadores I	6
1B	*FFI*	*Fundamentos Físicos de la Informática*	9
1B	INT	Inglés Técnico	6

3. Análisis Estadístico

3.1. Metodología

En principio, un primer análisis que podría responder a nuestro objetivo consistiría en plantear directamente un modelo de regresión lineal múltiple que ligara, el rendimiento obtenido por los alumnos en la asignatura Sistemas Operativos I (RSO1), con los correspondientes a las asignaturas de primer curso (variables explicativas), y de forma similar con la asignatura Sistemas Operativos II. Sin embargo, este análisis presenta el problema de que las variables explicativas están muy relacionadas entre sí (por ejemplo, los alumnos más brillantes en general obtienen buenos rendimientos en la mayor parte de las asignaturas). En estas situaciones puede resultar estadísticamente imposible diferenciar, a partir de los datos, los efectos de las diferentes variables, pudiendo incluso resultar que ninguna de ellas aparezca como estadísticamente significativa, pese a que su efecto conjunto sí que lo es claramente.

En consecuencia, se ha optado por plantear, en primer lugar, un Análisis de Componentes Principales (ACP) [8,9] sobre las variables correspondientes a los rendimientos de las asignaturas de primer curso en relación a la asignatura SO1. Con el fin de que no se reduzca el número de casos a analizar sólo se han considerado las asignaturas comunes para las dos

titulaciones, resultando un total de 9 asignaturas. El objetivo de este primer análisis es el de condensar o resumir la información contenida en estas 9 variables en un conjunto reducido de nuevas variables, no observables directamente e incorrelacionadas entre sí. A estas nuevas variables se les denomina componentes o factores. Una vez obtenidos estos factores, se ha planteado un modelo de regresión tomando RSO1 como variable dependiente y las nuevas variables (factores), obtenidas en el análisis previo, como variables explicativas o independientes. El mismo procedimiento se ha seguido para la asignatura SO2, esto es, primero un ACP y, tras obtener los factores, se plantea un modelo de regresión sobre los mismos considerando el rendimiento obtenido en la asignatura SO2 (RSO2) como variable dependiente.

Los ajustes de regresión se han llevado a cabo mediante la operativa "stepwise", de forma que sólo se introducen en el modelo resultante aquellas variables explicativas cuyo efecto resulta estadísticamente significativo. Esta técnica es especialmente adecuada cuando, como en el caso que nos ocupa, las variables explicativas están incorrelacionadas. Para el análisis de datos se ha utilizado el programa *Statgraphics* [10]. En los siguientes apartados se exponen los resultados obtenidos para ambas asignaturas.

3.2. Resultados para Sistemas Operativos I

La tabla 3 recoge los resultados del ACP, apreciándose que los cinco primeros factores explican más del 80% de la variabilidad total generada por los rendimientos en los alumnos de las nueve asignaturas de primer curso.

Tabla 3. ACP de las variables de primer curso relacionadas con SO1

Factor Number	Eigenval.	Percent. Variance	Cumulat Variance
1	3,37642	37,516	37,516
2	1,54822	17,202	54,718
3	0,961041	10,678	65,396
4	0,732391	8,138	73,534
5	0,645434	7,171	80,706
6	0,499232	5,547	86,253
7	0,475552	5,284	91,537
8	0,447568	4,973	96,509
9	0,314147	3,491	100,000

En la tabla 4 se recogen las correlaciones de los rendimientos de las nueve asignaturas estudiadas con las cinco primeras componentes obtenidas:

Tabla 4. Correlaciones de los factores con las nueve variables estudiadas

	Factor 1	Factor 2	Factor 3	Factor 4	Factor 5
RAD1	0,567253	-0,679390	0,14038	-0,071764	0,086728
RAD2	0,633882	-0,350387	-0,07610	-0,416459	-0,430482
RALG	0,669291	0,409814	0,01136	-0,203203	0,184302
RAM1	0,681861	0,360793	-0,12404	-0,189870	0,179656
REC1	0,726279	0,120046	-0,13643	0,196405	-0,429986
RFCO	0,650418	0,023542	-0,22401	0,633423	-0,021208
RINT	0,225784	0,286736	0,90318	0,113522	-0,164448
RIPR	0,514203	-0,655258	0,18778	0,116139	0,340295
RMAD	0,688506	0,373115	-0,01467	-0,101414	0,241188

Se constata en la tabla que para el primer factor todas las correlaciones con las diferentes asignaturas son positivas y del mismo orden de magnitud (en torno a 0.5-0.7), salvo el coeficiente de la asignatura Inglés Técnico (RINT) que es del orden de 0.22. Esta primera componente estaría indicando que los alumnos que son buenos estudiantes obtienen en general buenos rendimientos en todas las asignaturas. Por sus características especiales la asignatura Inglés Técnico (INT) se diferencia de las demás puesto que los alumnos pueden llegar a la ETSIA con diferentes niveles de inglés, y esto no está necesariamente relacionado con el rendimiento del alumno en otras materias. En consecuencia, el valor de este factor en cada alumno puede considerarse que mide el nivel general de este.

Es importante señalar que como los restantes factores están incorrelacionados con este primero, no medirán ya aspectos relacionados con el nivel general del alumno, sino otros relativos a su

mayor o menor predisposición hacia ciertos grupos de asignaturas.

Respecto al segundo factor, se observan coeficientes tanto positivos como negativos. Teniendo en cuenta los coeficientes mayores en valor absoluto, y para un mismo nivel promedio global recogido por la primera componente, se pueden diferenciar dos grupos de estudiantes: aquellos que presentan, en primer curso, mayor destreza en las asignaturas relacionadas con la programación: AD1, IPR, AD2 (valores negativos del factor) y aquellos alumnos que la presentan en las asignaturas matemáticas como pueden ser ALG, MAD y AM1 (valores positivos del factor).

El tercer factor claramente está midiendo el nivel de inglés del alumno, al presentar una correlación positiva muy alta (0.90) con esta asignatura y casi nula con las restantes. La existencia de este factor, pone también de manifiesto que el conocimiento de inglés está muy poco relacionado con el rendimiento académico de los alumnos en las restantes asignaturas de primero.

La cuarta componente diferencia a alumnos con buen rendimiento en FCO en relación al que tienen en AD2. Este factor no ha resultado significativo en el posterior análisis de regresión.

Por ultimo, los alumnos con valores negativos en el quinto factor tienen rendimientos relativamente mejores en asignaturas del segundo semestre que en las del primero, caracterizando por tanto a alumnos que han mejorado su rendimiento a lo largo del primer curso, mientras que los alumnos con valores positivos de este factor se caracterizan por haber ido empeorando a lo largo de dicho curso.

De acuerdo a la metodología expuesta en 3.1. y una vez obtenidas las componentes principales o factores, el siguiente paso ha consistido en plantear un modelo de regresión que permita analizar el efecto de estas sobre el promedio del rendimiento en la asignatura de segundo curso SO1. El modelo ha sido el siguiente: $E(RSO1) = \beta_0 + \beta_1 * FACTOR1 + \beta_2 * FACTOR2 + \beta_3 * FACTOR3 + \beta_4 * FACTOR4 + \beta_5 * FACTOR5$, donde los parámetros β_i recogen y cuantifican el efecto de los factores correspondientes ($FACTOR_i$) sobre el rendimiento medio en SO1. Los resultados obtenidos en la estimación *stepwise* del modelo, reteniendo sólo los factores cuyo efecto ha resultado estadísticamente significativo (para un riesgo de 1ª especie $\alpha = 0.05$) se muestran en la tabla 5.

Tabla 5. Modelo de regresión para RSO1

Dependent variable: RSO1

Parameter	Estimate	Standard Error	T Statist.	P-Value
CONSTANT	9,451	0,71	13,23	0,0000
FACTOR1	5,311	0,39	4,09	0,0000
FACTOR2	-1,177	0,57	-2,05	0,0410
FACTOR5	-4.458	0,89	-5,01	0,0000

El análisis pone de manifiesto que los efectos de FACTOR1, FACTOR2 y FACTOR5 sobre el rendimiento medio en SO1 son estadísticamente significativos (valores en la columna *P-value* inferiores a 0.05).

Se constata, en primer lugar, un efecto muy significativo y positivo del Factor 1, que lógicamente se explica por el hecho de que los alumnos con mejor nivel global en primer curso obtienen, en general, mejores calificaciones en SO1 que los alumnos con peor nivel global.

También ha resultado muy significativo estadísticamente el efecto del FACTOR5. Teniendo en cuenta la interpretación de este factor vista anteriormente, el signo negativo del coeficiente β_5, indica que los alumnos que mejoraron su rendimiento general en el segundo semestre del primer curso respecto al que tuvieron en el primer semestre, obtienen generalmente mejores resultados en esta asignatura SO1 del segundo curso que aquellos que empeoraron en el segundo semestre respecto al primero.

También ha resultado significativo el efecto del FACTOR2, si bien el nivel de significación de dicho efecto ha sido sensiblemente menos marcado que el de los otros dos factores anteriormente reseñados. El signo negativo del efecto de este factor, indica un mejor rendimiento en SO1 para aquellos alumnos que en primer curso tuvieron relativamente mejores resultados en las asignaturas de programación que en las puramente matemáticas.

Destaquemos, por último, que el nivel de inglés de los alumnos, aspecto recogido por el FACTOR3, no ha resultado relacionado

significativamente con el rendimiento académico en SO1.

3.3. Resultados para Sistemas Operativos II

El mismo procedimiento se ha seguido para analizar el rendimiento de la asignatura SO2, asignatura de 2º curso y 4º semestre.

La tabla 6 recoge los resultados del ACP realizado sobre estas nueve variables. Las cinco primeras componentes explican también en este caso más del 80% de la variabilidad total. Estos factores serán los únicos retenidos en el resto del análisis.

Tabla 6. ACP de las variables de primer curso relacionadas con SO2

Factor Number	Eigenval.	Percent. Variance	Cumulat. Variance
1	3,22914	35,879	35,879
2	1,67697	18,633	54,512
3	0,96728	10,748	65,260
4	0,72962	8,107	73,367
5	0,66997	7,444	80,811
6	0,51062	5,674	86,484
7	0,47994	5,333	91,817
8	0,44332	4,926	96,743
9	0,29315	3,257	100,000

Para interpretar estos factores se recogen a continuación, en la Tabla 7, sus correlaciones con las nueve variables estudiadas.

Tabla 7. Correlaciones de los factores con las nueve variables estudiadas

	Factor 1	Factor 2	Factor 3	Factor 4	Factor 5
RAD1	0,545661	-0,706632	0,152462	-0,076484	0,079798
RAD2	0,616160	-0,402082	-0,063060	-0,136271	-0,562949
RALG	0,630512	0,459040	0,007458	-0,345566	028793
RAM1	0,689899	0,372925	-0,034984	-0,242866	0,089258
REC1	0,678453	0,103011	-0,300992	0,391314	-0,304383
RFCO	0,662613	0,043062	-0,256791	0,491967	0,340863
RINT	0,235251	0,302816	0,854866	0,311517	-0,143661
RIPR	0,519286	-0,648307	0,221315	-0,074987	0,308264
RMAD	0,673705	0,376446	0,049615	-0,170225	0,115718

Se constata que el primer factor, con correlaciones claramente positivas y bastante elevadas con todas las variables, excepto con INT, tiene una interpretación análoga a la del obtenido en el estudio anterior, como un indicador del nivel general del alumno.

El segundo factor es también muy similar al obtenido en el apartado 3.2., en el sentido que los alumnos con valores negativos del mismo muestran un mejor rendimiento en las asignaturas de programación (como AD1, IPR o AD2) que en las de carácter más matemático (como ALG, MAD y AM1).

El tercer factor, al igual que sucedía en el análisis anterior, está claramente asociado al nivel del alumno en INT, resultando poco relacionado con el resto de las variables.

Sin embargo, el cuarto factor resulta diferente al encontrado en el análisis del apartado 3.2. Los alumnos con valores positivos del mismo se caracterizan especialmente por buenos resultados en FCO y EC1, en relación a los que obtienen en materias como ALG y AM1.

El quinto factor es también análogo al obtenido en el análisis anterior, en el sentido que los alumnos con valores negativos del mismo han obtenido mejores calificaciones en materias del segundo semestre del primer curso (como AD2 o EC1) que en las del primer semestre (como FCO, IPR o MAD).

De nuevo, siguiendo la metodología expuesta en 3.1., se ha planteado un modelo de regresión para analizar el efecto de estos cinco factores sobre el promedio del rendimiento en SO2. El modelo ha sido el siguiente: $E(RSO2) = \beta_0 + \beta_1 * FACTOR1 + \beta_2 * FACTOR2 + \beta_3 * FACTOR3 + \beta_4 * FACTOR4 + \beta_5 * FACTOR5$. Los resultados obtenidos en la estimación *stepwise* del modelo se muestran en la tabla siguiente, en la que se

reflejan únicamente los factores cuyos efectos han resultado estadísticamente significativos:

Tabla 8. Modelo de regresión para RSO2

Dependent variable: RSO2				
Parameter	Estimate	Standard Error	T Statist.	P-Value
CONSTANT	9,193	0,67	13,63	0,0000
FACTOR1	3,283	0,38	8,74	0,0000
FACTOR2	2,450	0,52	4,70	0,0000
FACTOR4	2,695	0,79	3,41	0,0006
FACTOR5	-2,791	0,82	-3,39	0,0007

Se constata que, análogamente a lo que sucedía en el modelo para RSO1, son muy significativos los efectos de las componentes 1 y 5, positivo el primero y negativo el último. La interpretación de estos efectos es la misma que la que se expuso en aquel caso.

También ha resultado en este caso muy significativo el efecto de la segunda componente. Sin embargo, el signo obtenido para dicho efecto ha sido el contrario al hallado en el análisis para SO1, poniendo de manifiesto que en la asignatura de SO2 los mejores resultados los obtuvieron los alumnos que en primer curso destacaron más en las asignaturas matemáticas que en las de programación.

Por su parte, la cuarta componente tiene también un efecto significativo sobre el rendimiento académico en SO2, a diferencia de lo que sucedió para la asignatura de SO1, indicando que los alumnos con buenos resultados en asignaturas sobre Estructura de Computadores (FCO y EC1) también obtienen buenos rendimientos en SO2.

La segunda componente, para el caso de SO1, y el conjunto de las componentes 2 y 4 para el caso de SO2, vienen a representar la influencia 'temática' entre estas asignaturas y las del primer curso. Así por ejemplo, en el caso de SO1 la segunda componente positiva expresaba la estrecha relación entre esta asignatura y las correspondientes del área de programación. Esto se explica por los contenidos de SO1, relacionados con la temática de procesos e hilos de ejecución (creación, planificación, sincronización, comunicación e interbloqueos) que requieren una sólida formación previa en cuestiones de algorítmica y de programación.

En cambio, los contenidos temáticos de SO2 están más relacionados con los conceptos de memoria virtual, la entrada/salida y los sistemas de ficheros, estudiados a diferentes niveles de abstracción. El estudio previo de los sistemas de memoria del computador (memoria central, memoria cache y virtual) a nivel físico, así como de la gestión de la entrada/salida a bajo nivel, se realiza en las asignaturas de Estructura de Computadores (FCO y EC1), relacionadas con la cuarta componente. De ahí la influencia de este factor en RSO2. Asimismo, todos estos contenidos suelen requerir el uso de un cierto aparato matemático, con menos requerimientos en la parte algorítmica y de programación. De ahí la influencia de la componente 2, con signo positivo, en la asignatura SO2.

Destaquemos, por último, que el nivel de INT de los alumnos, aspecto recogido por la tercera componente, tampoco ha resultado relacionado significativamente con el rendimiento académico en SO2.

4. Conclusiones

Los resultados del estudio estadístico realizado para las asignaturas de segundo curso SO1 y SO2 permiten alcanzar ciertas conclusiones:

- Hay siempre un factor de aprovechamiento global (primera componente) que expresa que los alumnos con un buen rendimiento en todas, o la mayoría, de las asignaturas de un curso, suelen mantener también un alto rendimiento en las de cursos sucesivos. Tal y como obtuvimos en un estudio anterior [11], un buen predictor de este factor de rendimiento global es la nota de acceso del alumno a la Universidad.
- Hay también un factor temático (componentes 2 y 4) que pone de manifiesto que los contenidos de cada asignatura (SO1 y SO2, en nuestro caso) suelen estar relacionados con contenidos previos de cursos anteriores. Un buen nivel en estos contenidos previos suele llevar a un mejor aprovechamiento de la materia en cuestión. La diferencia en la naturaleza de los contenidos de SO2 respecto a SO1, se refleja en el hecho, aparentemente sorprendente, de que el efecto muy significativo sobre ambas del factor 2 tenga signo contrario en un caso que en el otro.
- Se constata la escasa influencia del nivel de INT en el aprovechamiento de las asignaturas SO1

y SO2. Probablemente esto no es debido a que no sea necesario el estudio del inglés, sino a la existencia cada vez mayor de apuntes y libros en castellano, que hace que los alumnos tengan que recurrir cada vez menos a la lectura de textos originales en inglés para el seguimiento de las asignaturas.

- Por último, hay un factor estacional (quinta componente) que muestra la predisposición de los alumnos a obtener buenos resultados en asignaturas de segundo curso si en el segundo semestre del curso anterior mejoraron su rendimiento en relación al primer semestre. Da la impresión de ser más un factor de evolución temporal de los alumnos, que van mejorando rendimientos progresivamente, que una cuestión ligada a la semestralidad de las asignaturas, dado que el efecto de dicho factor tiene idéntico signo para SO1 y SO2, que se hallan en distintos semestres. La división de las asignaturas en semestres, propiciada por los últimos planes de estudios, ha introducido un factor de distinción, en cuanto a rendimiento académico, entre primer y segundo semestre, que no se constata en este estudio.

Algunas de estas conclusiones pueden parecer obvias o de sentido común, y coinciden con las opiniones expresadas de manera intuitiva por el personal docente de estas áreas. Lo interesante de este estudio es que se expresa, de forma cuantitativa, la influencia de todos esos factores que los docentes siempre hemos considerado de manera cualitativa, y también despeja las dudas acerca de la no influencia de otros.

Un trabajo futuro interesante sería el estudio de las posibles influencias horizontales (intra-cursos) entre las materias de un mismo curso.

Agradecimientos

A Jesús Lafuente por su colaboración en la realización de la herramienta informática; al Instituto de Ciencias de la Educación y a la Facultad de Informática por su apoyo financiero, y a la ETS de Informática Aplicada por facilitarnos los datos estudiados.

Referencias

[1] Zúnica, L., Blesa, P. y otros. Estudio del rendimiento académico de asignaturas con relación a asignaturas del curso anterior. Actas de IX Jornadas de Enseñanza Universitaria de Informática (Jenui 2003). Cádiz. 2003.

[2] BOE nº 15 de 17 de enero de 1997, pag 1895: "Recomendaciones del Consejo de Universidades a las Universidades en relación con la organización de los planes de estudio"..

[3] PID 9052 de la UPV. Libro de resúmenes de los PID de la UPV. 2001.

[4] Más, J., Alcover, R. 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.

[5] 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.

[6] 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.

[7] González, R. M. Rendimiento académico en la UPM: estudio longitudinal en primer ciclo, Vol. 1 y 2. ICE de la UPM, 1993.

[8] Mardia, K., Kent, J. Y Bibby, J. (1979). Multivariate Analysis. Ed. Academic Press, London

[9] Sharma, S. (1996). Applied Multivariate Techniques. Ed Wiley

[10] Statgraphics Plus v.5.0. Manugistics Inc. (2000).

[11] Más, J., Valiente, J.M. y otros. Estudio de la influencia sobre el rendimiento académico de la nota de acceso y procedencia (COU/FP) en la EU de Informática. Actas de VIII Jornadas de Enseñanza Universitaria de Informática (Jenui 2002). Cáceres. 2002.

Objetivos formativos del primer curso de las ingenierías informáticas y estrategias docentes relacionadas

Fermín Sánchez Carracedo
Dept. de Arquitectura de Computadors
Universitat Politècnica de Catalunya
Campus Nord, c/Jordi Girona 1-3, Mòdul D6
08034 Barcelona
Tel: 93 4017234, Fax: 93 4017055
e-mail: fermin@ac.upc.edu

Ricard Gavaldà Mestre
Dept. de Llenguatges i Sistemes Informàtics
Universitat Politècnica de Catalunya
Campus Nord, c/Jordi Girona 1-3, Edifici Omega
08034 Barcelona
Tel: 93 4017294, Fax: 93 4017014
e-mail: gavalda@lsi.upc.edu

Resumen

En este artículo se expone una propuesta de objetivos formativos para el primer curso de las titulaciones universitarias de informática y se define un conjunto de estrategias docentes para conseguirlos. Los objetivos formativos se han clasificado en tres grupos: (i) *contenidos técnicos*, (ii) *capacidades y aptitudes* y (iii) *actitudes, valores y normas*. El artículo se centra en los objetivos relativos a capacidades y aptitudes, en particular en aquéllos que pueden comenzar a conseguirse ya desde el primer curso, y describe algunas estrategias docentes que contribuyen a alcanzar dichos objetivos.

1. Introducción

La universidad española está tomando conciencia de una verdad aceptada desde hace ya tiempo en otros niveles del sistema educativo: el protagonista del proceso de aprendizaje es el alumno y la tarea del profesor es facilitar este proceso, no ser su centro. Así, los *objetivos formativos* de un proceso de aprendizaje deben expresar claramente qué debe haber aprendido el alumno al terminar dicho proceso. La tarea del profesor (y de la universidad) es poner los medios para que el alumno alcance estos objetivos.

La definición precisa de objetivos formativos es un paso crucial en todo proceso de aprendizaje, sea cual sea su ámbito (una clase, una asignatura, un curso o una titulación entera).

En este artículo se describe el trabajo realizado por un grupo de profesores de la Facultat d'Informàtica de Barcelona (FIB) de la Universitat Politècnica de Catalunya (UPC) para definir los objetivos formativos del primer curso de la Ingeniería en Informática y las Ingenierías Técnicas en Informática. Consideramos que estos objetivos son también válidos para el primer curso de la ingeniería informática en la futura titulación de Grado. El artículo se centra en los objetivos relacionados con la adquisición de capacidades y aptitudes, más que en los relacionados con la adquisición de conocimientos técnicos.

Algunas asignaturas del primer curso de la FIB habían definido anteriormente sus propios objetivos formativos, pero nunca se habían definido objetivos globales para el curso completo. De hecho, creemos que es una iniciativa pionera en este sentido.

Tener una definición conjunta de objetivos para el primer curso es interesante porque:

- El primer curso universitario representa un cambio importante para los estudiantes, tanto en el grado de exigencia como en el método de trabajo. Es fundamental que el estudiante sepa qué se espera de él.
- En el primer curso se ponen las bases, tanto a nivel conceptual como procedimental, para asimilar el resto de la carrera. Las distintas asignaturas deben trabajar con los contenidos y procedimientos de una manera coherente.
- Los objetivos del curso son algo más que la suma de los objetivos de las asignaturas. Para tener una coherencia global, lo lógico es diseñar primero los objetivos del curso y, a partir de ellos, diseñar los propios de cada asignatura (y no al revés, cómo suele hacerse).

Este artículo se basa en un trabajo desarrollado por los profesores responsables de las nueve asignaturas de primer curso de la FIB (entre los cuales se cuenta el primer autor) y el jefe de estudios del primer curso (el segundo autor) desde noviembre de 2000 hasta julio de 2002. El documento resultado del trabajo puede encontrase

en [1]. Para elaborarlo, se contó con la experiencia docente en primer curso de los propios autores, con las opiniones recogidas –explícitamente- de otros profesores (de primer curso y de cursos superiores) y con distintos documentos en que se definen objetivos formativos para carreras afines a la ingeniería informática [2,3].

El resto del artículo se organiza de la siguiente forma: en la Sección 2 se exponen las ventajas de tener objetivos formativos bien formulados en cualquier proceso de aprendizaje y se describe la taxonomía de Bloom para clasificar los distintos niveles de competencia; en la Sección 3 se clasifican los objetivos en tres grupos, se explica brevemente cada grupo y se detallan los objetivos de dos de los grupos (*capacidades y aptitudes* y *actitudes, valores y normas*), tanto para el primer curso como para toda la ingeniería informática; la Sección 4 explica algunas estrategias docentes para empezar a conseguir, ya en el primer curso, los objetivos de tipo *capacidades y aptitudes*, y la Sección 5 concluye el artículo.

2. La importancia de los objetivos formativos y algunos antecedentes

La definición de los objetivos de una asignatura es una herramienta fundamental para determinar el nivel de profundidad al que debe impartirse cada tema y qué deben saber los alumnos sobre ese tema. A partir de los objetivos deben elaborarse los contenidos, definir la metodología y estrategias docentes más apropiadas para cada tema y establecer los métodos de evaluación más adecuados. La definición de objetivos facilita, además, la coordinación entre asignaturas.

La definición del Espacio Europeo de Educación Superior supondrá grandes cambios en las nuevas titulaciones de informática [8]. Las recomendaciones de la Unión Europea para diseñar las nuevas titulaciones, que deben estar en funcionamiento antes de 2010, parten de la definición de las competencias que debe adquirir un alumno en cada carrera para llegar a una definición de objetivos y, a partir de ellos, diseñar los contenidos de las asignaturas.

De todos los esquemas propuestos en la literatura para clasificar los objetivos formativos, el más extendido es la taxonomía de Bloom [4]. En esta taxonomía se distinguen seis niveles. A continuación se detalla lo que se espera del alumno (qué debe ser capaz de hacer) para cada uno de los niveles:

- *Conocimiento*: recordar y reproducir información previamente suministrada.
- *Comprensión*: interpretar los conocimientos adquiridos y extraer conclusiones propias.
- *Aplicación*: utilizar los conocimientos adquiridos para resolver problemas reales (de complejidad media).
- *Análisis*: modelar y explicar sistemas o procesos usando conocimientos adquiridos.
- *Síntesis*: formular y aplicar modelos para resolver una situación compleja.
- *Evaluación*: emitir un juicio crítico sobre la validez de un modelo o una solución a un problema determinado.

Tanto ACM como IEEE [2,3] consideran que el nivel de competencia que debe alcanzar un ingeniero es el nivel de *Aplicación* y, en algunas materias, el de *Análisis*. Los niveles de *Síntesis* y *Evaluación* pertenecen al ámbito del doctorado.

Bloom trabajaba en el contexto de las humanidades, y aplicar su taxonomía en una titulación técnica no es tarea sencilla. La Universidad de Pittsburgh actualizó esta taxonomía en el año 2000 para diferentes campos de la ciencia y la ingeniería [5]. En el contexto de la informática, existen dos trabajos recientes. En [6] se describe un esquema de definición de objetivos basado en tres niveles jerárquicos. Un primer nivel de objetivos asignados por el centro a los departamentos, un segundo nivel de objetivos que los departamentos encargan a sus profesores y un tercer nivel de objetivos que los profesores exigen a sus alumnos. La formulación es jerárquica porque a cada objetivo de un nivel determinado le corresponden uno o varios objetivos del nivel inmediatamente inferior. En [7] se presenta una definición de objetivos basada en competencias profesionales, coherente con las ideas presentadas en [4]. Se redefine la taxonomía de Bloom, adaptándola a la ingeniería informática, y para cada uno de los niveles de competencia se detallan las capacidades que deben adquirir los alumnos y las habilidades que deben desarrollar.

3. Clasificación de objetivos formativos

La taxonomía de Bloom [4] distingue básicamente dos clases de objetivos: *generales* y *específicos*. Los objetivos generales tienen una granularidad

mucho más gruesa que los específicos y hacen referencia a capacidades que el alumno debe desarrollar de forma genérica. Los objetivos específicos, por el contrario, detallan de forma precisa los conocimientos y aptitudes que el alumno debe adquirir en un tema determinado.

En otros contextos educativos se distinguen tres tipos de objetivos: (i) *cognitivos*, (ii) *procedimentales* y (iii) *referentes a actitudes, valores y normas*. En este artículo clasificaremos los objetivos en tres grupos:

- *Relacionados con los contenidos técnicos de la carrera:* referidos al ámbito propio de la titulación (la ingeniería informática). En este grupo hay tanto objetivos *cognitivos* (por ejemplo, "el estudiante debe conocer el concepto de variable en un lenguaje de programación") como *procedimentales* (por ejemplo, "el estudiante debe ser capaz de diseñar un circuito sumador a partir de puertas lógicas elementales").
- *Relacionados con capacidades y aptitudes*: hacen referencia a capacidades generales, no ligadas a conceptos técnicos de la carrera, necesarias para un estudiante de ingeniería informática y posiblemente de muchas otras carreras. Por ejemplo, "el estudiante debe adquirir la capacidad de organizar su tiempo" o "el estudiante debe adquirir la capacidad de estudiar a partir de distintas fuentes de información".
- *Relacionados con actitudes, valores y normas*: se refieren a la disposición personal del estudiante con respecto a la sociedad. Por ejemplo, "el estudiante debe aceptar y seguir las normas de integridad académica que fije la universidad".

La clasificación en estas tres categorías permite presentar conjuntamente los objetivos *cognitivos* y *procedimentales* en el grupo de *contenidos técnicos*.

Los objetivos citados en la documentación consultada tienen un ámbito distinto del que se pretendía definir (el primer curso). O bien se refieren a una titulación completa (son mucho más ambiciosos) o a un tipo de conocimiento técnico muy concreto que abarca varias asignaturas monográficas de distintos cursos (por ejemplo, "la enseñanza de la programación").

En la mayoría de los casos, los objetivos formulados para asignaturas individuales suelen concentrarse exclusivamente en la categoría "contenidos técnicos propios de la informática". Es más, incluso dentro de los objetivos de tipo técnico, en muchas fuentes se recogen únicamente los que son de tipo *cognitivo* y se olvidan los (importantísimos) objetivos *procedimentales*.

Los autores no han encontrado ninguna fuente que haga explícitos aquellos objetivos relacionados con *capacidades y aptitudes* o referentes a *actitudes, valores y normas* que serían deseables para un primer curso de una carrera técnica. Sin embargo, es una opinión muy extendida que el fracaso universitario se explica más por deficiencias en *capacidades, aptitudes* y *actitudes* que por deficiencias en conocimientos iniciales. Por tanto, las principales aportaciones de este trabajo son:

- Citar explícitamente algunos objetivos relacionados con *capacidades y aptitudes* y relacionados con *actitudes, valores y normas*.
- Describir algunas estrategias docentes que favorecen la consecución de los objetivos relacionados con *capacidades y aptitudes.*

Muchas de estas técnicas y metodologías se aplican ya de manera habitual en la mayoría de los centros de enseñanza universitaria en informática. Por tanto, nuestra intención no es proponer técnicas novedosas, sino relacionarlas con los objetivos formativos a los que contribuyen.

3.1. Objetivos formativos relacionados con los contenidos técnicos de la carrera

Estos son los objetivos que tienen que ver con los conceptos, técnicas y procesos propios de la ingeniería informática. Aunque son los que de manera más directa se reflejan en los contenidos de las asignaturas, no es intención de este artículo discutirlos en detalle. En [1] se han dividido en cinco bloques temáticos:

- Razonamiento y lenguaje lógico-matemático
- Fundamentos matemáticos de la informática
- Fundamentos físicos de la informática
- Programación y algoritmia
- Estructura y funcionamiento de los computadores

Estos cinco bloques contienen, en total, unos 30 objetivos de tipo *cognitivo* y unos 30 de tipo *procedimental* (que, por motivos de espacio, no podemos indicar).

El detalle de estos objetivos es, sin duda, lo que permite identificar que la carrera es "de

informática". Paradójicamente, estos objetivos son los más variables y volátiles. Escuelas distintas podrían establecer contenidos substancialmente diferentes para su primer curso y, sin embargo, impartir una ingeniería informática coherente. En cambio, los objetivos de las otras dos categorías (*capacidades y aptitudes* y *actitudes, valores y normas*) se refieren a la adquisición de capacidades aceptadas sin discusión como muy convenientes, o incluso imprescindibles, en un estudiante de cualquier carrera científica o tecnológica: por ejemplo, la capacidad de resolución de problemas o el uso de métodos de estudio eficaces. Sin embargo, no parece haber unanimidad sobre si este tipo de objetivos deberían ser objetivos explícitos de un primer curso universitario: por ejemplo, suele cuestionarse si es misión de la universidad el enseñar valores a sus estudiantes, o si los hábitos correctos de estudio deberían suponerse adquiridos en niveles educativos anteriores (en cuyo caso la universidad puede legítimamente no preocuparse de los estudiantes de primer curso que tienen hábitos de estudio deficientes).

3.2. Objetivos formativos relacionados con capacidades y aptitudes

Los objetivos que se plantean a continuación son demasiado ambiciosos para ser asignados exclusivamente al primer curso. La meta, ya en sí bastante difícil, es que al acabar los estudios el alumno los haya alcanzado en un grado satisfactorio para el ejercicio profesional. No obstante, las bases para alcanzarlos deben ponerse precisamente en el primer curso, de forma que una vez acabado éste se haya hecho un cierto progreso en la consecución de cada uno de los objetivos. En la Sección 4 se detallan algunas estrategias para conseguir alcanzar (parcialmente) cada uno de estos objetivos al finalizar el primer curso.

Los objetivos formativos relacionados con capacidades y aptitudes que se identificaron para el primer curso son los siguientes:

- Incrementar la capacidad para resolver problemas
- Incrementar la capacidad para planificar y organizar el estudio
- Adquirir o reforzar hábitos de trabajo y estudio eficaces
- Incrementar la capacidad de trabajo en equipo a pequeña escala
- Incrementar la capacidad de razonamiento crítico
- Adquirir o reforzar la capacidad de búsqueda e integración de información
- Incrementar la capacidad de comunicación oral y escrita
- Adquirir la capacidad de usar de forma habitual y eficaz los servicios que ofrece la Universidad

Algunos de estos objetivos se formulan en [1] con mayor grado de precisión y detalle. No obstante, estos no son los únicos objetivos relacionados con *capacidades y aptitudes* que se deben exigir en una ingeniería informática. Al definir los objetivos de toda la carrera habría que incluir, sin duda, el incremento o adquisición de una lista de capacidades y aptitudes mucho más larga, entre las que destacan las capacidades de:

- Análisis y síntesis
- Organización y planificación
- Dirección de equipos y organizaciones
- Impartir formación (a los colaboradores)
- Comunicación efectiva en inglés
- Gestión de la información
- Toma de decisiones
- Trabajo en equipos multidisciplinares y/o en un contexto internacional
- Gestión de relaciones interpersonales
- Aprendizaje a lo largo de la vida y gestión de la trayectoria profesional

Se consideró que en un primer curso estas capacidades podían trabajarse en menor grado y que la lista de objetivos era ya suficientemente ambiciosa.

3.3. Objetivos formativos relacionados con actitudes, valores y normas

Existen unos valores básicos que todo alumno debería haber adquirido durante la enseñanza obligatoria, ya que definen lo que se entiende por ser "un buen ciudadano", y no debería ser misión de la universidad inculcarlos. No obstante, opinamos que el paso por la universidad es una gran oportunidad para reforzarlos o, como mínimo, para dar al estudiante la oportunidad de vivir en una organización que respeta y promueve activamente estos valores.

Los objetivos relacionados con *actitudes, valores y normas* deben tenerse en cuenta desde el primer día de clase, ya que definen tanto el

comportamiento del estudiante durante su etapa en la universidad como la postura que adoptará en sociedad cuando finalice sus estudios. Como objetivos más importantes queremos destacar que el estudiante debe:

- Aceptar las normas de integridad propias de un estudiante universitario, tanto en lo que se refiere a actos de evaluación como respecto a las normativas de utilización de los equipos, laboratorios y recursos materiales de la universidad.
- Aceptar que los estudios en una universidad pública tienen un coste para la sociedad y que, en contrapartida, la sociedad espera del estudiante dedicación y un buen rendimiento académico.
- Adquirir un cierto compromiso con valores como la solidaridad, la justicia y el progreso.
- Adoptar una actitud activa respecto al proceso de aprendizaje, asumiendo un papel protagonista en su propia formación.
- Respetar a la comunidad universitaria y a todas las personas en general, y evitar comportamientos que perturben el desarrollo normal de la docencia.

Como en el caso de las *capacidades y aptitudes*, hay otra serie de actitudes, valores y normas que un estudiante debería haber adoptado al terminar la carrera, pero que se consideró difícil formulación para el primer curso. Algunos ejemplos podrían ser:

- Reconocimiento de la diversidad y la multiculturalidad
- Motivación por la calidad y la mejora continua
- Sensibilidad por el medio ambiente

4. Estrategias docentes para conseguir los objetivos formativos relacionados con capacidades y aptitudes

En esta sección se proponen algunas técnicas y metodologías docentes que pueden emplearse para conseguir los objetivos formativos relacionados con *capacidades y aptitudes* descritos en la Sección 3.2.

4.1. Resolución de problemas

Para aumentar la capacidad de resolver problemas del estudiante es preciso que, desde el principio, se acostumbre a resolver los problemas por sí mismo, y no a copiar de la pizarra la resolución propuesta por el profesor u otro estudiante, ni a mirar las soluciones publicadas en los apuntes o en la página web de la asignatura.

Para ello, deben dedicarse suficientes horas a resolver problemas, tanto de clase como de trabajo personal. Consideramos muy importante dejarles leer por sí mismos los enunciados para aumentar su capacidad de comprensión del lenguaje técnico.

Esta metodología docente tiene como consecuencia que el número de problemas resuelto en clase es reducido. No obstante, este inconveniente puede paliarse proponiendo los problemas con suficiente antelación, de forma que el estudiante pueda tomarse en casa el tiempo necesario para pensar y tratar de resolverlos. Este método tiene además la ventaja de que cada estudiante puede dedicar a cada problema el tiempo que él necesite, de forma que el tiempo disponible en clase no es una limitación.

Resolver los problemas de este modo contribuye a mejorar la capacidad de abstracción del alumno y lo prepara para enfrentarse a problemas nuevos usando estrategias similares a las que ya ha usado en problemas anteriores.

Por otro lado, en clase se debería hacer énfasis explícito en las técnicas de resolución de problemas (en el sentido, por ejemplo, del clásico libro de Pólya [9], o de la Wikipedia [10,11]). De esta manera, se contribuye a mejorar la capacidad de abstracción del alumno y se lo prepara para enfrentarse a problemas nuevos.

4.2. Capacidad para planificar y organizar el estudio

La planificación del estudio requiere que el estudiante esté capacitado para fijar sus propios objetivos y prioridades, tanto académica como personalmente. Para ello, debe ser capaz de establecer prioridades entre las diferentes actividades que realiza y dentro de cada una de ellas. También debe ser capaz de evaluar, de una manera realista, sus propias posibilidades, sin sobrestimarse ni subestimarse. El alumno debe tomar sus propias decisiones y asumir sus errores.

Otra de las capacidades que se requieren es la capacidad de planificación (a lo largo del día, de

la semana y del curso). El alumno debe ser capaz de hacer estimaciones realistas del tiempo que le requerirá cada tarea y de modificar dicha planificación sobre la marcha. Para desarrollar esta capacidad, es conveniente dar orientaciones al estudiante sobre el tiempo que es aconsejable dedicar a cada parte de cada asignatura y sobre cuál es la metodología de trabajo más adecuada. Esta orientación es más importante al principio, en el primer curso, ya que el alumno tiene hábitos de estudio, en muchos casos, inapropiados para la universidad. Esta orientación se puede suprimir paulatinamente en cursos posteriores para fomentar la autonomía del estudiante.

La *evaluación continua* a lo largo del curso es una de las estrategias que suelen emplearse para forzar al estudiante a desarrollar esta capacidad, al exigirle un esfuerzo repartido a lo largo del curso.

Para ayudar al estudiante a evaluar sus posibilidades de forma realista, fijar sus prioridades y aprender a planificar su trabajo, la universidad puede crear la figura del tutor, un profesor que siga más o menos de cerca la carrera académica del alumno, especialmente durante el primer curso, y que le aconsejará en aquellas decisiones que deba tomar.

4.3. Hábitos de trabajo y estudio

Este punto está muy relacionado con el anterior. La adquisición de hábitos correctos requiere que el estudiante sea capaz de identificar los lugares y momentos más productivos para el estudio, además de reconocer y reducir las distracciones que le hacen perder eficacia.

La evaluación continua, de nuevo, contribuye a que el estudiante adopte un correcto ritmo de estudio. Por ejemplo, puede ser conveniente exigirle semanalmente la realización de alguna tarea (preparar problemas para resolver en clase o para entregar al profesor, realizar una exposición sobre un tema, estudiar un tema que el profesor no explicará en clase, etc.).

El alumno debe también adquirir el hábito de evaluar honestamente el progreso realizado en cada materia, identificar los puntos débiles y tomar nota de los errores cometidos para no repetirlos. Debe ser capaz de autoevaluar su conocimiento de un tema para decidir si ha de seguir trabajando o no en él. Utilizando la taxonomía de Bloom, debe saber distinguir, como mínimo, entre *conocer* (nivel conocimiento), *entender* (nivel comprensión) y *saber hacer* (nivel de aplicación).

Finalmente, el estudiante debe ser capaz de realizar un esfuerzo intelectual sostenido, trabajando en un mismo problema el tiempo necesario para obtener una solución y entendiendo que la recompensa por ese esfuerzo no tiene por qué ser inmediata.

También la evaluación continua ayuda a llevar la asignatura al día y a adquirir estos hábitos. Se pueden realizar durante el curso tres o cuatro pequeños exámenes, de 15-30 minutos, puntuables para la nota final, sin avisar previamente el día del examen, para que el alumno se acostumbre a llevar al día las asignaturas. La resolución y discusión de estos exámenes en la pizarra, por parte de los alumnos, inmediatamente después de realizarlos, contribuye a desarrollar el resto de capacidades, en especial el razonamiento crítico que se discute en la Sección 4.5.

4.4. Trabajo en equipo

El alumno debe ser capaz de formar su propio grupo de trabajo para estudiar en común las diferentes asignaturas que ha matriculado. De esta forma, se adquiere el hábito de compartir información y comparar los diferentes métodos de resolución de un problema y las soluciones obtenidas. Esto le permite aprender de sus propios aciertos y errores y de los de sus compañeros, tanto en contenidos como en procedimientos.

Para desarrollar esta capacidad se puede hacer que los alumnos trabajen en grupos reducidos (dos o tres personas) para realizar y entregar las prácticas de las diferentes asignaturas. También se pueden formar grupos de tres o cuatro estudiantes en algunas clases para resolver enunciados que se presten a la discusión, tanto dentro del grupo como posteriormente con otros grupos.

4.5. Razonamiento crítico

La capacidad de razonamiento crítico es fundamental en un ingeniero, y debe ser cultivada desde el primer curso y desarrollada en dos vertientes diferentes: la capacidad de analizar críticamente el proceso de solución y la capacidad de analizar críticamente la solución en sí misma.

Una forma sencilla de despertar la capacidad de analizar críticamente el proceso de solución consiste en discutir las soluciones propuestas por

los estudiantes a los problemas propuestos en la asignatura. Un alumno sale a la pizarra después de resolver el problema personalmente, explica su solución y la discute con los demás. El profesor ejerce de moderador y participa en la discusión.

Este método da muy buenos resultados, especialmente cuando el profesor "selecciona bien" al alumno. Por ejemplo, se puede escoger a un estudiante que no haya resuelto el problema correctamente, pero que se haya acercado a la solución. En este caso, es conveniente que los fallos cometidos representen adecuadamente los de un grupo significativo de estudiantes de la clase.

Para desarrollar la capacidad de analizar críticamente la solución obtenida es preciso obligar al alumno a analizar los resultados obtenidos y a decidir si "parecen" correctos. Esto supone que, en algunos casos, el estudiante debe pensar previamente acerca de la magnitud o el tipo de resultado que debe obtener. El estudiante debe meditar sobre si el resultado obtenido tiene o no sentido y si lo puede verificar por otra vía. Muchos de los estudiantes de nuestras aulas no poseen esta capacidad y se conforman rápidamente con la primera solución que obtienen, incluso cuando realizan un examen, a pesar de que la solución sea obviamente incorrecta.

4.6. Capacidad de búsqueda e integración de la información

El estudiante debe acostumbrarse a aprender de distintas fuentes y a no conformarse únicamente con los apuntes que toma, generalmente bastante mal, en las clases presenciales del profesor. Debe aprender a detectar cuándo le falta más información para comprender adecuadamente la materia o para adquirir las habilidades requeridas por cada asignatura. Puede obtener esta información de la biblioteca, de Internet, de las intranets de las asignaturas o de otros recursos docentes ofrecidos por la universidad. Las consultas con el profesor también son importantes en este punto, y el estudiante debería acostumbrarse a utilizarlas, desde primer curso, siempre que sea necesario para su progreso.

El auge de Internet ha propiciado que en la red pueda encontrarse mucha información que no ha sido convenientemente contrastada. Es común encontrar apuntes y trabajos de las asignaturas más variopintas escritos por otros estudiantes. Lamentablemente, es también habitual que esta documentación esté plagada de errores. Es muy importante que el alumno disponga de la capacidad crítica suficiente para filtrar adecuadamente esta información errónea.

Para desarrollar esta capacidad se puede encargar a los estudiantes la realización de un trabajo sobre un tema, o bien el estudio por su cuenta de un tema que no será explicado (aunque sí discutido) en clase.

4.7. Comunicación oral y escrita

Una de las habilidades imprescindibles en un ingeniero es tener una adecuada capacidad de comunicación para presentar con claridad los resultados de su trabajo o lo que pretende conseguir de su equipo de colaboradores.

Esta capacidad debe comenzar a desarrollarse ya en el primer curso. Una estrategia posible es hacer que los estudiantes expliquen la resolución que proponen para un cierto problema, discutiéndola con el resto de la clase. También pueden presentar oralmente en la clase los trabajos que hayan realizado.

Consideramos importante que haya alguna entrega escrita de documentación en la que el aspecto formal sea importante. Puede incluso exigirse que alguna de las entregas esté redactada a mano, para conferir valor a aspectos como la pulcritud, la caligrafía y la ortografía que, de otro modo, quedan enmascarados por los procesadores de textos y correctores ortográficos automáticos.

Corregir problemas resueltos por otros compañeros de clase puede ayudar también al alumno a darse cuenta de cómo debe (y cómo no debe) entregarse una documentación.

Estos aspectos deben ser adecuadamente evaluados dentro de su nota final para que el alumno sea plenamente consciente de su importancia.

4.8. Uso habitual y eficaz de los servicios que ofrece la Universidad

La consecución de este objetivo permitirá al estudiante aumentar la probabilidad de éxito en el resto de objetivos formativos. El alumno debe conocer y usar, de acuerdo con su finalidad, los sistemas informáticos que la universidad pone a su

alcance: ordenadores, software básico, software especializado de las asignaturas, intranets y correo electrónico (para comunicarse con los profesores, la administración y otros alumnos) entre otros. También es importante que conozca a los interlocutores que la universidad le ofrece para resolver dudas y problemas: sus profesores, los profesores responsables de asignatura, su tutor, la delegación de alumnos, la secretaría académica, el jefe de estudios, etc. Este objetivo se favorece mediante una buena campaña de información desde el mismo momento de la matriculación.

5. Conclusiones

En este artículo se resume una formulación de objetivos formativos de un primer curso de la Ingeniería Informática presentada en detalle en [1]. Los objetivos se clasifican en tres categorías: *contenidos técnicos* de la Ingeniería Informática, *capacidades y aptitudes* y *actitudes, valores y normas*. El artículo se centra en los objetivos de adquisición de *capacidades y aptitudes* y, dentro de ellos, en los que pueden empezar a abordarse desde el primer curso. Se describen algunas estrategias para conseguir estos objetivos que son extrapolables a otros cursos de la carrera.

Agradecimientos

Este trabajo ha sido financiado en parte por el proyecto CICYT TIN2004-07739-C02-01

Además de los autores, participaron en la redacción del documento [1] los profesores Albert Avinyó, Jordi Cortadella, Jordi Martí, Montserrat Maureso, Glyn Morrill, Albert Rubio, Joan Trias y Jordi Tubella. La fase inicial del trabajo contó con una *Ayuda a la Mejora de la Calidad Docente Universitaria* de la convocatoria 2000 del *Dep. d'Universitats, Recerca i Societat de la Informació de la Generalitat de Catalunya*. También queremos agradecer a la Facultat d'Informàtica de Barcelona y al Departament d'Arquitectura de Computadors de la UPC su ayuda durante la elaboración de este trabajo.

Versiones anteriores ligeramente distintas de este artículo se han presentado en JENUI2004 [12] y en Novática [13].

Referencias

[1] *Objectius de la Fase de Selecció de les titulacions de la FIB.* Facultat d'Informàtica de Barcelona, julio 2003. Disponible en http://www.lsi.upc.es/~gavalda/docencia/dofs.pdf

[2] *IEEE / ACM Computing Curricula*. http://www.computer.org/education/cc2001/

[3] *Accreditation Criteria*. Accreditation Board for Engineering and Technology, Inc., http://www.abet.org/

[4] *Taxonomía de los objetivos de la educación, Tomo I (conocimientos) y Tomo II (dominio afectivo)*. B.S. Bloom, J.T. Hastings y G.F. Madaus, Ed. Marfil, Alcoy 1973.

[5] *Bloom and Krathwohl Definitions of Levels and McBeath Action Verbs*. The University of Pittsburg, 2000. http://www.engrng.pitt.edu/~ec2000/ec2000_downloads.html

[6] *Formulación de los objetivos de una asignatura en tres niveles jerárquicos*. J.J. Navarro, M. Valero-García, F. Sánchez y J. Tubella. JENUI2000.

[7] *Niveles de competencia de los objetivos formativos de las ingenierías.* M. Valero-García y J.J. Navarro, JENUI2001.

[8] *Las futuras titulaciones universitarias de Informática en España dentro del Marco del Espacio Europeo de Educación Superior*. F. Sánchez y M.R. Sancho, Novática 168, Abril 2004, pp. 40-45.

[9] *How to Solve It*. G. Pólya. 1ª edición en Princeton University Press, 1945.

[10] *How to Solve It.* Entrada en *The Wikipedia, the Free Encyclopedia*, http://en.wikipedia.org/wiki/How_to_Solve_It

[11] *Problem Solving*. Entrada en *The Wikipedia, the Free Encyclopedia*, http://en.wikipedia.org/wiki/Problem-solving

[12] *Objetivos formativos y estrategias docentes para el primer curso de las ingenierías informáticas*. F. Sánchez y R. Gavaldà. JENUI2004.

[13] *Propuesta de objetivos formativos para el primer curso de las ingenierías informáticas y de algunas estrategias docentes para conseguirlos*. F. Sánchez y R. Gavaldà, Novática, aparición prevista para Julio 2005

Autoevaluación y co-evaluación: estrategias para facilitar la evaluación continuada

Miguel Valero-García, Luis M. Díaz de Cerio
Dept. d'Arquitectura de Computadors
Escola Politècnica Superior de Castelldefels, Universitat Politècnica de Catalunya
Avda. del Canal Olímpic, s/n
08860 Castelldefels (Barcelona)
{miguel, ldiaz}@ac.upc.edu

Resumen

Allá donde se intentó implantar de forma generalizada, la evaluación continuada acabó por provocar frustración entre el profesorado, por diferentes razones, una de las cuales es el importante incremento de carga de trabajo debida a la corrección de los ejercicios de los alumnos. En nuestra opinión, se suelen mezclar dos aspectos que es importante separar: (a) Evaluación Formativa: evaluar e informar frecuentemente al alumno de forma cualitativa, sobre lo bien o lo mal que va en el curso (cuestión fundamental para el aprendizaje) y (b) Evaluación Calificativa: evaluar cuantitativamente y asignar calificaciones a las tareas, ejercicios, etc., a lo largo del curso (cuestión secundaria respecto al aprendizaje). Cuando se separan estos aspectos, es posible plantear sistemas de evaluación continuada que satisfacen el requerimiento fundamental: mantener informado al alumno, a un coste razonable para el profesor. En este trabajo se propone el uso de la autoevaluación y la co-evaluación como sistemas de evaluación formativa a coste razonable, y se analizan las implicaciones de diferentes esquemas.

1. Introducción: La promesa de la evaluación continuada (la teoría y la realidad)

Está ampliamente aceptado que un ingrediente de la docencia de calidad es un sistema de retroalimentación que permita al alumno mantenerse puntualmente informado sobre su progreso (o falta de progreso) en el plan de aprendizaje [1]. Después de resolver un ejercicio, por ejemplo, un alumno necesita saber si lo ha hecho bien o mal, y por qué. Y necesita saberlo pronto, para poder tomar rápidamente las acciones correctoras necesarias. En otras palabras, se trata de evitar que el alumno descubra el día del examen final (o peor aún, al conocer las notas del examen final) que ha entendido bien poca cosa del material del curso.

A pesar de esta obviedad, no es fácil encontrar asignaturas con un buen sistema de evaluación continuada, especialmente cuando el número de alumnos por profesor es elevado. De hecho, una de las quejas más frecuentes de los alumnos es que no tienen una buena información sobre su progreso a lo largo del curso, y que con frecuencia se ven sorprendidos en los exámenes, cuando se pone en evidencia que su rendimiento real está muy por debajo de su predicción.

La evaluación continuada ha sido un estandarte en materia de mejora de la docencia en muchos sitios. Así por ejemplo, fue uno de los ingredientes principales de la reforma de los planes de estudio en la Universidad Politécnica de Cataluña (UPC), a inicios de los 90. Sin embargo, en muchos casos, y por razones de coste tanto desde el punto de vista del profesor como desde el punto de vista del alumno, los sistemas de evaluación continuada han ido simplificándose hasta convertirse en poco más de un examen parcial a mitad del cuatrimestre, y un último examen al final, con lo que el objetivo básico de la evaluación continuada (mantener bien informado al alumno sobre el estado de su aprendizaje) no se cumple.

En este artículo proponemos una serie de ideas que pueden resultar de utilidad para orga-

nizar un sistema de evaluación continuada a un coste asumible. En concreto, en la siguiente sección separamos los dos conceptos clave que intervienen en la evaluación continuada: información continuada (evaluación formativa) y calificación continuada (evaluación calificativa). En la sección 3 proponemos dos técnicas para realizar la evaluación formativa con bajo coste y por tanto disminuyendo en global el coste de la evaluación continuada. En la sección 4 veremos cómo corresponden las técnicas propuestas respecto a los niveles de competencia según la conocida taxonomía de Bloom. En las secciones 5 y 6 presentamos diferentes metodologías para llevar a cabo las técnicas propuestas y finalmente, en la sección 7, presentamos las conclusiones.

2. Información continuada versus calificación continuada (separemos las cosas)

En esta sección consideramos algunas cuestiones teóricas sobre la evaluación y sus características. Estas consideraciones son la base en la que se apoyan las propuestas que se realizan en este artículo.

Cuando se trata de la evaluación del nivel de aprendizaje de nuestros alumnos, solemos distinguir entre dos tipos, según el uso que se hace de la información:

- Evaluación formativa: se usa para guiar y mejorar los procesos de enseñanza y aprendizaje
- Evaluación calificativa: se usa para determinar la calificación que acredita el nivel de aprendizaje conseguido por el alumno.

Por otra parte, un sistema de evaluación (tanto calificativa como formativa) puede tener (entre otros) los siguientes atributos:

- Precisión y fiabilidad: El resultado de la evaluación es el mismo con independencia de la persona que realice la evaluación, o del momento en que se realice la evaluación (fiabilidad), y ese resultado tiene poco margen de error (preciso).
- Prontitud: El resultado de la evaluación está en manos del alumno y del profesor lo antes posible, después de haber realizado el acto a evaluar.

Es interesante ahora analizar la relación que existe entre los tipos de evaluación y los atributos de la evaluación. En particular, puede afirmarse que:

La evaluación calificativa debe ser precisa y fiable, puesto que está en juego el expediente del alumno, que es un documento oficial que puede tener mucha trascendencia a la hora de buscar trabajo, optar a becas, etc. Sin embargo, no es especialmente crítico que los resultados de la evaluación calificativa estén disponibles con prontitud[1].

El atributo más importante de la evaluación formativa es la prontitud. En este caso, también es deseable que el sistema sea preciso y fiable, pero estos atributos son secundarios. En otras palabras, cuando un alumno hace, por ejemplo, un ejercicio, lo importante es que el alumno sepa pronto si las decisiones importantes han sido acertadas. En cambio, no es excesivamente importante saber si el resultado merece una nota de 6.5 o una de 7 (sí que lo es, insistimos, en el caso de la evaluación calificativa).

Finalmente, es importante observar que lo que hace que un sistema de evaluación sea costoso en términos de tiempo de profesor es que el sistema sea preciso y fiable. Efectivamente, el tener que determinar una calificación, típicamente con una precisión de 0.5 puntos, implica un tiempo de análisis del trabajo realizado por cada alumno, que puede llegar a suponer una carga total de trabajo importante cuando hay muchos alumnos por profesor, y muchos actos de evaluación a lo largo del curso.

Lo importante ahora es observar que, en el caso de la evaluación formativa (la que nos interesa en el contexto de este trabajo), es aceptable sacrificar precisión y fiabilidad (y por tanto, reducir coste para el profesor), siempre y cuando se mantenga el requisito de prontitud. Este es el principio subyacente en las propuestas que se desarrollan en las secciones siguientes.

En resumen, cuando se habla de evaluación continuada, conviene distinguir entre información continuada y calificación continuada. La

[1] No obstante, casi siempre la prontitud acaba siendo necesaria para satisfacer los requisitos del sistema administrativo, y del calendario académico, que debe cerrar procesos en una fecha determinada para dar inicio al siguiente cuatrimestre.

información continuada es imprescindible para que el proceso de enseñanza-aprendizaje tenga salud. La calificación continuada es costosa y no es en absoluto necesaria. Incluso en muchos casos, especialmente en el caso de la organización cuatrimestral, no es conveniente calificar (no confundir con evaluar) al alumno antes de la última fase del curso, debido a que parte de su nota queda determinada cuando todavía no ha tenido la oportunidad de asimilar y ejercitar los conceptos básicos. Éste es, por ejemplo, el caso de un primer curso de enseñanza de la programación, en la que todo lo que puede calificarse en la primera mitad de un cuatrimestre tiene poca relevancia comparado con la fase final del curso, que es cuando el alumno está en condiciones de resolver ejercicios de una mínima entidad.

3. Autoevaluación y co-evaluación: información con prontitud y a bajo coste

Las estrategias de autoevaluación y co-evaluación pueden usarse como base para la organización de un sistema de evaluación que proporcione información con prontitud, aunque para ello tengamos que renunciar a la precisión y a la fiabilidad. La idea básica de estas estrategias es que los alumnos pueden ser colaboradores del profesor, en este caso, en las tareas de evaluación; ya que los profesores, por lo general, no disponen de ayudantes para realizar la labor evaluadora. Si los alumnos han de colaborar en la evaluación, las únicas dos opciones de las que disponemos son: evaluarse a sí mismos (autoevaluación) o evaluar a otros compañeros (co-evaluación). En concreto, cuando se usa la estrategia de la autoevaluación, es el propio alumno el que determina en qué medida su trabajo está bien o mal siguiendo las instrucciones del profesor. En el caso de la co-evaluación, cada alumno evalúa el trabajo de uno o varios de sus compañeros, también siguiendo las instrucciones del profesor.

La autoevaluación y la co-evaluación proporcionan información con prontitud, puesto que si el profesor tiene preparadas las instrucciones con antelación, los alumnos pueden realizar la evaluación inmediatamente después de realizar el trabajo y obtener las conclusiones rápidamente. Lógicamente, la evaluación será menos precisa y fiable que si la hubiese realizado el profesor, puesto que el profesional es él, y no los alumnos.

Además de resolver el problema básico que nos concierne en este trabajo, la autoevaluación y la co-evaluación tienen otras virtudes que conviene tener bien presentes [2]. En concreto, en el caso de la autoevaluación:

- Los alumnos van interiorizando los criterios de corrección que el profesor hace explícitos a través de las instrucciones para la autoevaluación. Esto permite a los alumnos ajustar cada vez más sus respuestas a lo que el profesor espera.
- Los alumnos desarrollan el hábito de la reflexión, y la identificación de los propios errores, cuestión fundamental cuando se trata de formar personas con capacidad para aprender de forma autónoma.

En el caso de la co-evaluación, además de las virtudes anteriores, podemos mencionar también las siguientes:

- Los alumnos se esfuerzan más, impulsados por la motivación de quedar bien ante los ojos de sus compañeros (este tipo de motivación suele ser superior a la de quedar bien ante los ojos del profesor).
- Los alumnos desarrollan el hábito de criticar de forma constructiva el trabajo realizado por compañeros con los que van a tener que continuar colaborando. Ésta es también una habilidad fundamental que se echa en falta con frecuencia en el mundo profesional.

Es posible que al plantear un sistema de evaluación continuada basado en la autoevaluación y en la co-evaluación los alumnos manifiesten unas ciertas reticencias, e incluso puedan acusar al profesor de cargarles a ellos con un trabajo y una responsabilidad que no les corresponde. En estos casos, conviene admitir que, efectivamente, uno de los objetivos de estas estrategias es reducir el tiempo que el profesor dedica a la evaluación para poder dedicar ese tiempo a otras tareas igualmente importante para el aprendizaje. Pero además, conviene tener a mano todo el repertorio de virtudes de estas estrategias que, bien planteadas, proyectarán ante los alumnos la idea de que el profesor ha pensado en lo que es

bueno para sus alumnos a la hora de establecer los elementos de su programa.

4. La evaluación y los niveles de competencia

Otro elemento importante a la hora de establecer un sistema de evaluación continuada es el análisis del nivel de competencia de los objetivos formativos que se van a evaluar. La taxonomía de referencia para el estudio del nivel de competencia de los objetivos formativos es la taxonomía de Bloom [3], según la cual un objetivo formativo pertenece a uno de los siguientes niveles, en orden creciente de competencia: *conocimiento, comprensión, aplicación, análisis, síntesis o evaluación.*

En [4] se hizo un trabajo de adaptación de la taxonomía de Bloom al ámbito de la enseñanza de la ingeniería, y se plantearon algunas reflexiones sobre el equilibrio que debe existir (y el desequilibrio que muchas veces existe) entre nivel de competencia, métodos docentes y métodos de evaluación.

En el contexto de este trabajo, en el que lo que nos interesa es la evaluación, proponemos una taxonomía simplificada, basada en tres niveles de competencia. A continuación se describen cada uno de estos niveles y se da un ejemplo de ejercicio (o tipo de ejercicio) de cada nivel, perteneciente al ámbito de la enseñanza de la programación de ordenadores.

- *Conocimiento:* Requiere que el alumno recuerde datos, hechos, información que le ha sido suministrada con anterioridad.
    1. Describe la sintaxis y la semántica del bucle "for" del lenguaje C
    2. Indica cuál es el rango de representación de los enteros de tipo short, en lenguaje C
- *Comprensión*: Requiere que el alumno utilice un determinado procedimiento sistemático (una "receta") sobre un caso particular.
    3. Determinar el valor final de una variable después de ejecutar una secuencia de sentencias de lenguaje C
    4. Escribir el código en lenguaje C para ordenar, mediante el método de la burbuja, un vector de caracteres.
- *Aplicación*: Requiere que el alumno elija, de entre las "recetas" que ha comprendido previamente, la más apropiada para resolver un determinado problema.
    5. Determinar cuál es la estructura de datos más adecuada para resolver un problema
    6. Determinar qué operaciones de un determinado programa pueden ser realizadas mediante procedimientos y funciones de librerías ya existentes.

Es importante hacer notar que los nombres que Bloom da a los niveles de competencia muchas veces no corresponden directamente al significado de la palabra en castellano. Por tanto, muchos ejemplos que nosotros podamos considerar de comprensión o de aplicación no corresponden directamente con los respectivos niveles de competencia.

En la taxonomía simplificada que se propone, los niveles de conocimiento y comprensión se corresponden directamente con los dos primeros niveles de la taxonomía de Bloom. Sin embargo, el nivel de aplicación de la propuesta corresponde a una fusión de los niveles de aplicación y superiores de la taxonomía de Bloom. Consideramos que esta simplificación es válida en el contexto de asignaturas iniciales de primer ciclo, en las que rara vez se supera el nivel de aplicación en la taxonomía de Bloom.

Cuando el ejercicio que debe realizar el alumno es de nivel de conocimiento o de comprensión, la respuesta es única o admite muy pocas variaciones. Además, en el caso de comprensión, los resultados intermedios en el proceso de aplicar la "receta" también son únicos. Por ejemplo, en el caso del ejercicio 4, no hay en realidad muchas formas distintas de codificar el algoritmo de la burbuja para ordenar un vector de caracteres. En estas condiciones, la evaluación es muy sencilla: la solución del alumno debe asemejarse a la solución "oficial", y cualquier diferencia es, en potencia, un error del alumno. Este principio es base para los esquemas de autoevaluación propuestos en la siguiente sección.

Cuando el ejercicio es de nivel de aplicación, diferentes alumnos pueden dar respuestas distintas, todas ellas válidas. Ahora no tiene sentido usar una solución "oficial" como base para la evaluación (aunque sí puede ser útil

> Para cada uno de los ejercicios siguientes, debes preparar un informe de autoevaluación que se compone de dos partes:
>
> *a) Solución al ejercicio.*
>
> *b) Errores que nunca más volveré a cometer.*
>
> Para escribir la parte (b) debes usar la solución oficial del ejercicio. Puesto que cada uno de los ejercicios tiene una solución única, cualquier diferencia entre tu solución y la solución oficial puede ser:
>
> - Un error en la solución oficial (cosa poco probable).
> - Un error tuyo.
> - Una diferencia admisible entre las soluciones.
>
> Compara tu solución con la oficial, identifica las diferencias y clasifica cada una de ellas según los tipos anteriores. Escribe tus conclusiones en la sección "Errores que nunca más volveré a cometer", de tu informe de autoevaluación.

Figura 1: Instrucciones para la autoevaluación.

como ejemplo). Lo que se necesita es explicitar los atributos que deben tener las soluciones correctas, de manera que la evaluación consiste en identificar los atributos propuestos en las respuestas a evaluar. Por ejemplo, en el caso del ejercicio 5, uno de los atributos de una solución correcta es que la estructura de datos propuesta ocupe un espacio razonable de memoria, o en otras palabras, que no haya soluciones alternativas que ocupen mucho menos espacio de memoria. Así pues, la identificación clara de los atributos de las soluciones correctas será el principio básico para el sistema de co-evaluación que se propone en la sección 6.

5. Autoevaluación y el nivel de comprensión

En esta sección se propone un esquema de autoevaluación que puede usarse cuando los objetivos formativos a evaluar corresponden al nivel de conocimiento o comprensión.

La idea básica es partir de una solución "oficial" al ejercicio, y basar la autoevaluación en la comparación de la propia solución con la solución "oficial". Puesto que, dada la naturaleza del ejercicio, no hay muchas variaciones válidas posibles en la respuesta, cualquier diferencia entre la respuesta del alumno y la solución "oficial" es un error potencial. Por tanto, la autoevaluación consiste en identificar las diferencias, reconocer aquellas que corresponden a errores, y justificar aquellas que son variaciones admisibles. Las instrucciones para los alumnos podrían ser las mostradas en la figura 1. Es muy importante que el alumno entienda la importancia de hacer con rigor la comparación de la propia solución con la "oficial". Al realizar la autoevaluación por primera vez, es habitual que el alumno escriba, en la sección de *Errores que nunca más volveré a cometer*, cosas como: "Mi solución es claramente distinta de la oficial, pero creo que también es correcta". En un caso así, lo adecuado es devolver la autoevaluación al alumno y exigirle que realice correctamente la identificación y clasificación de las diferencias.

Los aspectos de logística para organizar el sistema de autoevaluación también son muy importantes. Se proponen, tres posibilidades:

1. Los alumnos no entregan las autoevaluaciones al profesor. Es un material de uso propio. Si bien esta estrategia cumple con el propósito perseguido (informar al alumno con prontitud, con una carga de trabajo baja para el profesor), no es recomendable porque:
   - El profesor no tiene ocasión de intervenir para asegurarse que los alumnos hacen bien la autoevaluación (y las primeras veces, seguro que no la harán bien).
   - El profesor no tiene información de cómo van los alumnos. No puede realizar accio-

nes correctoras, ni a nivel individual ni colectivo.

2. El profesor pide, de vez en cuando, los informes de autoevaluación. De esta forma, puede asegurarse de que los alumnos están haciendo bien la tarea, y tiene información sobre las dificultades de cada uno de los individuos, y de las dificultades generales de la clase. Idealmente, el profesor explica a sus alumnos, al inicio del curso, que uno de sus objetivos para el curso es que aprendan a identificar sus propios errores, de manera que calificará los informes de autoevaluación (en función de si se han identificado bien los errores), y usará esa calificación para obtener parte de la nota final del alumno. Naturalmente, esta opción implica algo más de trabajo para el profesor. La opción admite dos variantes:

- De vez en cuando, el profesor recoge el informe de autoevaluación de todos los alumnos, correspondiente a un ejercicio.
- De vez en cuando, el profesor recoge todos los informes de autoevaluación de un determinado alumno o grupo de alumnos.

La primera opción tiene la ventaja de que el profesor puede tener una idea clara de las dificultades generales del grupo con un determinado tipo de ejercicios. Tiene el inconveniente de que se trabaja con muchas hojas sueltas, cosa que puede resultar incómoda. Con la segunda opción, el alumno entrega al profesor una carpeta bien organizada con sus informes de autoevaluación (y, posiblemente, otros materiales del curso). Además, al analizar esta carpeta, el profesor puede obtener más fácilmente una visión nítida de la evolución del alumno (hasta qué punto su rendimiento va mejorando). Otra ventaja adicional es que el profesor puede pedir la carpeta con más frecuencias a aquellos alumnos que necesitan más ayuda.

3. Además de calificar la calidad de la autoevaluación (como en la opción anterior) el profesor usa la calificación que se asignan los alumnos para obtener la nota final de la asignatura. En este caso, en la guía para la autoevaluación, además de la solución oficial deben aparecer los criterios precisos para asignar una calificación a la solución. La ventaja de esta opción es que el alumno tiene una mayor responsabilidad en el proceso, y tiende a tomarse más en serio la autoevaluación, al ver que hay un impacto directo mayor en la calificación final. Sin embargo, también hay inconvenientes importantes: el alumno puede centrarse más en la nota que en la identificación de errores, la elaboración de las instrucciones para la autoevaluación se complica[2], y el profesor debe, probablemente, dedicar más tiempo a analizar los informes de autoevaluación. Por otra parte, no es una alternativa apropiada cuando la organización y contenidos del curso no recomiendan el asignar calificaciones hasta la fase final en la que los alumnos están en condiciones óptimas para ser calificados.

Cualquiera que sea el esquema adoptado, es muy importante tener en cuenta que los alumnos necesitan repetir el proceso con una cierta frecuencia para aprender a autoevaluarse, y a sacar provecho de la actividad. Las experiencias puntuales (por ejemplo, un único ejercicio de autoevaluación en todo el curso) suelen generar más frustración y desorientación que otra cosa, y sólo son recomendables si se trata de probar la mecánica, de cara a una implementación generalizada en el futuro.

6. Co-evaluación y el nivel de aplicación

Cuando se trata ejercicios de nivel de aplicación, una solución "oficial" no es suficiente para realizar la evaluación, puesto que el ejercicio admite varias soluciones correctas y distintas entre sí. Lo importante es explicitar las características que debe tener una determinada solución para que pueda ser considerada correcta.

Una forma de presentar a los alumnos las características de las soluciones correctas es usar rúbricas [5] como las que se muestra en la figura 2. En este caso, se trata de evaluar el código para resolver un determinado problema de programación. En la columna de la izquierda se identifican diferentes criterios de evaluación del

[2] Muchos profesores insisten en lo complicado que puede resultar el explicitar los criterios para que el alumno pueda calcular una calificación precisa. Sin embargo, eso no es más difícil que explicitar los criterios de corrección que debe usar un grupo de profesores que se reparten la corrección de un ejercicio del examen final, cosa que hacemos de forma habitual.

Código			
	Nivel de calidad		
Criterio	**3** **Notable**	**2** **Suficiente**	**1** **Insuficiente**
Correcto	La aplicación funciona bien en todos los casos. No he encontrado ningún fallo.	Hay (como máximo) un par de situaciones en las que el programa no ha funcionado bien.	La aplicación falla constantemente.
Robusto	La aplicación resiste sin bloquearse todos los errores típicos que puede cometer un usuario "poco hábil". No he conseguido que se cuelgue.	Es razonablemente robusto. No es fácil que se quede colgado, pero en uno o dos casos se bloqueó.	La aplicación no es robusta en absoluto. Se queda colgada con frecuencia ante errores típicos del usuario al entrar datos.
Amigable	El usuario no tiene ninguna duda, en ningún momento, sobre cómo interactuar con la aplicación, qué datos debe suministrar y cómo interpretar los resultados y mensajes de la aplicación.	Los mensajes e información que da la aplicación son suficientes para trabajar bien. Sin embargo, en alguna ocasión he tenido algunas dudas sobre lo que hay que hacer o cómo hay que hacerlo.	El usuario tiene dudas constantes sobre lo que le está pidiendo la aplicación, y es difícil interpretar los resultados y mensajes en pantalla.
Comparado con el nuestro	Este código es mejor.	Este código es similar.	Este código es peor.

Figura 2: Ejemplo de rúbrica para la evaluación de un código

código. En las columnas de la derecha se identifican, para cada uno de los criterios de evaluación, las características que debe reunir una determinada solución para que pueda ser considerada *notable*, *suficiente* o *insuficiente* (en este ejemplo, se consideran sólo tres niveles de calidad). La tarea del evaluador es identificar en la solución que tiene que evaluar, las características señaladas en la rúbrica, y determinar el nivel de calidad para cada uno de los criterios de evaluación. El informe de evaluación se completa con los argumentos en los que el evaluador se basa para tomar la decisión, como por ejemplo:

a) El código falla en los casos de prueba 1 y 4.
b) El mensaje que avisa al usuario para que introduzca datos no es claro: tengo dudas sobre el formato que debo usar.
c) El código está muy mal indentado. Tengo muchas dificultades para identificar los bloques de código.

Sin duda, la evaluación del nivel de aplicación es más difícil de objetivizar que la evaluación del nivel de conocimiento o comprensión. En el caso, por ejemplo, del argumento (b) de la lista anterior, dos personas distintas pueden tener opiniones diferentes sobre la claridad del mensaje que el programa ofrece al usuario. Además, con toda probabilidad, el autor del código opinará que su mensaje es claro. Por estas razones, la evaluación del nivel de aplicación no acaba de combinarse bien con la estrategia de autoevaluación (cuando se trata de cosas opinables, no somos buenos jueces de nosotros mismos).

En cambio, la estrategia de la co-evaluación se adapta mejor a la naturaleza de este tipo de evaluación. Un alumno puede juzgar de forma más neutral el resultado del trabajo de otros compañeros, naturalmente con la ayuda de una rúbrica bien elaborada. Además, su propia solución es un punto de referencia importante para valorar los méritos y deméritos de las soluciones de los compañeros. Esa es la razón de que, en la rúbrica que se muestra como ejemplo, la última fila haga referencia a una comparación entre la solución propia y la evaluada.

Los aspectos logísticos de la co-evaluación son más complicados que los de la autoevaluación, simplemente porque los resultados de los trabajos de los alumnos deben ser recogidos y redistribuidos para ser evaluados por otros. Las soluciones a este problema dependerán mucho de las circunstancias de cada caso, por lo que poco más puede decirse aquí al respecto con carácter general.

Los comentarios que se han hecho en el apartado anterior en cuanto al uso que hace el

profesor de los resultados de la autoevaluación también son aplicables en el caso de la co-evaluación. En todo caso, conviene ser consciente de que los alumnos no suelen mostrarse favorables a asignar a los compañeros calificaciones que puedan afectar a la nota final. Por tanto, si se opta por esta alternativa, es importante tener a punto el argumento para convencerles de que la habilidad de emitir críticas constructivas, y juzgar el trabajo de los compañeros es importante para el ejercicio profesional.

7. Conclusión

Las técnicas que se plantean de forma teórica en este trabajo se están poniendo en práctica en la asignatura Laboratorio de Programación (LP), perteneciente a la Ingeniería Técnica en Telecomunicaciones, de la Escuela Politécnica Superior de Castelldefels (UPC). En realidad, han sido las experiencias en esta asignatura las que nos han ayudado a poner en orden nuestras ideas, en la forma presentada aquí.

La asignatura ofrece un escenario ideal. En la primera parte, los objetivos formativos son de nivel de comprensión. Los alumnos deben escribir códigos en Visual C++ que resuelvan casos particulares mediante algoritmos conocidos. En esta primera parte se utiliza la técnica de la autoevaluación, los informes de autoevaluación han de incluir una justificación correcta, no sólo de los errores, sino de las diferencias en general y esta nota se usa en el cálculo de la nota final.

En la segunda parte de la asignatura los objetivos son de nivel de aplicación. Los alumnos deben tomar decisiones relativas a las estructuras de datos y algoritmos que hay que usar para resolver un determinado problema. Las actividades se realizan en modo proyecto, de forma que los alumnos, trabajando en grupo, deben resolver un problema. La técnica de la co-evaluación basada en rúbricas se usa para que cada grupo evalúe el trabajo de otros.

No es objetivo de este trabajo dar detalles de implementación, ni analizar los resultados obtenidos para una asignatura concreta. La orientación teórica que hemos dado al artículo es más general, y trata de mostrar la separación entre Evaluación Formativa y Evaluación Calificativa como una nueva perspectiva al modelo enseñanza-aprendizaje que puede aplicarse a diferentes asignaturas. En todo caso, sí pueden apuntarse algunas conclusiones, obtenidas a partir de encuestas de opinión (especialmente sobre la autoevaluación, con la que llevamos más tiempo experimentando):

- Los alumnos perciben la autoevaluación como algo positivo. En particular, creen que les permite mantenerse puntualmente informados.
- Los alumnos aprenden pronto la mecánica. El porcentaje de autoevaluaciones que deben ser corregidas es pequeño.
- El tiempo de dedicación de los profesores a la supervisión es asumible. En el caso de la asignatura LP, estamos hablando de 1 hora a la semana, para un grupo de 40 alumnos.
- En versiones previas de la asignatura, la nota de autoevaluación contaba para la nota final. Esto producía que los alumnos pusieran mucho énfasis en cálculo de la nota, y menos en la identificación de los errores cometidos. Actualmente esta nota no cuenta para la nota final y se da más peso a la calidad del informe en cuanto a la justificación de diferencias (no solo de los errores) entre la solución del estudiante y la solución oficial.

Referencias

[1] A. W. Chickering y Z. F. Gamson, Seven Principles for Good Practice in Undergraduate Education, http://www.hcc.hawaii.edu/intranet/committees/FacDevCom/guidebk/teachtip/7princip.htm

[2] A.W. Bangert, Peer Assessment: A Win-Win Instructional Strategy for Both Students and Teachers, *J. Cooperation & Collaboration in College Teaching*, Vol. 10, No. 2, p. 77.

[3] B.S. Bloom et al, Taxonomy of Educational Objectives: Handbook I, Cognitive Domain. Nueva York: David McKay, 1956.

[4] M. Valero-García y J.J. Navarro, Niveles de competencia de los objetivos formativos en las ingenierías, *VII Jornadas sobre la Enseñanza Universitaria de la Informática JENUI 2001*, p. 149

[5] Ideas and Rubric, Instructional Intranet, Chicago Public Schools. http://intranet.cps.k12.il.us/Assessments/Ideas_and_Rubrics/ideas_and_rubrics.html

Funciones, capacidades, habilidades y actitudes de los informáticos en las empresas:¿cómo podemos mejorar?

Ferran Virgós Bel
Dep. de Lenguajes y Sistemas Informáticos (LSI)
Escuela Universitaria de Ingeniería Técnica Industrial de Barcelona (EUETIB)
Universidad Politécnica de Cataluña (UPC)
Barcelona
Ferran.Virgos@upc.edu

Resumen

En el trabajo presentamos un modelo para la identificación de *gaps* (factores críticos no cubiertos suficientemente) en la función TSI (Tecnologías y Sistemas de Información) en las organizaciones. El modelo parte de una identificación de actividades básicas de la función TSI apoyada en la bibliografía, aunque posteriormente puedan actualizarse. La herramienta resultante es aplicable como evaluación "formativa" en una organización concreta, sector o área geográfica, con el objetivo de mejorar su planificación. Pero también puede ser de gran utilidad para reorientar la estructura curricular de los estudios en informática o, simplemente, la metodología didáctica aplicable. Para avanzar en este sentido, se realiza un estudio empírico concreto, con muestras procedentes de dos entornos geográficos (y universitarios) distintos. Las conclusiones nos permiten la enumeración de los 12 *gaps* más importantes.

1. Antecedentes y objetivo

A menudo hemos asistido, en numerosos foros, a largas e infructuosas discusiones entre hombres de empresa y profesores universitarios, donde los primeros se quejaban de que la universidad no forma a sus estudiantes con la base necesaria para enfrentarse a la realidad del mundo empresarial. Por el contrario, los representantes universitarios suelen aducir que su misión debe centrarse en ofrecer una base científica de tipo general que facilite una fácil adaptación posterior.

Seguramente a ninguna de las posiciones le falta su razón, aunque parte de ésta tenga su origen en un cierto egoísmo o comodidad, pues a los empresarios les encanta encontrar profesionales formados en resolver "exactamente" sus necesidades, mientras los profesores, demasiado a menudo, tenemos tendencia a explicar "lo que nos gusta", o simplemente "lo que sabemos".

Pero Universidad y Empresa están condenados a entenderse y, recientemente, los rectorados y decanatos se han dado cuenta de la importancia de contar con la opinión del mundo empresarial donde sus estudiantes, más tarde o más temprano, deberán insertarse. Las asignaturas optativas son un buen camino en este viaje, pero, en cualquier caso, este objetivo de confluencia no es fácil, entre otras cosas porque no es único. Piénsese en las diferencias entre sectores industriales, zonas geográficas, tamaño de empresa e incluso posicionamiento estratégico de las mismas.

En este contexto, el objetivo básico de nuestro trabajo se centra en proponer un método conducente a la "identificación" de *gaps* en las funciones de los profesionales de TSI en la empresa; de hecho, en cualquier organización. El método debería poder ser utilizado al nivel que se deseara, ya sea empresa individual, sector o área regional, siendo, en consecuencia, una herramienta de tipo general.

Será necesario partir de la identificación de unos factores "deseables" en la función TSI. Estos factores se utilizarían como base para construir una herramienta de consulta de tipo operativo. En cualquier caso, la flexibilidad del modelo deberá posibilitar la actualización y/o adaptación de los factores para sucesivas aplicaciones.

Finalmente, a título orientativo, se trataría de aplicar la herramienta construida a una determinada población, con el objetivo de generar una primera lista de *gaps* críticos que nos permitiría el establecimiento de unas conclusiones preliminares.

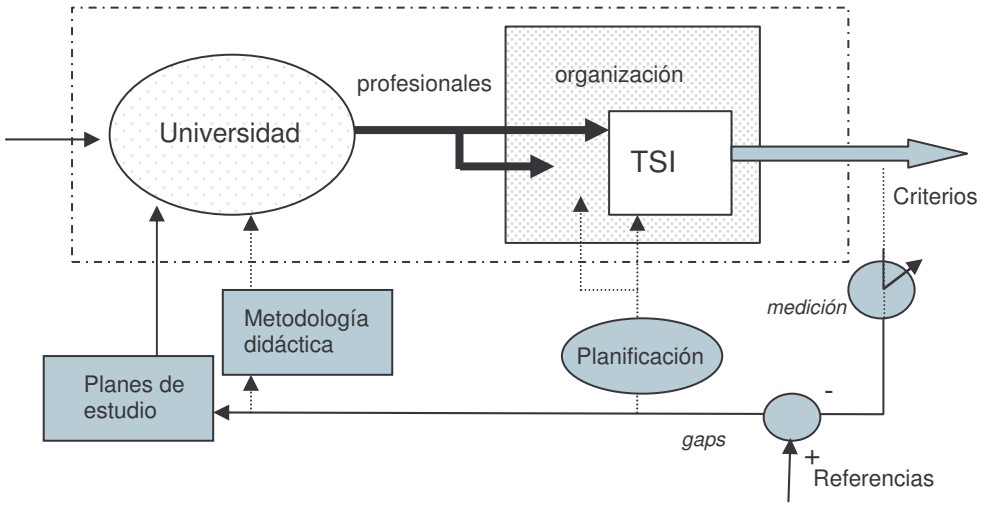


Figura 1: el modelo

2. El modelo de evaluación formativa

Las universidades forman profesionales, basándose en unos planes de estudio y unos métodos didácticos. Estos profesionales se incorporan a las organizaciones para contribuir a alcanzar los objetivos de éstas.

Un modelo de evaluación "formativa", pretende valorar las salidas del sistema seleccionadas como criterios, comparando con unas referencias deseables, identificando *gaps* que puedan utilizarse en mejorar la planificación de la organización.

Por extensión, si transmitimos esta información a las instituciones universitarias, éstas podrán usar la misma para mejorar sus planes de estudio y/o adaptar su metodología. Este es nuestro planteamiento, recogido en la Figura 1.

Para continuar adelante con este modelo y poder aplicarlo, necesitamos conocer cuáles son los "criterios" (actividades, tareas o funciones) que nos permitirían certificar la excelencia de la aportación de TSI al óptimo funcionamiento de la organización.

3. Identificando las funciones básicas de TSI en las organizaciones

Llegados a este punto, parece lógico no pretender inventar la rueda y detenernos para realizar una revisión bibliográfica. Repasando nuestros archivos, encontramos que ya en 1987, Brancheau & Wetherbe, como consecuencia de un estudio realizado en USA [1], extraen como conclusión una lista de diez puntos, a modo de decálogo, de lo que se pide a un director TSI. Su particular decálogo incluye los puntos siguientes:

BW01. Mejorar la Planificación estratégica.
BW02. Utilizar las TSI para obtener ventaja competitiva.
BW03. Facilitar el aprendizaje organizativo y el uso de las TSI.
BW04. Mejorar el conocimiento del papel/ contribución de las TSI.
BW05. Alinear la organización de la función de TSI con la de la empresa.
BW06. Facilitar y gestionar los entornos de usuario final (IUF).
BW07. Promocionar el uso efectivo de la información como recurso.
BW08. Desarrollar la arquitectura de Información.
BW09. Medir la efectividad y productividad de la función TSI.
BW10. Integrar la función tratamiento de datos convencional (Centro de Proceso de Datos, CPD), informática de usuario (IUF), fabricación y comunicaciones.

En un trabajo posterior, realizado por Toni Moynihan [2], se identifican once aspectos clave en relación a la función TSI. En este caso, la lista completa es:

TM01. Compartir datos entre sistemas y departamentos.
TM02 Optimizar la calidad de la Planificación en TSI y su relación con la planificación global del negocio.
TM03. Impulsar la adopción de los estándares TSI de la empresa por parte de las divisiones.
TM04. Adoptar un estilo de justificación para aprobar e impulsar proyectos.
TM05. Conseguir un compromiso de colaboración de los usuarios en el desarrollo de los proyectos.
TM06. Utilizar las técnicas del estado del arte para soportar los procesos clave.
TM07. Utilizar las TSI para obtener ventaja competitiva.
TM08. Asegurar la calidad de formación y conocimientos del personal propio de TSI.
TM09. Optimizar el nivel de uso de la informática de usuario final y tecnología de comunicaciones.
TM10 Tener como objetivo la rapidez en la implantación de nuevos sistemas y sensibilidad del departamento de TSI al respecto.
TM11. Ofrecer la contribución de TSI a mejorar (en forma sensible) la eficiencia de la organización.

Más recientemente, en un trabajo publicado en la *Sloan Management Review* en 1996 (y, posteriormente, en Harvard-Deusto), el prestigioso John F. Rockart, director hasta el año 2000 del *Centre for Information Systems Research (CISR)* del *Massachussets Instiute of Technology (MIT),* junto a otros colaboradores, realizaban un estudio similar [3] en relación a nuevas prácticas de gestión de TSI en 50 empresas y un estudio comparativo de organizaciones de 4 países (incluyendo USA y Japón). El resultado del trabajo se plasma en ocho ejes, directrices, normas o actividades esenciales para la gestión de TSI a finales del siglo XX y principios del XXI. Los principios identificados son:

Ra01. Conseguir una coordinación estratégica de doble vía.
Ra02. Desarrollar relaciones eficaces con la Dirección de línea.
Ra03. Proporcionar y poner a punto nuevos sistemas.
Ra04. Crear y gestionar la infraestructura.
Ra05. Dotar de nuevas capacidades a la organización de TSI.
Ra06. Gestionar asociaciones con proveedores.
Ra07. Obtener buenos resultados.
Ra08. Rediseñar y gestionar la organización federal de TSI.

4. Nuestra propuesta de factores

A partir del análisis de los trabajos anteriores, y su actualización de acuerdo con nuestra propia experiencia y las consultas realizadas, construimos una nueva relación incluyendo 20 propuestas de factores o funciones básicas de TSI (la lista no pretendía ser tomada como algo cerrado sino, más bien, como una relación abierta de factores):

FV01. Encargarse de la explotación y mantenimiento del sistema transaccional básico.
FV02. Crear y gestionar la infraestructura globalmente (en especial redes, comunicaciones y seguridad), controlando los costes.
FV03. Proporcionar y poner a punto nuevos sistemas de desarrollo propio (dep. TSI).
FV04. Proporcionar e integrar paquetes estándar y *ERPs (Enterprise Resource Planning).*
FV05. Potenciar y gestionar la subcontratación en TSI (*outsourcing*).
FV06. Formar al usuario para potenciar el desarrollo autónomo.
FV07. Estar siempre al día en tecnología.
FV08. Facilitar las tareas de los usuarios en su puesto de trabajo, incluyendo soporte en ofimática y otros complementarios como Internet, videoconferencia, etc.
FV09. Impulsar la formación empresarial de los técnicos para que “entiendan” los procesos de negocio.
FV10. Gestionar el departamento de TSI como una empresa, intentando obtener buenos resultados y difundir esa información para ganar credibilidad.
FV11. Impulsar y dar soporte a la asociación con clientes y proveedores para integrar la cadena de valor extendida.

FV12. Conseguir la coordinación estratégica con la propia estratégica organizativa (no se trata sólo de colaborar con la estrategia sino ayudar a definirla).
FV13. Informar y formar a la alta dirección (asesoría).
FV14. Resucitar los antiguos "Comités de Dirección informática" impulsando la colaboración de todos en un objetivo común.
FV15. Añadir valor a los productos, ayudando a conseguir y mantener la fidelidad del cliente.
FV16. Dar más soporte a la toma de decisiones con sistemas basados en *DataWarehouse, data-mining* y/o inteligencia artificial .
FV17. Buscar la rapidez en la respuesta a las peticiones de sistemas.
FV18. Impulsar la organización "federal" (equilibrio entre centralización y descentralización en las decisiones y explotación informáticas).
FV19 Impulsar el desarrollo genérico del concepto de *eBusiness.*
FV20 Impulsar y dar soporte a la gestión del conocimiento en la empresa.

A efectos de verificación, realizamos una tabla para recoger una aproximación de la cobertura que ofrece nuestra lista de factores a los 3 conjuntos de las referencias bibliográficas consideradas.

5. La herramienta de medición

Nuestra intención era utilizar los 20 factores recogidos en el anterior epígrafe como "criterios" (elementos de la salida del sistema a medir) según el modelo global representado en la Figura 1. Pero no disponiendo de una estimación de los valores de las "referencias", pensamos que nuestra medición debía, asimismo, evaluarlas. Por ello, decidimos recoger no solamente la "realidad" sino también la referencia deseable. En consecuencia, para cada uno de los 20 factores decidimos proponer dos preguntas:

- ¿Debería hacerse (DH)?
- ¿Se hace realmente (HR)?

Para realizar la pregunta decidimos utilizar una escala cualitativa *Likert* de 7 niveles con el cuarto como elemento neutro:

1. totalmente en desacuerdo (-3)
2. bastante en desacuerdo (-2)
3. más bien en desacuerdo (-1)
4. neutro (0)
5. más bien de acuerdo (+1)
6. bastante de acuerdo (+2)
7. totalmente de acuerdo (+3)

Esta parte de la consulta nos permitiría conocer la opinión del encuestado en cuanto al valor de referencia (Debería Hacerse, DH) y su opinión sobre la situación real en su organización (se Hace Realmente, HR).

Estos datos aportarían una doble utilidad:

- En efecto, el examen de los valores medios de DH, debería permitirnos detectar el nivel de "acuerdo" con cada factor y, en consecuencia, justificar la construcción de una lista que podría tener una lectura "normativa", en el sentido de actuar a modo de "*check-list*" en el diseño o la auditoria de cualquier organización.
- Pero, además, la "diferencia" entre las evaluaciones (DH - HR) de cada factor nos define un *gap* que constituye un claro "diagnóstico formativo", al identificar un déficit sobre el que deberíamos actuar.

Pero sólo con estos datos, no teníamos ninguna información de la "criticidad" del factor ya que, en efecto, podría considerarse respecto a un determinado factor que resultaría positivo y no se está aplicando pero, al mismo tiempo, admitir que este hecho es poco relevante.

Para tener información complementaria de esa criticidad, decidimos preguntar una tercera opinión al encuestado en relación a la importancia o peso del factor, en una escala de 5:

Es crítico/importante hacerlo
0. en absoluto
1. algo importante
2. bastante importante
3. muy importante
4. absolutamente esencial

Este tercer coeficiente (criticidad o peso, P) será importante tanto para matizar el acuerdo con los principios (DH), identificando los más importantes, como para identificar los *gaps* más críticos.

Debe notarse que la herramienta así construida ya es terminal (y la primera propuesta de nuestro trabajo) y puede utilizarse en cualquier

organización concreta, constituyendo el resultado un auténtico "diagnóstico *ad-hoc*" de la misma. Pero también puede utilizarse como una herramienta base de un estudio general. Esta fue nuestra opción para continuar adelante.

Automatización	2,78%	Ingeniería	5,56%
Automoción	5,56%	Papelero	2,78%
Banca-financiero	13,89%	Sector publico	2,78%
Construcción	8,33%	Servicios	5,56%
Distribución	2,78%	Software	2,78%
Educación	5,56%	Telecomunicaciones	13,89%
Electrónica	2,78%	ND (no disponible)	5,56%
Farmacéutico	2,78%	Otros	16,67%

Tabla 1: distribución de la muestra por sectores

6. Mecánica de la encuesta y su población

Dada la inexistencia de ayuda externa, optamos por buscar un planteamiento con un mínimo coste. La solución que encontramos consistió en solicitar las respuestas a los alumnos de un Master en Dirección y Organización de Empresas de la EUETIB (UPC) y en un MBA ofrecido por la Universidad de Andorra (UdA) en colaboración con *Les Heures* (master de tres universidades catalanas).

Con el debido consentimiento de los directores de ambos cursos, los alumnos recibieron la encuesta acompañada de una carta de presentación que, al mismo tiempo explicaba la manera de responder a la misma.

Se recogieron un total de 36 respuestas válidas (24 de los alumnos del Master UPC y 12 de los alumnos del MBA de la UdA). Algunos de los encuestados dejaron alguna pregunta en blanco, cosa que se ha tenido en cuenta, lógicamente, en el momento de realizar el tratamiento de los datos

Los dos conjuntos de datos (Barcelona-UPC y Andorra-UdA) presentaron diferencias muy poco significativas, por lo que se desestimó la posibilidad de un tratamiento separado optando por tratar la muestra como un todo.

A las 20 preguntas correspondientes a la evaluación de los factores, se añadieron dos preguntas más. Una referente al número de niveles hasta la dirección general (Director General =0) para conocer el "nivel" del encuestado en su organización. Otra, relativa al número de años de experiencia del encuestado (en cualquier empresa). También se añadió una pregunta "abierta" para que los encuestados pudieran proponer otros factores, pero solamente uno de ellos utilizó esa posibilidad.

La muestra constituía una buena representación sectorial (ver Tabla 1). La situación jerárquica de los encuestados en la empresa se repartía entre 1 y 4 niveles hasta la dirección general (Tabla 2, a continuación).

0	2,78%
1	25,00%
2	13,89%
3	25,00%
4	19,44%
más o ND (no disponible)	13,89%

Tabla 2: Niveles hasta la dirección general

También los años de experiencia ofrecía un buen abanico, situándose, básicamente entre 2 y 20, con la mayoría entre 4 y 10, como corresponde al perfil de este tipo de Masters (ver Tabla 3).

hasta 2 años	11,11%
entre 2 y 4	13,89%
entre 4 y 6	19,44%
entre 6 y 8	13,89%
entre 8 y 10	19,44%
entre 10 y 15	11,11%
entre 15 y 20	5,56%
más de 20	2,78%
no especificado	2,78%

Tabla 3: Años de experiencia

	DH	HR	P	DH x P	GAP	desv	inf	sup	GAP x P
1	1,88	0,76	3,12	5,86	1,12	1,39	0,65	1,60	3,50
2	1,94	0,60	3,14	6,11	1,34	1,45	0,84	1,84	4,22
3	1,29	0,06	2,85	3,69	1,24	1,88	0,59	1,88	3,52
4	1,10	0,23	2,84	3,11	0,87	1,80	0,25	1,49	2,47
5	0,49	-0,06	2,06	1,00	0,54	2,50	-0,32	1,40	1,12
6	2,19	-0,36	3,19	7,01	2,56	1,70	1,97	3,14	8,16
7	2,36	0,42	3,19	7,54	1,94	1,37	1,47	2,42	6,21
8	2,29	-0,17	3,09	7,05	2,46	1,69	1,88	3,04	7,58
9	2,11	-0,61	2,83	5,98	2,72	1,83	2,09	3,35	7,71
10	1,20	-0,63	2,40	2,88	1,83	1,38	1,35	2,30	4,39
11	1,47	-0,64	2,72	4,01	2,11	1,58	1,57	2,65	5,75
12	1,65	-1,00	2,65	4,36	2,65	1,41	2,16	3,13	7,01
13	2,17	-0,08	3,03	6,56	2,25	1,30	1,81	2,69	6,81
14	0,37	-1,20	1,97	0,73	1,57	1,77	0,96	2,18	3,10
15	2,00	-0,17	2,92	5,83	2,17	1,48	1,66	2,68	6,32
16	1,12	-1,24	2,24	2,51	2,36	1,58	1,82	2,91	5,30
17	2,08	-0,47	2,89	6,02	2,56	1,76	1,95	3,16	7,38
18	1,31	-0,66	2,50	3,28	1,97	1,79	1,36	2,58	4,92
19	1,86	-0,67	2,92	5,43	2,53	1,63	1,97	3,09	7,37
20	2,31	-0,60	3,14	7,27	2,91	1,56	2,38	3,45	9,16

Tabla 4: Tabulación general de los resultados

7. Los resultados

En la tabla 4, se recogen los resultados del tratamiento previo de las encuestas:

- En las tres primeras columnas se presentan las medias de las correspondientes respuestas (DH, HR y el Peso), mientras en la cuarta, hemos añadido por cálculo el producto (DH x P) para "definir" un nuevo concepto de importancia que no solamente contemple el acuerdo, siendo también la criticidad del factor. En la Figura 2 recogemos la representación gráfica de estas 4 variables.

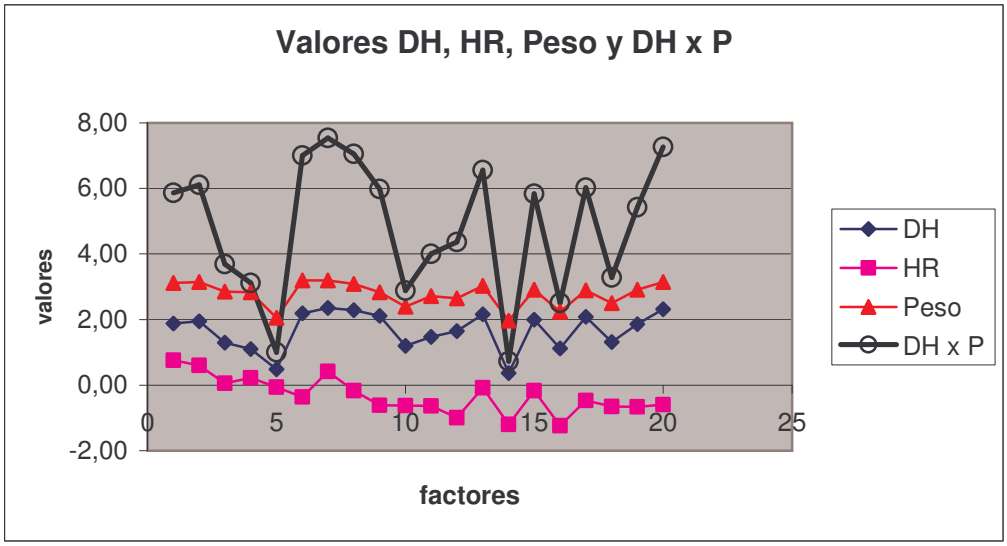


Figura 2: Representación gráfica de los resultados

Si procedemos a una ordenación de los factores según la cuarta columna (DH, ponderado), obtendremos la primera conclusión, de tipo general, de nuestro trabajo: aquellas funciones que los directivos de línea, esperan de los profesionales de TSI.

Indirectamente, por aplicación del modelo de la Figura 1, las conclusiones deberían resultar de interés para los responsables de crear los planes de estudios de los futuros profesionales.

	DHc
7	2,36
20	2,31
8	2,29
6	2,19
13	2,17
9	2,11
17	2,08
15	2,00
2	1,94
1	1,88
19	1,86
12	1,65
11	1,47
18	1,31
3	1,29
10	1,20
16	1,12
4	1,10
5	0,49

	DHc x P
7	7,54
20	7,27
8	7,05
6	7,01
13	6,56
2	6,11
17	6,02
9	5,98
1	5,86
15	5,83
19	5,43
12	4,36
11	4,01
3	3,69
18	3,28
4	3,11
10	2,88
16	2,51
5	1,00

Tabla 5: Ordenación de DH y DH x P

- La quinta columna de la Tabla 4 recoge la media de las diferencias (GAP= DH-HR, donde pequeñas diferencias corresponden al redondeo). La sexta columna contiene los valores de la desviación tipo de las diferencias. Esta desviación se ha usado para calcular los límites inferior y superior (columnas 7 y 8) de los intervalos de confianza del 95% calculados con la t de *Student* (α= 0,05). Puede verse como todos los factores a excepción solamente del 5, presentan intervalos de confianza que no incluyen el cero. Es decir, detectan *gap*.
- Finalmente, en la última columna, se recoge el GAP ponderado. En la Figura 3 se representan gráficamente estos *gaps* que aparecen, ya ordenados, en la Tabla 6.

20	9,16
6	8,16
9	7,71
8	7,58
17	7,38
19	7,37
12	7,01
13	6,81
15	6,32
7	6,21
11	5,75
16	5,30
18	4,92
10	4,39
2	4,22
3	3,52
1	3,50
14	3,10
4	2,47
5	1,12

Tabla 6: Los 12 *gaps* más críticos

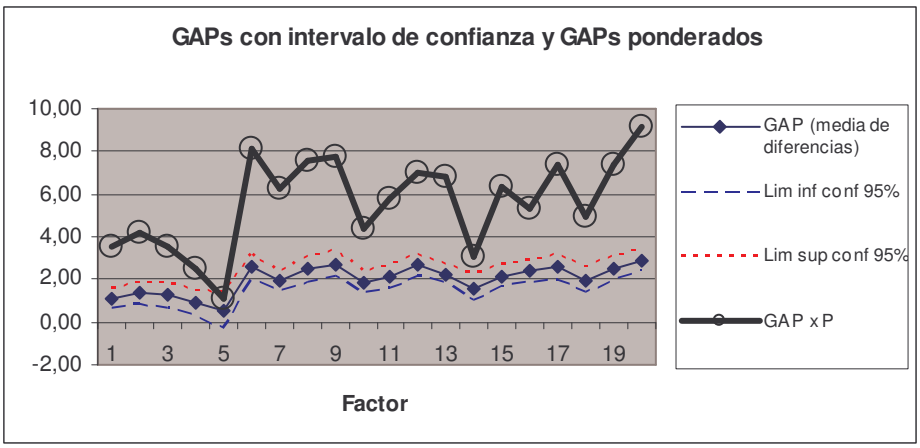


Figura 3: Representación gráfica de los GAPs (con su intervalo de confianza 95% y GAPs x P

8. Conclusiones

La Planificación y el control de gestión de cualquier actividad exigen la definición previa de unos objetivos y un seguimiento continuo de los niveles realmente alcanzados. El conocimiento de la diferencia o *gap* constituye un elemento básico para el éxito final del proceso.

Nos hemos planteado ¿qué se exige en una organización a su departamento de TSI? y hemos propuesto un modelo de evaluación formativa y herramienta asociada que pueden ser de gran utilidad para realizar un estudio específico de las necesidades de la función TSI de cualquier organización. Pero, además, la herramienta incluye la posibilidad de detectar los déficits o *gaps*, incluso identificando los más críticos. A titulo orientativo hemos aplicado la herramienta en un colectivo concreto para permitir extraer unas conclusiones preliminares

En particular, la Tabla 6 muestra una relación de los 12 *gaps* críticos detectados en nuestro estudio, que han resultado ser:

FV20. Impulsar y dar soporte a la gestión del conocimiento en la empresa.
FV06. Formar al usuario para potenciar el desarrollo autónomo.
FV09. Impulsar la formación empresarial de los técnicos para que "entiendan" los procesos de negocio.
FV08. Facilitar las tareas de los usuarios en su puesto de trabajo.
FV17. Buscar la rapidez en la respuesta a las peticiones de sistemas.
FV19. Impulsar el desarrollo genérico del concepto de *eBusiness*.
FV12. Ayudar a definir la estrategia organizativa.
FV13. Informar y formar a la alta dirección (asesoría).
FV15. Añadir valor a los productos, ayudando a conseguir y mantener la fidelidad de los clientes.
FV07. Estar siempre al día en tecnología
FV11. Dar soporte a la cadena de valor extendida.
FV16. Dar más soporte a la toma de decisiones.

Es curioso ver cosas como que los usuarios piden en primer lugar a los profesionales de las TSI que estén al día en tecnología (FV07) pero, al mismo tiempo, consideran que eso ya lo hacen y por tanto, como *gap* no aparece hasta la posición 10. Otros aspectos, en cambio, centran su preocupación. Un estudio realizado en un ámbito más general, puede ser de gran utilidad, también, en un sector industrial o zona geográfica; y la información resultante, de gran aplicación a la universidad, en cuanto a aportar directrices de orientación curricular y de metodología, ambos significativos en relación al proceso de Bolonia.

De hecho, es nuestra intención aplicar la herramienta a la valoración de habilidades, actitudes y, en general, elementos transversales de ese proceso. El libro blanco sobre el título de grado en ingeniería informática en España *(www.aneca.es)* ya lo hace, si bien no se orienta a detectar los *gaps.*

En cualquier caso, estamos valorando realizar algún ajuste en nuestra herramienta dado que ciertos revisores nos han hecho ver que no se entiende suficientemente la diferencia entre (DH) y (P). Quizás podría convertirse en "favorable a hacerlo (FH)" y peso (P), siendo más fácil entender que alguien puede ver como favorable que los presentadores de TV lleven corbata y, al mismo tiempo, considerar que eso no es importante, siendo más relevante que sean simpáticos o tengan buena voz. También, estamos planteando pedir directamente la valoración del *gap*, en lugar de calcularlo por diferencia, para conseguir mayor precisión.

Referencias

[1] [Brancheau & Weterbe, 1987] Brancheau, J.C. & Weterbe, J.C. *Key issues in Information Systems management.* MIS Quarterly (11, 1). March 1987.

[2] [Moynihan, 1990] Moynihan, Tony. *What Chief executives and senior managers want from their IT departament?.* MIS Quarterly. March 1990 .

[3] [Rockart et al, 1996]. John F. Rockart, Michael J. Earl, Jeanne W. Ross. *Eight Imperatives for the NEW IT organization*. Original de MIT Sloan Management Review (reproducido, posteriormente en Harvard Deusto). Fall 1996.

CEDI 2005

Innovación, Calidad y Evaluación Docente II

Decálogo para el profesor (de informática) novel

Faraón Llorens y Rosana Satorre
Dpto. de Ciencia de la Computación e Inteligencia Artificial
Universidad de Alicante
03080 Alicante
e-mail: {faraon,rosana}@dccia.ua.es

Resumen

En este artículo presentamos distintas reflexiones derivadas de nuestra preocupación por la docencia. Reflexiones que nos han llevado junto con alumnos de diversos cursos de formación pedagógica a elaborar una serie de consejos, que pensamos pueden ser de utilidad para todo aquel que se enfrente por primera vez ante sus alumnos.

1. Introducción

Está de moda el *know-how*, el saber *cómo* se hace algo. Pero ese saber hacer no está claro en la labor del profesor[1], e incluso parece descuidada en la labor del profesor universitario. Este 19 de marzo, día del padre, llegó a nuestras manos una postal con el texto "Convertirse en padre no es difícil. Ser un padre sí"[2]. Tener hijos no nos convierte en padres, del mismo modo en que poseer un piano no vuelve a su dueño en pianista. Esto, trasladado al asunto que nos ocupa, nos lleva a dos consideraciones, que son la base de las reflexiones de este artículo. La primera es que el hecho de tener alumnos no te convierte en profesor. La segunda, que convertirse en profesor no es difícil, pero *ser* un profesor, sí. Aunque encontrar trabajo o aprobar una oposición sea en muchos casos una verdadera carrera de obstáculos, no implica necesariamente que se esté preparado para enseñar. Pero es que la tarea no es fácil. No existe una receta que nos explique detalladamente cómo enseñar.

Con esta filosofía decidimos escribir unas reflexiones sobre ello y presentarlo en las Jornadas de Enseñanza Universitaria de Informática del año 2004 (JENUI 2004) [18], más preocupados en lo que nos aportaría el proceso de elaboración del mismo y en la sesión pública de debate, que en la redacción final del mismo. Un artículo que no pretendía dar normas de cómo dar una buena clase, pues no existen y una buena clase no se puede "normalizar". Pero algo se debe poder trasmitir a una persona que se acerca por primera vez a la tarea de enseñar. De hecho, los autores sí que somos parte preocupada e involucrada en la búsqueda de buenas fórmulas de enseñanza y aprendizaje y en ese sentido pretendíamos aportar como granito de arena, diversas ideas que hemos ido debatiendo durante muchos, muchísimos momentos, y que recogimos en un primer momento en forma de *decálogo del profesor novel* [10].

Además, en dicho escrito presentamos la experiencia que habíamos desarrollado con nuestros estudiantes del módulo de Didácticas Especificas (Informática) del CAP (Curso de Aptitud Pedagógica) impartido el curso 2003/2004 en la Universidad de Alicante. En unas primeras sesiones les incitamos a la reflexión sobre el proceso de enseñanza, en general y de la informática en particular. Posteriormente se vieron algunas herramientas que podían ayudarles en esa tarea. Y finalmente les pedimos que intentaran resumir en unos consejos su idea sobre la labor del profesor, en forma de normas a seguir. Su visión nos parece que puede sernos muy útil ya que tienen aún muy cercana su vida de estudiantes pero al mismo tiempo están planteándose ya su futuro como profesores.

Así que tras la experiencia del curso 2003/2004 y a partir de sus resultados decidimos seguir profundizando en el tema, aunque todo hay que decirlo, consideramos que son muchos los cursos y la experiencia a recoger para llegar a un satisfactorio nivel de profundidad. Con el artículo presentado conseguimos iniciar la reflexión sobre el profesor novel genérico, es decir, cualquier profesor novel de cualquier filosofía o campo.

[1] Utilizaremos el género masculino por facilitar la lectura del texto, aunque en todo momento estaremos haciendo referencia tanto a profesores como profesoras, alumnos como alumnas.

[2] Wilhelm Busch (1832-1908), poeta alemán.

Nuestra intención cuando empezamos era la de fijar pautas que ayudasen al profesor novel de informática pero las reflexiones que hicimos posteriormente nos llevaron a unos consejos generales. Así que de nuevo en las clases del CAP del curso 2004/2005 volvimos a lanzar la reflexión a nuestros estudiantes y en este artículo presentamos los resultados de diversos debates.

La estructura del artículo es la siguiente. Iniciamos este documento con una pequeña reflexión sobre qué es y cómo vemos el proceso enseñanza y aprendizaje, en general, para posteriormente particularizarlo a la enseñanza de la Informática. A continuación explicamos el diseño y desarrollo de la experiencia en clase, para pasar a enumerar tanto nuestra propuesta de decálogo para un profesor novel como las presentadas por nuestros estudiantes. Finalmente extraemos algunas conclusiones derivadas de este trabajo.

2. La Enseñanza

Como ya hemos comentado en la introducción estamos en la época del *know-how*, en la que importa y adquiere un papel relevante el saber *cómo* se hace algo. Pero ese saber hacer tan de moda en la técnica y la investigación no es usado regularmente en la docencia. Incluso personajes de prestigio dedicados a ella reconocen su impotencia y llegan a escribir "lo siento; después de muchos, muchísimos años de tratar de enseñar y tratar todo tipo de métodos diferentes, realmente no sé cómo hacerlo" [7]. Ello denota que no es fácil ni existe una fórmula mágica que nos diga *cómo* enseñar.

La enseñanza/aprendizaje es un proceso bipolar, donde en un extremo se encuentra la enseñanza cuyo protagonista principal es el profesorado y en el otro el aprendizaje cuyo protagonista principal es el alumnado. Está claro que ambos términos no son lo mismo, pero son dos caras de una misma moneda y por tanto indisolubles. A partir de este momento cuando utilicemos cualquiera de los dos vocablos, en realidad nos estaremos refiriendo al binomio enseñanza/aprendizaje (E&A). Ambos conceptos son complementarios y en función del punto de vista y dónde se ponga mayor énfasis aparecen distintos juegos de palabras (enseñar a enseñar, aprender a enseñar, enseñar a aprender, aprender a aprender) que iremos desarrollando a lo largo del artículo.

Pero las reflexiones sobre el proceso de enseñanza no son nuevas, "la primera finalidad de la enseñanza fue formulada por Montaigne: es mejor una mente bien ordenada que otra muy llena. /.../ Una mente bien formada es una mente apta para organizar los conocimientos y de este modo evitar su acumulación estéril" [12]. Nuestra labor como profesores ante nuestros estudiantes está limitada por el espacio y el tiempo. No debemos pretender transferirles todo lo que consideramos que deben saber simplemente por el mero hecho de contárselo. Debemos traspasar las barreras del tiempo y enseñarles a que quieran y puedan continuar aprendiendo al abandonar nuestras aulas, deben *aprender a aprender*. Más si cabe en nuestro campo, dónde el avance de la ciencia y la técnica, y en particular de la informática, es vertiginoso. Cuando los alumnos que ahora tenemos ante nosotros se conviertan en profesionales es probable que los instrumentos de que dispongan en el ejercicio de su actividad y las técnicas que empleen sean sensiblemente diferentes a las que nosotros hayamos podido describir. Lo que ahora importa, no es tanto poseer una información determinada, sino fundamentalmente haber adquirido la capacidad para descubrir y saber encontrar esa información. Concebido así el proceso educativo, la misión encomendada al educador cambia, pasando en gran medida a transformarse en un director y organizador de la situación de aprendizaje. Debemos estar preparados para *enseñarles a aprender*.

El papel del profesorado es crucial, es la piedra angular de la universidad, son los buenos profesores los que harán posible cualquier planteamiento original o innovador de la formación universitaria. Así, "los buenos profesores, *los maestros*, atraen el interés por aprender del estudiante más que el mejor texto, y tiene algo de mágico la manera en que la admiración personal por el *profesor sabio y bueno* se traslada a la materia que imparte, la convierte en fácil, más apasionante, y consigue hacer brotar la verdadera vocación" [11]. Los docentes deben hacer un esfuerzo para asimilar algunos conceptos fundamentales que se repiten continuamente, pero sobre los que nunca se reflexiona lo suficiente y,

tristemente, pocas veces tienen un claro reflejo en la actividad docente.

Uno de los aspectos claves en la labor docente es la *evaluación* [1,5]. Estamos tan convencidos de su importancia, que, en un escrito de extensión reducida como éste, vamos a dedicarle un párrafo en exclusividad. Los alumnos estudian aquello en lo que saben van a ser evaluados (¿entra en el examen?), convirtiéndose la evaluación en el centro del aprendizaje. Por tanto, si se cambian los métodos docentes utilizados sin modificar los métodos de evaluación, el esfuerzo está condenado al fracaso o tiene mucho menor impacto. Por ello debe existir una coherencia total de los criterios de evaluación con los elementos restantes de la planificación docente (objetivos, contenidos y actividades). Podíamos definir la evaluación como aquel proceso complejo que comprende la obtención, por medio de los más variados procedimientos, de información útil acerca de cualquier tema, que permitirá emitir juicios, y en consecuencia, tomar decisiones al respecto. Al hablar de evaluación debemos plantearnos cuatro cuestiones: ¿quién evalúa? ¿qué evaluar? ¿cómo evaluar? y ¿para qué evaluar? Según la respuesta a estas preguntas tendremos distintos tipos de evaluación, de forma que la diversidad en la evaluación enriquece el proceso de E&A. Se trata de concebir y utilizar la evaluación como un instrumento y recurso de aprendizaje que permita suministrar retroalimentación adecuada a los alumnos, y al propio profesor, contribuyendo a la mejora de la enseñanza.

Una reciente investigación dirigida por el profesor Álvarez Rojo [2] describe lo que sería un perfil deseable del profesor universitario, esquematizado en un profesional motivado y entregado a su trabajo docente, que desarrolla una metodología activa en el aula que permite al alumno ser el protagonista de su aprendizaje, que potencia el trabajo en grupo y la cooperación, la discusión y el debate, que conecta el estudio con la realidad y que sabe utilizar los recursos audiovisuales y tecnológicos a su alcance. Uno puede asustarse al ver la cantidad de cosas que debe saber y saber hacer un profesor universitario. Ya lo hemos repetido varias veces, enseñar no es fácil, pero su lado bueno, la dedicación a la enseñanza es muy gratificante.

Para finalizar este apartado introductorio añadiremos que el optimismo y la confianza en un mundo mejor deben acompañar al profesor; en palabras de Fernando Savater, "en cuanto educadores no nos queda más remedio que ser optimistas, ¡ay! Y es que la enseñanza presupone el optimismo tal como la natación exige un medio líquido para ejercitarse. Quien no quiera mojarse, debe abandonar la natación; quien sienta repugnancia ante el optimismo, que deje la enseñanza y que no pretenda *pensar* en qué consiste la educación. Porque educar es creer en la perfectibilidad humana, en la capacidad innata de aprender y en el deseo de saber que la anima, en que hay cosas que pueden ser sabidas y que merecen serlo, en que los hombres podemos mejorarnos unos a otros por medio del conocimiento" [14].

3. La Enseñanza de la Informática

Una primera pregunta que debemos plantearnos es ¿qué han de *saber* y *saber hacer* los profesores para favorecer un aprendizaje efectivo de los alumnos e impartir una docencia de calidad? Para la elaboración del saber didáctico es imprescindible la integración del conocimiento de la materia, del conocimiento de los procesos de E&A y del conocimiento de la práctica docente. Un buen profesor de informática además de tener un nivel adecuado de la misma debe saber transmitir este conocimiento a los demás [9]. Claramente hay un consenso generalizado entre el profesorado acerca de la importancia de un buen conocimiento de la materia a enseñar. Y los alumnos son extraordinariamente sensibles a este dominio de la materia por el profesorado. Evidentemente, una falta de conocimientos científicos constituye la principal dificultad para que los profesores afectados se impliquen en actividades formativas nuevas. La segunda dificultad para una actividad docente creativa, procede de aquello que los profesores ya sabemos, de lo que constituye el *pensamiento docente de sentido común*. Los profesores tienen ideas, actitudes y comportamientos sobre la enseñanza debidos a una larga formación ambiental durante el período en que fueron alumnos. Así, no debemos caer en situaciones que nos lleven a reducir el aprendizaje de las ciencias a ciertos conocimientos y, a lo sumo, algunas destrezas, olvidando aspectos históricos, sociales, actitudes,

etc. Debemos descartar para siempre la idea errónea de que enseñar es fácil.

Es habitual que los profesores de Informática sean titulados en Informática, o titulaciones similares (ya que la titulación de informática es relativamente reciente). Podríamos decir que es un requisito básico el dominio, incluso avanzado, de la materia. Pero pocas veces se tiene una formación en educación. La formación en educación queda relegada al profesorado de los niveles primarios del sistema educativo. Incluso, las investigaciones sobre E&A en las Facultades de Educación suelen centrase en los niveles Primario y Secundario, rara vez en el Universitario. Pero cada vez más, existen profesores de informática preocupados por conocer las teorías sobre educación y aplicarlas a la enseñanza de sus materias [15]. Apareciendo así, lo que podríamos llamar *Didáctica de la Informática*, aún incipiente en nuestro país, pero ya consolidada a nivel internacional ("Computer Science Education" – CSE). Hay titulaciones que ofertan como asignaturas optativas del currículo académico "Didáctica de …" y existen áreas de conocimiento específicas de tales campos (Didáctica de las Matemáticas, …). Hasta lo que nosotros conocemos, no hay nada similar en Informática.

No debemos confundir la *enseñanza de la informática* con la *informática en la enseñanza*. Son campos distintos, aunque muy relacionados, no tanto por sus contenidos sino por las personas interesadas en ellos. Brevemente hablaremos de los dos aspectos, aunque nuestro principal énfasis está en el primero.

Al querer abordar la enseñanza de la informática nos encontramos con una primera cuestión ¿qué es genérico a la enseñanza y qué es particular a la enseñanza de la informática? Es muy difícil separar los temas y determinar cuáles son específicos de la didáctica de la informática, pero es relativamente fácil adaptarlos al caso concreto de la informática.

Existen numerosos trabajos que afirman la conveniencia de la aplicación del *constructivismo* a la enseñanza de la Informática [3]. El constructivismo es una corriente metodológica que enfatiza la construcción de nuevo conocimiento y maneras de pensar mediante la exploración y la manipulación activa de objetos e ideas, tanto abstractas como concretas. En este sentido, considera tan importante o más que la presentación ordenada de un cuerpo de conocimientos, plantear situaciones problemáticas a cuyo estudio está asociado dicho conocimiento de forma que es el estudiante quien aprende al construir activamente significados, convirtiéndose en responsable de su propio aprendizaje. Para ello es básico contar con las preconcepciones del alumno (lo que hay en el cerebro del que va a aprender tiene importancia), ya que los conocimientos que pueden conservarse permanentemente en la memoria no son hechos aislados, sino aquellos muy estructurados y que se relacionan de múltiples formas con lo conocido.

Recientemente, especialistas en educación aducen que los aspectos cognitivos individuales deben ser complementados con aproximaciones sociales que tengan en cuenta el efecto sobre el estudiante de las interacciones, con los demás y con el entorno ("Situated Learning"). Es una teoría educativa que puede ser muy útil en la enseñanza de la Informática [4].

En cuanto al segundo aspecto, el papel de la informática, y en general de las tecnologías digitales, en la educación, lo que podríamos llamar *Informática Educativa*, está más desarrollado [16]. Términos como multimedia, simulaciones, enseñanza asistida por ordenador, redes de ordenadores, internet y enseñanza a distancia forman ya parte del vocabulario de los educadores. Las nuevas tecnologías han entrado de lleno en el mundo de la educación, pero el verdadero impacto del uso de las mismas no ha hecho más que empezar [6,13]. Los avances en las Ciencias Cognitivas nos han permitido conocer mejor cómo pensamos, cómo resolvemos problemas y cómo aprendemos. Los avances en la tecnología son continuos. La puesta en común de ambos permitirá desarrollar aplicaciones que puedan ayudar a los estudiantes a aprender mejor y que al mismo tiempo tengan un interfaz más humano y amable [8].

Pero el uso de las tecnologías digitales en sí mismo no aporta valor didáctico al proceso de E&A, sólo son una herramienta, aunque, desde luego, muy potente, al servicio de una idea educativa. El uso de la tecnología debe aportar al proceso de E&A un "valor añadido", que no se puede conseguir con los métodos alternativos existentes, llegando incluso al punto en que reformule el propio modelo educativo.

Una última reflexión antes de pasar al siguiente apartado. No debemos limitarnos a enseñar asignaturas relacionadas con la informática, sino que además debemos ser los motores del uso de la tecnología digital en nuestros centros. Pero, ¿por qué nosotros? Porque conocemos la tecnología (y no le tenemos miedo) y nos dedicamos a la enseñanza.

4. Explicación de la experiencia

Este es el tercer curso en que los autores de esta ponencia impartimos clases en el CAP (Curso de Aptitud Pedagógica) organizado por el ICE (Instituto de Ciencias de la Educación) de la Universidad de Alicante. Somos profesores del módulo de Didáctica Específica para los titulados en Informática [17]. Pese a llevar bastantes años enseñando, ésta era nuestra primera experiencia en lo que podríamos llamar *enseñar a enseñar*. Curiosamente nos hemos encontrado con alumnos a los que hacía pocos años les estábamos enseñando informática, pero a los que ahora les debíamos enseñar a enseñarla (que no es lo mismo). Ellos acudían al curso porque querían *aprender a enseñar*, ya saben suficiente informática y necesitan saber trasmitirla.

En este sentido, el CAP sería lo más cercano que conocemos a un curso de *¿cómo aprender a dar clases?* o, en este caso más específico, a un curso sobre *Didáctica de la Informática*. El objetivo principal de este módulo es el de *enseñar a enseñar informática*. Nuestro alumnos son ya titulados universitarios, por lo tanto ya saben bastante informática, pero posiblemente no se les haya hablado explícitamente de la manera de enseñarla y ellos no hayan reflexionado detenidamente sobre el proceso de E&A. Se trata pues de ayudarles en estas reflexiones y mostrarles algunas técnicas y herramientas que les pueden ayudar en su futura labor docente. Y para ello tratamos tres aspectos: la enseñanza en general, la enseñanza de la informática en particular y la integración de la informática en la enseñanza.

En esa labor, y durante la segunda edición del módulo (curso 2003/2004) nos planteamos poder condensar las reflexiones generales sobre E&A, al estilo de las del apartado segundo, en 10 consejos que les daríamos a un compañero que se acercara a nosotros para solicitarnos orientación ante sus primeras clases. El limitarlos a diez, nos obliga a un trabajo de síntesis y de centrarnos en aquellos aspectos que consideramos básicos y que a la vez hemos intentado que sean útiles. Nuestro "decálogo para el profesor novato" aparece en el siguiente apartado. Esperamos que os sea provechoso, o al menos interesante, no porque lo suscribáis al cien por cien, sino porque os incite a la reflexión. Cada profesor escribiría su propio decálogo, que sería personal y diferente al de los demás. Esa pluralidad enriquece el proceso de enseñanza.

Una vez que redactamos nuestro decálogo quisimos dar un paso más. No se lo mostramos a los estudiantes hasta el final de la clase, y la actividad que ellos debían hacer hasta ese momento era redactar su propio decálogo, el del grupo de clase. La operativa, por tanto, quedaba del siguiente modo, en pequeños grupos escribirían cuatro consejos para un profesor que empieza. Una vez redactados los cuatro consejos juntábamos los de todos los grupos en un único documento, obteniendo 20 consejos. Entre todos (gran grupo), por consenso, debían eliminar o agrupar hasta reducirlos a diez consejos, que son los que finalmente suscribiría el grupo. Como impartimos dos grupos, en el apartado 6 aparecen los decálogos de cada uno de ellos, tal como quedaron al finalizar la clase.

Tras esta primera experiencia y la exposición pública en las Jenui2004, durante el tercer año de impartición del CAP (curso 2004/2005) decidimos reflexionar sobre los consejos del curso anterior e intentar ampliar, matizar o especializar los consejos hacia el docente en informática. Para ello, les presentamos los resultados del curso pasado, es decir, los consejos propuestos por sus compañeros y les animamos a discutir en pequeños grupos sobre estos consejos y sobre su aplicación en el campo de la docencia en informática.

El análisis de los resultados puede ser interesante ya que tenemos dos puntos de vista distintos pero complementarios: por un lado la experiencia reflejada en el decálogo de los profesores y por otro la visión de la inmediatez en la situación de los estudiantes (hace escaso tiempo eran alumnos y en un futuro cercano serán profesores). Al tratarse de un curso del CAP, deberíamos asumir que nuestros estudiantes están motivados por la docencia y habrán reflexionado sobre ella, ya que parece que la han elegido como una opción para su futuro profesional. ¿O

únicamente la eligen como salida fácil y cómoda? En la presentación del módulo, les hicimos presentarse también a ellos y, tristemente, esa era la postura mayoritaria. En los siguientes apartados mostramos los resultados obtenidos.

5. Un Decálogo para el profesor novel

Conscientemente, con premeditación y alevosía, hemos elegido los títulos de este apartado y del artículo. Hablamos de "*un* decálogo para el profesor novel" porque cada profesor podrá elaborar *su* decálogo, que se parecerá a éste o diferirá totalmente; pero lo que aparece a continuación es *nuestro decálogo*. Hablamos de "decálogo para el profesor (de informática) novel", poniendo el calificativo de informática al profesor entre paréntesis. Aunque nuestra pretensión era hacer un segundo decálogo específicamente para informática, no lo hemos hecho y nos hemos limitado a complementar el decálogo inicial. Lo hemos hecho porque por un lado la definición de decálogo[3] nos ha permitido liberarnos de la autolimitación inicial impuesta de diez consejos; segundo porque, como ya hemos dicho, no se puede separar la enseñanza de la enseñanza de la informática, se puede particularizar, complementar, pero no son dos cosas excluyentes y distintas. Así, el decálogo elaborado por nosotros ha quedado reflejado como sigue:

No existe una fórmula mágica que nos diga cómo enseñar
Enseñar no es fácil ni existe una fórmula mágica que nos diga cómo enseñar. "Lo siento; después de muchos, muchísimos años de tratar de enseñar y tratar todo tipo de métodos diferentes, realmente no sé cómo hacerlo".

Para enseñar no basta con saber
No siempre el que más sabe de una materia es el que mejor la enseña. No basta con conocer la materia que impartes, hay que saber transmitirla, comunicar, motivar a los estudiantes, ilusionar, enganchar …

Ya tienes alumnos, ahora debes convertirte en un profesor
Seguro que ya sabes mucho de tu campo de conocimiento, llevas años formándote para ello. Pero debes prepararte para ser profesor. Existen toda una serie de teorías de la educación. Aunque no puedas (debas) ser un experto en ellas, acércate a conocerlas, te ayudarán mucho en tu trabajo. El constructivismo y el aprendizaje basado en proyectos, entre otras, son metodologías muy adecuadas para la enseñanza de la informática.

Aprende el que trabaja
No se aprende por ósmosis, por contagio. De oírte hablar y de verte hacer ejercicios no se aprende. El aprendizaje es un proceso en el que los estudiantes deben participar activamente, construir sus conocimientos de forma significativa.

Los "maestros" dejan marca
El buen profesor, el "maestro", perdura en la memoria de sus alumnos. Inexorablemente el profesor se encuentra con las limitaciones espaciales y temporales en su labor: un horario determinado, unos determinados días y en un aula concreta. El buen maestro consigue que sus enseñanzas traspasen el aula, que hablen de ello fuera de las clases y que pervivan en el recuerdo de los estudiantes.

La curiosidad favorece el aprendizaje
Si despiertas en tus alumnos la curiosidad, sentirán mayor necesidad de aprender. Realmente se empieza a conocer algo cuando empezamos a cuestionárnoslo, a hacernos preguntas. El profesor debe transmitir su entusiasmo por aquello que se está estudiando. Dicho entusiasmo es detectado por el alumnado y despierta el interés por la materia y las clases.

Establece claramente las reglas de juego
Establece claramente, e incluso puedes negociar con ellos, las reglas de juego al principio del proceso: contenidos, criterios de evaluación, normas de trabajo, mecanismo de comunicación, … Mantén las promesas, no debes engañarles ni ir cambiando de opinión sobre la marcha. Sé honesto con ellos.

Eres su profesor, no su "colega"
Como hace muy poco que tú estabas al otro lado de la barrera, puedes sentirte tentado a querer ser el "colega" de tus alumnos. Podrás ser más

[3] **decálogo**. (Del lat. decal_gus, y este del gr. ________). 1. m. Conjunto de los diez mandamientos de la ley de Dios. 2. m. Conjunto de normas o consejos que, aunque no sean diez, son básicos para el desarrollo de cualquier actividad. (*Real Academia Española*)

accesible, más cordial, pero no olvides que ahora eres su profesor. Ten en cuenta que tarde o temprano tendrás que evaluarlos y emitir una nota, y deberás ser lo más objetivo, justo e imparcial posible.

Aprovecha los recursos docentes y la experiencia de otros que te han precedido
Ni eres el único ni el primero que realizas esa tarea. Casi seguro que hay otros profesores que han pasado por donde tú te encuentras ahora. Pídeles consejo, aprovecha su experiencia. Puedes encontrar recursos didácticos que te ayudarán. Los recursos tecnológicos son un complemento, una ayuda, pero sin contenido no sirven para nada. Son un medio, no un fin en sí mismos. Úsalos adecuadamente y de forma efectiva.

A tus estudiantes les apasiona la tecnología; aprovéchate de ello
Un profesor de informática tiene una audiencia motivada predispuesta al uso de las tecnologías actuales. Pero el uso de las tecnologías digitales en la educación no son un fin en sí mismo, sino un medio para alcanzar el objetivo del aprendizaje, por lo que deben aportar un "valor añadido"

La comunicación no se limita a lo que dices
Los estudiantes no sólo escuchan tus palabras, en tu interacción con ellos, reciben continuamente información procedente de tu actitud, tu entusiasmo por la materia, tu forma de actuar, … Casi dos terceras partes de la comunicación entre las personas es no verbal y se transmite a través de los gestos, las expresiones y el lenguaje corporal.

Cuida la puesta en escena
El profesor debe ser un actor que se enfrenta a una audiencia y la puesta en escena son puntos a su favor. Debemos controlar la voz, cuidando la entonación de las distintas frases y hablando con seguridad. Los cambios de ritmo del discurso nos pueden ayudar a captar la atención, así como las pausas nos ayudan a remarcar los puntos clave. Es conveniente mantener el contacto visual con los asistentes; la mirada establecerá una relación amistosa con la audiencia. Evita las muletillas[4].

[4] **muletilla**. voz o frase que se repite mucho por hábito. (*Real Academia Española*)

No trates de impresionarlos, sino de que aprendan
Que tus alumnos queden impresionados te puede venir bien en el primer contacto con ellos, sin embargo, deben aprender, no quedarse boquiabiertos. Lo que quieres es que sientan que deben saber, que deben aprender la información que les vas a suministrar, no que queden abrumados ante tanto conocimiento.

6. Otros Decálogos para el profesor novel

Como ya hemos dicho, en el curso 2003/2004 la experiencia se aplicó a dos grupos. Aquí aparecen las dos propuestas[5], del grupo 1.1 y del grupo 1.2., sin entrar en explicaciones, únicamente las hemos enumerado pues aparecen desarrolladas en el artículo de JENUI 2004 [10]. Las propuestas del curso 2004/2005 se muestran como grupo 2.1 y grupo 2.2 y no se limitan a diez consejos, pues se les dejó total libertad para que en un tiempo limitado establecieran su decálogo.

6.1. Grupo 1.1

- *Motiva y haz participar a los alumnos.*
- *Sé distante al principio para que acabes teniendo buenas relaciones.*
- *No hagas monótonas tus clases.*
- *Nunca decir que eres novato.*
- *Evaluar continuamente.*
- *Llevar las clases bien planificadas y bien preparadas.*
- *Conocer a los alumnos lo antes posible.*
- *Cuidar el vestuario y la imagen.*
- *Elaborar una lista de preguntas frecuentes.*
- *Evitar los enfrentamientos directos.*

6.2. Grupo 1.2

- *No pretender saberlo todo.*
- *Preparar la clase.*
- *Dominio de la materia a impartir.*

[5] Tanto la redacción como el contenido se puede pulir. Nosotros no los hemos querido tocar, no hemos "cambiado ni una coma" a lo que se redactó en clase. Hay que tener en cuenta que se redactaron en la misma aula y en un tiempo aproximado de hora y media por grupo.

- *Conocer el entorno y replantear los objetivos en función de los alumnos.*
- *Paciencia y optimismo.*
- *Hacer una presentación inicial.*
- *Hacerte respetar.*
- *Implícate en el proyecto.*
- *Establecer claramente el programa y el sistema de evaluación al principio de curso.*
- *No imponer tu ideología en ciertos temas.*

6.3. Grupo 2.1

Sé generoso
Enséñales todo lo que conoces, no te guardes nada por miedo a que sepan más que tú.

Marca objetivos y sé coherente con los objetivos
Una vez fijados los objetivos, mantenlos hasta el final, no vayas variando, ni cambiando de opinión.

Insiste sobre los objetivos a medida que avanzas en las explicaciones
Está claro que los objetivos se fijan al inicio del curso, pero es aconsejable, que cada vez que se presente la oportunidad se recuerden.

Para enseñar y aprender hace falta voluntad
Si no estás dispuesto a repetir un sinfín de veces lo que explicas más vale que no te dediques a esto, pues hay que repetir y repetir hasta que se entienda.

Fuerza la participación con casos prácticos
Propón numerosos casos prácticos para que la gente se anime a participar.

Pon a disposición del alumno la experiencia del profesor
Recuerda tu experiencia y ayúdales con los pequeños trucos que empleaste para entender las distintas materias.

Lo que explicas sirve para algo más que para aprobar
Una forma de que vean que lo que explicas sirve para algo es que lo relaciones con ejemplos reales, no sólo teóricos.

Grupos heterogéneos
Intenta que los grupos que se forman para realizar prácticas, sean heterogéneos, para que todo el mundo aporte su punto de vista.

Predisposición a repetir y recordar lo anterior
Debes saber que no basta con decir las cosas una vez, hay que insistir e insistir.

Disfruta con lo que explicas
Si tus alumnos te ven disfrutar con lo que explicas, quedan contagiados y disfrutan aprendiendo.

Mantén la distancia
Compórtate en cada momento como lo que eres, eres su profesor, no su compañero.

Implícate en los trabajos
Cuando les pidas que te entreguen algo coméntales los resultados, que no les quede la idea de que lo que entregan no sirve para nada y de que el profesor ni lo mira.

Olvida que sabes informática, ponte en su lugar
Igual que cuando vas al médico pides que te den explicaciones de forma que las entiendas, no cometas el mismo error con tus alumnos, ponte a su nivel de conocimientos.

6.4. Grupo 2.2

Conócelos y sabrás a lo que te enfrentas
Si desde el primer día eres consciente de sus conocimientos, te será más fácil saber de qué pie cojean.

Programación temporal de los contenidos
Establece una programación temporal de los contenidos y dala a conceder a tus alumnos desde el inicio del curso.

Motivación laboral
Un modo de motivarles es relacionar lo que explicas con el mundo laboral con el que se van a encontrar, relacionarlo con la realidad.

Actualízate continuamente
Renueva los ejemplos, las prácticas, etc, cada año, pues además de incrementar su batería de ejercicios les muestras que tú también trabajas, que no utilizas lo mismo año tras año.

Haz las pausas necesarias
Detente de vez en cuando, déjales un tiempo para reflexionar, o para hacer ejercicios. Establece pausas.

Relaciona lo que impartes con el objetivo global de la asignatura
Cada sesión, cada tema no es algo independiente del resto del temario y existe una razón para

impartirlo en un momento determinado y no en otro. Haz que conozcan el porqué de cada bloque o unidad temática.

Relaciona la asignatura con el resto
Tu asignatura no es una asignatura aislada en el conjunto de la titulación, relaciónala con las otras asignaturas.

Sé un buen modelo para el alumno
Tus alumnos copian, esto es así. Pero lo bueno, es que también lo hacen de ti.

Controla la voz
No resultes monótono y aburrido, provoca cambios de tono.

Cuenta batallitas
De vez en cuando cuenta alguna experiencia relacionada con lo que imparte, les será más fácil recordarlo.

Pon ejemplos amenos
Ya sabes que con imaginación se pueden encontrar ejemplos que resulten más amenos que otros. Utilízalos.

Reconoce tus errores
Todos somos humanos y por tanto, todos nos podemos equivocar, no te importe reconocer tus errores, tus alumnos te lo agradecerán.

Conoce las necesidades del mercado laboral
Orienta tus explicaciones y ejercicios según va a avanzando y exigiendo el mercado laboral.

7. Conclusiones

Hemos presentando cinco "Decálogos para el profesor de informática novato". El primero de ellos ha sido elaborado por los profesores que firmamos este artículo, y es fruto de una dilatada reflexión sobre el proceso docente, que nos ha llevado a intentar condensar en alrededor de diez puntos nuestra visión del mismo.

Los otros cuatro han sido elaborados por grupos de alumnos del CAP. Tras leer el decálogo propuesto por cada uno de los grupos, observamos que con algunas diferencias la mayoría se centran en los mismos aspectos: motivación, respeto, dominio de la materia, … Sin embargo, resulta curioso que nadie se detenga a hablar sobre el proceso enseñanza/aprendizaje, es decir, en cómo hacer que sus estudiantes aprendan, sino más bien, en la imagen que el profesor debe darles.

No queremos decir que unos sean mejores que otros. Aquí quedan plasmados los cinco y que cada uno saque sus propias conclusiones.

Contrastan estos resultados con lo que esperábamos. Todos nuestros esfuerzos se centran en encontrar o aproximarnos a la "fórmula" que nos diga cómo enseñar mejor y ellos centran su preocupación en no mostrar debilidad. Esa actitud defensiva puede parecer justificada en un primer momento. Pero deben ser capaces de superarla. Ya que esa misma actitud defensiva hace que no se quieran asumir riesgos en clase y, evidentemente, los métodos pedagógicos innovadores plantean riesgos.

Una posible razón para nuestra decepción puede ser el ya comentado de qué no todos los alumnos del CAP se sienten realmente profesores. Sigue imperando la idea de que la docencia es una salida profesional fácil y cómoda. Todos desean las "vacaciones del maestro" y la "seguridad del funcionario". El aumento experimentado en los tres últimos años de los titulados de informática que acuden al CAP, avala esta impresión. Ante un mercado de trabajo no tan boyante, el recién titulado (y no tan reciente) ve en la docencia una alternativa profesional.

Como ye hemos comentado antes, un profesor de informática debe saber informática, eso no lo pone nadie en duda y es requisito para acceder al puesto de trabajo. Pero además debe saber enseñar, hacer que el alumno aprenda. Se habla de que el alumno debe "aprender a aprender", pero a los profesores ¿se nos ha *enseñado a enseñar*? En este trabajo hemos pretendido dar unos brochazos sobre la tarea de enseñar y nuestra experiencia al *enseñar a enseñar*, y en particular al *enseñar a enseñar informática*. Tras una reflexión general sobre la enseñanza, hemos intentado recoger las diferencias existentes dadas las características específicas de la materia con la que tratamos, la Informática.

Agradecimientos

Queremos agradecer a los alumnos de nuestros grupos del módulo de Didácticas Específicas: Informática del Curso de Aptitud Pedagógica de la Universidad de Alicante, en los cursos 2003/2004 y 2004/2005, por su colaboración, así como a los

profesores, compañeros del departamento, que han impartido dicho módulo del curso con nosotros, Patricia Compañ, Paco Mora, Pepe Balmaceda, David del Arco y Herminia Pastor.

También quisiéramos agradecer a los asistentes a la ponencia "Decálogo para el profesor novel" en las Jenui 2004 por sus intervenciones y en especial a Juan José Escribano por hacernos notar que aunque hablemos de decálogo no estamos obligados a limitarnos a diez.

Referencias

[1] Allen, D. *La evaluación del aprendizaje de los estudiantes. Una herramienta para el desarrollo profesional de los docentes.* Redes en Educación, 4, Paidós, 2000.

[2] Álvarez Rojo, V. y Lázaro Martínez, Á. *Calidad de las Universidades y Orientación Universitaria*. Colección orientación, Ediciones Aljibe, 2002.

[3] Ben-Ari, M. "Constructivism in Computer Science Education", *Journal of Computers in Mathematics and Science Teaching*, vol. 20, no. 1, pp. 45-73, 2001.

[4] Ben-Ari, M.. "Situated Learning in Computer Science Education", *Computer Science Education*, 2003.

[5] Brown, S. y Glasner, A. *Evaluar en la Universidad. Problemas y nuevos enfoques*. Universitaria, Narcea, 2003.

[6] Carnoy, M. *Las TIC en la enseñanza: posibilidades y retos*. Lección inaugural del curso académico 2004-2005. Universitat Oberta de Catalunya (UOC).

[7] Feynman, R. P. *El placer de descubrir*. Dacrontos, Editorial Crítica, 2000.

[8] Forbus, K. D. y Feltovich, P. J. *Smart Machines in Education. The Coming Revolution in Educational Technology.* The MIT Press / AAAI Press, 2001.

[9] Gal-Ezer, J. y Harel, D. "What (Else) Should CS Educators Know?", *Communications of the ACM*, vol. 41, no. 9, pp. 77-84, September 1998.

[10] Llorens Largo, F y Satorre Cuerda, R. *Decálogo para el profesor novel*. X Jornadas de Enseñanza Universitaria de la Informática. pág. 171-178. Editorial Thomson, 2004.

[11] Michavila, F. *La salida del laberinto. Crítica Urgente de la Universidad*. Editorial Complutense, 2001.

[12] Morin, E. *La mente bien ordenada. Repensar la reforma, Reformar el pensamiento*. Tercera edición, Editorial Seix Barral, 2001.

[13] OCDE. *Los Desafíos de las Tecnologías de la Información y las Comunicaciones en la Educación*. Organización para la Cooperación y el Desarrollo Económicos (OCDE) y Ministerio de Educación, Cultura y Deporte, 2002.

[14] Savater, F. *El valor de educar*. 3ª, Editorial Ariel, Barcelona, 1997.

[15] Web de la Asociación de Enseñantes Universitarios de la Informática, AENUI: http://www.aenui.org

[16] Web de la Asociación para el Desarrollo de la Informática Educativa, ADIE: http://chico.inf-cr.uclm.es/adie

[17] Web de la asignatura correspondiente al módulo de didáctica específica de informática del CAP de la Universidad de Alicante: http://www.dccia.ua.es/dccia/inf/asignaturas/MDEI

[18] Web de las X Jornadas de Enseñanza Universitaria de la Informática, JENUI 2004: http://www.dccia.ua.es/jenui2004

Obstáculos al aprendizaje cooperativo universitario: una mirada a los estudios de informática y a la Universitat Jaume I

V. Javier Traver* Joan A. Traver**

*Dep. de Llenguatges i Sistemes Informàtics

**Dep. d'Educació

Universitat Jaume I

{v|j}traver@uji.es

Resumen

El aprendizaje cooperativo (AC) es una forma de aprendizaje activo que presenta, según los expertos, importantes beneficios, tanto educativos como sociales. Sin embargo, el impacto que éste ha tenido en la universidad española es más bien escaso. Además, en el caso particular de los estudios de informática, la presencia de formas de aprendizaje cooperativo resulta poco más que anecdótica. En este artículo, además de mostrar la escasa incursión práctica del AC, elaboramos e indagamos en las posibles causas de este fenómeno.

1. Introducción

Recientemente apuntábamos [4] una serie de motivos que pueden explicar por qué, pese a su interés formativo casi indiscutible, el aprendizaje cooperativo (AC) está prácticamente ausente de las aulas de la universidad española. En el presente artículo resumimos estas posibles causas y aportamos nuevos indicios acerca de la medida en que el AC está presente (o ausente) en la universidad. Nuestro estudio no es, ni mucho menos, exhaustivo, pero creemos que proporciona una idea de la situación general en la universidad española y, más concretamente, en los estudios de informática.

Como veremos, no es lo mismo el aprendizaje cooperativo (AC) que el trabajo en grupo (TG). En este artículo hemos diferenciado las propuestas que se centran en uno u otro planteamiento. Pero, aunque es conveniente realizar esta distinción, en algunos análisis de datos hemos agrupado los dos tipos de propuestas en función de su cercanía de intención. Las propuestas de TG, normalmente, se convierten en la entrada natural al AC.

Este artículo se estructura de la siguiente manera. Tras esta introducción, resumimos las características y beneficios del AC, y señalamos los motivos que pueden estar dificultando su implantación (Sección 2). Con el fin de analizar algunos de estos motivos en situaciones concretas, estudiamos el caso de nuestra universidad (Sección 3), y el de las conocidas jornadas nacionales sobre enseñanza universitaria de informática (Sección 4). Concluimos el artículo con un resumen y una discusión (Sección 5).

2. Aprendizaje cooperativo

El AC obedece a un paradigma de enseñanza-aprendizaje centrado en el *proceso*, con una perspectiva *dialógica*, comunitaria o sociocultural de la educación [5]. No debe confundirse AC con TG (Tabla 1). Una característica fundamental del AC es la *interdependencia positiva*: los estudiantes pueden lograr sus objetivos *si y sólo si* los demás participantes consiguen también los suyos [3]. Para ello, es fundamental tanto la *unidad de meta*, como la *colaboración* de todos para conseguirla. Por otro lado, la *interacción* social entre iguales se considera un aspecto clave para el aprendizaje.

Tabla 1: Trabajo grupal (tradicional) frente a trabajo cooperativo (adaptado de [3])

Elemento	Trabajo en Grupo	Trabajo Cooperativo
Interés	Resultado del trabajo	Máximo rendimiento de todos
Responsabilidad	Sólo grupal	Individual
Grupos	Homogéneos	Heterogéneos
Liderazgo	Único y personal	Compartido
Ayuda	Libertad para decidir si ayudar y a quién	Responsabilidad de ayudar a los otros miembros
Meta	Completar la tarea	Máximo aprendizaje posible
Habilidades sociales	Se suponen	Se enseñan (y se aprenden)
Papel del profesor	Mero evaluador del resultado final	Interventor y supervisor del trabajo
Lugar de trabajo	Fuera del aula	En el aula

Entre los beneficios del AC, podemos mencionar un aumento de la motivación por aprender, una mayor retención del conocimiento, una comprensión más profunda, y una actitud más positiva hacia la asignatura en cuestión [2]. Pero, además de los beneficios académicos, el AC permite aumentar la autoestima, y desarrollar actitudes positivas tales como el respeto, la ayuda y la colaboración. Por estos motivos, la organización cooperativa de la clase se muestra superior a las organizaciones competitiva e individualista, aunque todas ellas pueden ser complementarias [3].

Las dificultades al AC señaladas en [4] las resumimos y adaptamos en las Tablas 2, 3 y 4.

Tabla 2: Posibles obstáculos al AC por parte del estudiantado

Falta de formación
Suspicacia frente al AC
Contrariedad por asistir a clase
Preferencia por la pasividad en el aula
Miedo a: – la interacción con sus compañeros – las situaciones sociales – que alguien no de la talla – la autonomía intelectual
Sensación de no aprender tanto
Sobrecarga de trabajo
Escepticismo sobre su utilidad

3. La situación en la UJI

3.1. La Universitat Jaume I (UJI)

Creada en 1.991, la UJI es una universidad relativamente joven. Las titulaciones y departamentos se organizan en tres centros: la Facultad de Ciencias Humanas y Sociales (FCHS), la Facultad de Ciencias Jurídicas y Económicas (FCJE) y la Escuela Superior de Tecnología y Ciencias Experimentales (ESTCE). La Tabla 5 muestra la distribución por centros de los 990 profesores de la UJI (a fecha de julio de 2.004).

3.2. Cursos de formación del PDI

Hemos apuntado [4] que la falta de recursos formativos puede explicar parte de la escasa presencia de AC. Para sustentar o rechazar esta hipótesis resulta conveniente recurrir al análisis de los cursos formativos organizados en el seno de la propia universidad.

La Tabla 6 ofrece el número de cursos formativos ofrecidos por la UJI a todo el PDI, por cursos académicos desde el 1.998–99. Hemos contabilizado como cursos de AC no sólo aquéllos que lo son de forma exclusiva sino también cursos del estilo «Metodología en do-

Tabla 3: Posibles obstáculos al AC por parte del profesorado

Formación pedagógica insuficiente
Desinterés por temas docentes
Inercia en la forma de enseñar
Dificultad de la aplicación del AC
Tiempo y esfuerzo extra necesarios
Compromiso docencia-investigación
Miedo a: – la inseguridad ante la novedad del AC – las reacciones de los alumnos – no saber arbitrar en los conflictos – las opiniones de otros profesores – perder el control de la clase – no avanzar lo suficiente el temario – perder el protagonismo en el aula
Falta (o pérdida) de ilusión o motivación
Mitos: – AC no tiene gran sentido en la Universidad – AC no es aplicable en todas las asignaturas
No ver en el AC un complemento, una alternativa, una necesidad, o una solución
Escepticismo sobre la utilidad y los beneficios del AC

Tabla 4: Posibles obstáculos al AC por parte de la sociedad y la Universidad

Sociedad competitiva, cultura contrapuesta a las ideas del AC
Infraestructura y recursos inadecuados o insuficientes
Fomento de la investigación y desprestigio de la docencia
¿PDI? «Personal Docente *e* Investigador» ó «Personal Docente *o* Investigador»
Despreocupación por la calidad de la enseñanza
Conocimiento insuficiente (investigaciones sobre AC relativamente recientes)

cencia universitaria», que suelen incluir en su contenido la docencia basada en AC. Sólo se

Tabla 5: PDI en los tres centros de la UJI (datos de julio de 2.004)

Centro	Núm. Profesores	%
FCHS	335	34
FCJE	247	25
ESTCE	408	41
Total	990	100

han contemplado como cursos formativos los que lo son como tal, y están abiertos a todos los docentes. Excluimos, por tanto, cursos de salud laboral, jornadas en los que se exponen proyectos de mejora educativa, seminarios específicos organizados por ciertos departamentos y dirigidos a sus docentes, etc. Los datos del actual curso 2.004–05 son provisionales, pues sólo incluyen los programados para el primer semestre.

Del total de las 109 acciones formativas, 12 (11 %) corresponden de forma directa al AC como opción metodológica u organizativa, de las cuales sólo una se centraba totalmente en el AC como propuesta de trabajo; las demás lo abordaban conjuntamente con otras opciones. Otros 15 cursos (14 %) corresponden a propuestas de TG.

Sumando los cursos tanto de AC como de TG, puede apreciarse un cierto incremento en términos absolutos. Ello puede indicar una mayor oferta formativa a los profesores. En porcentaje respecto al número total de cursos, el incremento apunta a un interés o sensibilidad especial por parte de la Universidad en dar este tipo de formación.

A la vista de estos datos, creemos que la falta de oferta formativa no es, en este caso, un motivo significativo de la baja tasa de aplicación del AC. Ahora bien, sería interesante analizar el perfil del profesorado que participa en estos cursos, sus motivaciones, sus necesidades, sus actitudes. Seguramente, este análisis nos aportaría claves importantes.

Curiosamente, hablamos de formación del PDI, pero no del propio estudiantado. No parece haber ninguna iniciativa en esta línea, aunque creemos que sería muy oportuno que

Tabla 6: Número de cursos de formación para el PDI sobre trabajo en grupo (TG) o aprendizaje cooperativo (AC) o ambos (G/C)

	Número de cursos			
Curso	TG	AC	Total	G/C (%)
98–99	2	1	11	27
99–00	3	1	18	22
00–01	2	1	18	17
01–02	1	3	17	24
02–03	5	2	24	29
03–04	2	3	16	31
04–05	0	1	5	20
Total	15	12	109	25

hubiese algún tipo de formación del estudiante, encaminada a instaurar una «cultura» de aprendizaje cooperativo, y que beneficiaría a los propios estudiantes y a los profesores.

3.3. Proyectos de mejora educativa

Tabla 7: Número de proyectos de mejora educativa sobre trabajo en grupo (TG) y aprendizaje cooperativo (AC). Números absolutos y relativos (porcentaje entre paréntesis)

Curso	TG (%)	AC (%)	Total
99–00	1 (8)	2 (17)	12
00–01	1 (2)	3 (7)	41
01–02	2 (3)	7 (9)	80
02–03	4 (1)	11 (11)	98
03–04	10 (8)	19 (15)	129
04–05	9 (6)	21 (14)	145
Total	27 (5)	63 (12)	505

La Fig. 1 muestra la evolución en el tiempo de los proyectos de mejora educativa por centros. Sin la menor duda, la FCHS destaca de forma evidente en la cantidad de proyectos de mejora realizados. Y el motivo no es la mayor cantidad de PDI (Tabla 5), sino más bien que es en la FCHS donde están los departamentos que tradicionalmente han mostrado mayor preocupación por temas docentes. Además, departamentos como el de educación aúnan los intereses docentes e investigadores, lo que les sitúa en una situación de clara «ventaja» respecto a los docentes del resto de departamentos. De hecho, podemos ver que la ESTCE, pese a contar con el mayor porcentaje de PDI, no representa un porcentaje paralelo en cuanto a proyectos de mejora. Respecto a los departamentos de informática, el número de proyectos ha oscilado entre 1 y 5 por año.

Sería interesante observar la distribución, también por centros, de los proyectos de mejora referidos a TG y AC. Desafortunadamente, los documentos de que disponemos no contienen estos datos, pues la relación de proyectos es global de toda la Universidad. Sin embargo, a nivel cualitativo sí hemos podido constatar que son los departamentos de la FCHS (en particular los de Educación y Psicología), y algunos de la FCJE (e.g., Derecho) los que más sensibilidad parecen mostrar hacia el trabajo en grupo y el aprendizaje cooperativo.

Tal vez el dato más significativo en el contexto de este artículo es que, en cuanto a la informática, no ha habido *ningún* proyecto referido a AC. Este dato es bastante ilustrativo, si bien tampoco debe llevarnos a pensar que todos los docentes en informática ignoran el tema del aprendizaje cooperativo o no lo aplican para nada. De hecho, uno de los autores del presente documento, docente en informática, trata de incluir AC en algunas de sus clases y, sin embargo, no tiene un proyecto de mejora. En este sentido, puede haber experiencias de AC que permanecen «ocultas», aunque intuimos que se trata de una minoría.

Con todo, pensamos que, de forma global, hay una importante cantidad de proyectos sobre TG/AC a nivel de la UJI. Este dato es algo prometedor, y nos anima al optimismo. Es muy posible que, a corto-medio plazo, este tipo de innovaciones en el terreno de la enseñanza-aprendizaje se haga extensivo a más y más profesores. Artículos como éste pueden, probablemente, aportar su grano de arena en la difusión del AC como una beneficiosa alternativa (o, al menos, un interesante complemen-

to) a estrategias docentes más clásicas como las exposiciones magistrales.

Si nos centramos en los proyectos de mejora, el incremento progresivo de los mismos resulta evidente (Tabla 7, Fig. 1), dato ya apuntado en [1]. Los motivos que podemos apuntar son varios:

- La existencia de los programas de mejora educativa se va conociendo cada vez más entre el profesorado. Aunque siempre hay profesores noveles, y el profesorado de la UJI es, como la propia universidad, relativamente joven, cada vez más profesores conocen estos proyectos, por compañeros, por la publicidad de la propia USE (*Unitat de Suport Educatiu*, Unidad de Apoyo Educativo), etc.

- Desde el curso 1.998–99 se vienen realizando en nuestra universidad tanto módulos formativos para los profesores noveles como seminarios y talleres abiertos a todo el PDI. Creemos que esta oferta formativa explica, por un lado, el incremento de proyectos de mejora educativa realizados y, por otro, proyectos sobre temas concretos. Por ejemplo, sospechamos que algunos proyectos de mejora se han solicitado a raiz de la participación de algún profesor en algún modulo formativo concreto.

- En los actuales procesos de acreditación para las nuevas figuras LOU parece que se está teniendo en cuenta que el candidato cuente con la participación de alguno de estos proyectos. Esta motivación extrínseca parece suficientemete importante [1].

- El proceso de convergencia hacia el espacio europeo de educación superior recoge el AC como una de las principales propuestas en las que basar los proyectos de innovación y mejora docente para avanzar hacia mayores niveles/cotas de optimización y calidad educativa en la enseñanza universitaria. No es extraño, por tanto, que una gran parte de los proyectos de mejora educativa analizados y que tienen que ver con el proceso de convergencia europeo, tomen el AC como uno de los aspectos fundamentales para articular sus propuestas de innovación educativa.

4. Aprendizaje cooperativo en las JENUI

Un indicio acerca del grado de presencia del aprendizaje cooperativo en los estudios universitarios de informática nos lo puede proporcionar las actas de las JENUI (Jornadas de Enseñanza Universitaria de la Informática).

Para analizar estas actas, hemos recurrido una base de datos en formato BibTeX de todas las ponencias (incluyendo conferencias invitadas, *demos*, recursos didácticos, etc.), y hemos buscado ciertas palabra clave en los títulos. Somos conscientes de las limitaciones que supone clasificar una ponencia por su título. Es posible, por ejemplo, que haya cierta aplicación de aprendizaje cooperativo tras una ponencia cuyo título no menciona esta palabra. Sin embargo, en general, la mención explícita de ciertas palabra clave en el título de un documento da a entender a qué se quiere hacer hincapié sobre el contenido de ese documento. En este sentido, buscar por palabra clave, pese a su sencillez, resulta efectivo.

Hemos distinguido dos grupos de palabras clave: por un lado, «grupo» y «equipo», y, por otro, «cooperativo» y «colaborativo». De este modo pretendemos separar el trabajo en grupo (que no necesariamente es cooperativo), del aprendizaje cooperativo en sí. Resulta evidente que los resultados obtenidos de este modo no son fiables al cien por cien, pero creemos que sí son muy orientativos.

En la Tabla 8 presentamos, para cada año, el número de ponencias de cada una de estas dos clasificaciones («TG» y «AC»), así como el número total de ponencias en el JENUI. El JENUI se viene realizando desde 1.994, pero los dos primeros años no hubo libro de actas, tan sólo unas 6 conferencias (ninguna de ellas sobre AC), y en 1.996 no se celebró ningún congreso. Así, los datos más representativos son a partir de 1.997.

A la vista de la tabla, resulta patente la es-

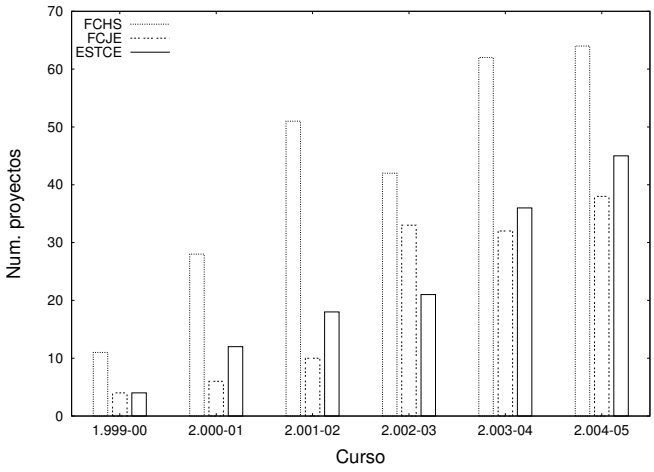


Figura 1: Evolución del número de proyectos de mejora educativa por centros

Tabla 8: Número de ponencias sobre TG (trabajo en grupo) y AC (aprendizaje cooperativo) en las JENUI

	Número de Ponencias		
Año	TG	AC	Total
1.994	0	0	6
1.995	0	0	6
1.996	-	-	-
1.997	1	1	67
1.998	2	0	79
1.999	1	0	64
2.000	1	1	90
2.001	1	0	99
2.002	3	2	80
2.003	1	1	80
2.004	0	3	64
Total	12	6	635

casa presencia, tanto en términos absolutos como en términos relativos (en relación al total de ponencias) de trabajos sobre AC. Aunque ligeramente mayor, el número de artículos sobre TG es también reducido. No resulta evidente ningúna tendencia al alza o a la baja, si bien es posible entrever un tímido incremento de AC, de forma muy débil y poco constante. La mayor incidencia del TG se debe, posiblemente, a que se trata de una muy conocida forma de hacer que trabajen los alumnos.

Otra reflexión interesante es la siguiente. En las JENUI tienden a participar aquellos profesores que sienten cierta inquietud en mejorar la calidad de su docencia y de los resultados de aprendizaje (hecho que también se da en los proyectos de mejora educativa). En este sentido, si el impacto del AC entre este colectivo es tan reducido, no es difícil imaginar la situación general en la enseñanza de informática en la universidad española. Por supuesto, se trata de una opinión subjetiva, y es posible que hayan experiencias de AC que no se hagan públicas. Pero la evidencia de estos datos nos inclina a tener la impresión de que, en docencia de informática se sabe poco o se desconoce casi todo sobre el AC. O, aún conociéndolo, no se sabe o no se quiere aplicar.

En la Tabla 9 se muestran, a título ilustrativo, los títulos de estas ponencias, lo que proporciona mayor información sobre la naturaleza de estos trabajos. Por ejemplo, el trabajo en grupo se da en mayor medida en asignaturas de *ingeniería del software*, o de programación y estructuras de datos. Puede ocurrir que no todas las asignaturas se presten por igual al trabajo en grupo o cooperativo. En algunos de estos títulos parece derivarse la no cooperación entre los miembros de los grupos. Por ejemplo, la expresión «reparto de la carga de trabajo» hace más alusión a la concepcón clásica de TG. En otros casos, no es evidente si puede haber AC o no, como en el caso de «prácticas en grupos virtuales». En general, creemos que la presencia de nuevas tecnologías («foros virtuales», «sistema hipermedia», «docencia virtual») ha de tomarse con especial precaución, pues, tratándose de estos educadores de personas con conocida inclinación tecnológica, los aspectos técnicos pueden cobrar mayor relevancia que su trascendencia docente. En cualquier caso, muchas de estas afirmaciones son, en buena medida, atrevidas especulaciones, y hay que conocer de cerca cada trabajo para evaluar de modo objetivo si en él se aplica AC o no.

5. Conclusiones

Para observar la repercusión práctica del AC, hemos estudiado el caso concreto de la Universitat Jaume I en cuanto a oferta formativa y a proyectos de mejora educativa. También hemos analizado las actas de las JENUI como fuente de los avances de innovación pedagógica en docencia informática. Tomando estos datos como significativos, y atreviéndonos a extrapolar al ámbito nacional, los resultados apuntan a que el AC:

- Parece estar cobrando alguna importancia, de forma general y sin una tendencia clara, en las aulas de nuestra universidad.
- Tiene un impacto muy poco significativo en la docencia de la informática.

No obstante, pensamos que la tendencia a medio-largo plazo será que el AC también se adentre en los departamentos y titulaciones de informática, aunque tal vez con cierto retraso respecto a otras áreas de conocimiento.

Parece que a los docentes en informática no les preocupe demasiado la mejora de la calidad de la docencia, al menos no la asociada a las propuestas de AC. Este hecho resulta llamativo, y hasta cierto punto preocupante, dada la importancia que el aprendizaje de habilidades colaborativas tiene para el futuro desarrollo profesional en cualquier perfil profesional. Es intrigante si, además de los motivos generales apuntados aquí (y que, ciertamente, no son pocos ni banales), existen otros propios de la informática o de sus docentes.

Agradecimientos

A Miguel Ángel Fortea, por su ayuda en el análisis de los proyectos de mejora educativa. A Joe Miró, por sus ficheros de las ponencias de todas las JENUI.

Referencias

[1] E. Alcón Soler et al. Consolidación del programa de ayudas para la mejora e innovación educativa de la Universitat Jaume I. En *Actas de la II Jornada de Mejora Educativa de la UJI*. Centro de Publicaciones de la Universitat Jaume I, 2003.

[2] Richard M. Felder y Rebecca Brent. Navigating the bumpy road to student-centered instruction. *College Teaching*, 44:43–47, 1996.

[3] Rafaela García, Joan Andrés Traver, y Isabel Candela. *Aprendizaje cooperativo: fundamentos, características y técnicas*. Editorial CCS, Madrid, 2001.

[4] V. Javier Traver y Joan A. Traver. ¿Por qué no enseñamos a aprender cooperativamente? En *Actas de las X Jornadas de Enseñanza Universitaria de Informática, JENUI 2004*, páginas 297–304. Alicante, julio 2004.

[5] G. Wells. *Indagación dialógica. Hacia una teoría y una práctica socioculturales de la educación*. Paidós, Barcelona, 2001.

Tabla 9: Títulos de ponencias sobre trabajo en grupo (TG) y aprendizaje cooperativo (AC) en las JENUI

Año	Ponencias sobre TG	Ponencias sobre AC
1.997	«Metodología docente orientada a grupos de trabajo»	«Colaboración de alumnos de diferentes cursos»
1.998	«La Programación en grupos de trabajo» «Una experiencia docente de diseño y desarrollo de proyectos de software en grupo utilizando las normas ISO 9000-3»	
1.999	«Prácticas en grupos virtuales en "Estructura de la Información"»	
2.000	«Clases dirigidas a grupos especiales de alumnos para la asignatura "Algoritmos y Estucturas de Datos 2"»	«CALIOPE: Una arquitectura para el aprendizaje autónomo colaborativo»
2.001	«Uso de técnicas de dinámica de grupos para sensibilizar a los alumnos de Ingeniería del Software en los problemas de comunicación»	
2.002	«Propuesta metodológica para la mejora de la calidad y la excelencia de la Educación Superior en Informática mediante el fomento del trabajo en equipo» «La docencia virtual como herramienta de apoyo en una metodología orientada a grupos de trabajo. Aplicación a la asignatura Nuevas Tecnologías de la Programación» «Reparto de la carga de trabajo en la realización de prácticas en grupo mediante una herramienta de estimación»	«Un modelo para aplicación sistemática de aprendizaje cooperativo» «Integración del aprendizaje individual y del colaborativo en un sistema hipermedia adaptativo»
2.003	«Desarrollo de actividades en grupos coordinados sobre el modelado y simulación del proceso de transmisión de datos»	«Una componente *e-learning* de aprendizaje colaborativo para el proyecto IDEFIX»
2.004		«Aprendizaje cooperativo: implantación de esta técnica en dos asignaturas reformadas y evaluación de resultados» «¿Por qué no enseñamos a aprender cooperativamente?» «Foros virtuales colaborativos en línea aplicados a procesos de tutorización»

Cambio de modelos basados en la enseñanza a modelos basados en el aprendizaje. Una experiencia práctica

Carlos Catalán[1], Raquel Lacuesta[1], Alejandro Hernández[2]
[1]Dpto. de Informática e Ingeniería de Sistemas
[2]Dpto. de Economía y Dirección de Empresas
Escuela Universitaria Politécnica de Teruel
Universidad de Zaragoza
44003 Teruel
e-mail: {ccatalan,lacuesta,alex}@unizar.es

Resumen

Un aspecto importante en la adaptación al espacio europeo de educación superior es el cambio de modelos basados en la enseñanza a modelos basados en el aprendizaje. Este trabajo presenta una experiencia sustentada en el método de aprendizaje basado en problemas, con un enfoque interdisciplinar y en la cual participan tres asignaturas.

1. Introducción

La vertiginosa rapidez con la que en los últimos años se producen los avances tecnológicos, especialmente en el sector de las tecnologías de la información y comunicaciones, hace cada vez más imprescindible que los profesionales actualicen de forma constante sus conocimientos. En esta línea los procesos de convergencia universitaria europea [8] [5] indican como un factor importante el que los titulados adquieran la capacidad de aprendizaje continuo (*lifelong learning*), o aprender a aprender. Para lograr tal fin está tomando importancia la necesidad de aplicar métodos docentes más centrados en el estudiante (aprendizaje) que en el profesor (enseñanza) [6] [14] [4]. Uno de estos métodos es el *aprendizaje basado en problemas* o *problem-based learning* (PBL).

1.1. Método PBL

Este método persigue que los alumnos tengan un rol más activo en su aprendizaje. En el modelo tradicional de enseñanza, el profesor expone primero la información y posteriormente busca su aplicación en la resolución del problema. Por el contrario, en PBL se presenta el problema, se identifican las necesidades de aprendizaje, se busca la información necesaria y finalmente se regresa al problema [7]. En este método tienen importancia tanto la adquisición de conocimientos como el desarrollo de habilidades y actitudes. Actualmente se considera que PBL puede ser uno de los métodos adecuados para los nuevos modelos de educación superior basados en el aprendizaje [20] [10] [7] [3]. Las primeras experiencias de este método se han dado en las ciencias biomédicas [2]; posteriormente otras disciplinas, incluyendo la informática [9] [15] [18], han utilizado también PBL.

Según [2] puede definirse PBL como el aprendizaje que resulta del proceso de trabajar hacia el conocimiento de la resolución del problema. Las reglas de oro de PBL son [11]:

1. Los alumnos han de asumir la responsabilidad de su propio aprendizaje.
2. Los problemas planteados han de ser intencionadamente poco estructurados y deben permitir interpretaciones libres.
3. El aprendizaje no se ha de dirigir hacia una súper especialización de los conocimientos, sino hacia un abanico de disciplinas o temas.
4. Lo que los alumnos aprenden en las fases de estudio y de aprendizaje autónomo, ha de aplicarse posteriormente al problema práctico propuesto.
5. Es esencial efectuar una síntesis final de todo aquello que se ha aprendido durante el proceso de resolución del problema. Es necesario discutir qué conceptos o principios se han asumido bien, y cuáles sería necesario reforzar, antes de iniciar el proceso de evaluación.
6. La evaluación y auto-evaluación ha de llevarse a término al finalizar cada problema y en el momento de acabar la unidad curricular completa.

7. La evaluación individualizada de los alumnos se realizará siempre midiéndola en función de los objetivos previamente propuestos.
8. Los temas y las actividades han de estar en todo momento conectadas con el mundo real, y aportar valores apreciados en los ámbitos sociales y profesionales.
9. El trabajo en equipo cooperativo, la colaboración para aprender y la autonomía responsables, han de ser tomadas como competencias clave esenciales en el trabajo.
10. El PBL ha de constituir la base pedagógica del currículo y no sólo una parte de la didáctica curricular.

A la vista de estos principios observamos que en PBL los alumnos deben asumir una mayor libertad de acción y responsabilidad. Igualmente la figura del profesor contrae un nuevo papel: encaminar al alumno en el proceso de aprendizaje. Debe ser un tutor que realice un papel *activador* más que *facilitador* [13] [7] [3] [4]. El éxito o el fracaso de PBL dependen en gran medida de la preparación y entrenamiento del profesor-tutor.

Creemos que PBL debería permitir desarrollar las cualidades profesionales que se demandan en el mundo actual: aprendizaje continuo, autonomía, trabajo en grupo, espíritu crítico, capacidad de comunicación y planificación [6] [16] [17] [10].

2. Descripción de la experiencia

El objetivo principal ha sido realizar una experiencia de trabajo interdisciplinar basada en PBL. Existen diversas fuentes que abordan los aspectos prácticos de aplicación de este método docente. En nuestro caso hemos utilizado principalmente [13] [12] [7].

En la experiencia realizada durante los cursos 2002/2003 y 2003/2004 han participado tres profesores con sus respectivas asignaturas, todas ellas de la titulación de Ingeniería Técnica en Informática de Gestión: Bases de Datos II, Comercio Electrónico e Interfaces de Usuario. Estas asignaturas interactúan y se complementan de forma natural en el desempeño profesional de estos titulados. Las tres forman parte del segundo cuatrimestre del tercer curso de la titulación. Se decidió establecer grupos de trabajo de tres alumnos, siendo la formación de los grupos libre.

Una cuestión inicial fue explicar el nuevo método docente a los alumnos, de tal forma que éstos, acostumbrados a una enseñanza tradicional, pudiesen entenderlo y aceptarlo. Para ello, se intentó que los alumnos conocieran los objetivos perseguidos y los beneficios que se esperaba obtener.

Cabe señalar que todos los alumnos no estaban matriculados en todas las asignaturas. Esto atañe a otro de los puntos iniciales a resolver: cómo integrar la experiencia dentro de una organización de plan de estudios clásica, esto es, tres asignaturas que deben ser evaluadas con calificaciones separadas e individuales. Además, en la Universidad de Zaragoza existen tres convocatorias anuales de las cuales el alumno puede elegir dos. La solución adoptada consistió en definir adecuadamente en un proceso de negociación con cada grupo las diferentes partes del proyecto, asociándolas a las asignaturas, y las diferentes responsabilidades, asociándolas a las personas.

Se tuvo en cuenta que el proyecto propuesto cumpliera las características que debe tener un problema apropiado en PBL [13], a saber: debe ser relevante y de interés para los alumnos, tener objetivos y etapas claras, y ser complejo en el sentido de tener distintas soluciones y naturaleza interdisciplinar. Muchas situaciones reales del mundo profesional pueden reunir esas características. Nos planteamos aquí el caso de un supuesto cliente que encarga la realización de un proyecto con un fin determinado.

El primer año se decidió proponer varios proyectos posibles, debiendo elegir los grupos de trabajo uno de ellos. El hecho de estar todos relacionados con la programación Web los hacía atractivos para los alumnos, en concreto se propusieron: una tienda o comercio virtual, una revista o periódico virtual y un servicio de subasta electrónica.

Los requisitos debían ser definidos por el grupo: productos a vender, subastar o temas de la revista, tipos de usuarios/clientes y funcionalidad, aunque debían ser validados por el profesor representando éste el papel de cliente. Una vez definidos debían mantenerse a lo largo de todo el proyecto. Tenían que elegir también las herramientas y tecnologías necesarias. Esta elección debía estar fundamentada en las características que las hicieran óptimas para el desarrollo del proyecto. Así pues, los alumnos debían encontrar la solución a un problema planteado con distintas soluciones posibles,

teniendo que asumir su propio aprendizaje en los aspectos o materias necesarias para ello; este último punto es la esencia del método PBL [11]. Los conocimientos a adquirir estaban relacionados con seguridad, protección de datos y normativas relacionadas; gestión de usuarios, productos o noticias; usabilidad y accesibilidad. Es decir, conocimientos objeto de interés de las asignaturas participantes: Comercio Electrónico, Bases de Datos II e Interfaces de Usuario. Se sugirió a los grupos que adoptaran un nombre de grupo y asignaran también un nombre al proyecto. Un dato relevante es que los aspectos básicos de gestión y planificación de proyectos son desarrollados en la asignatura Ingeniería de Software II cursada el cuatrimestre anterior.

Durante el cuatrimestre se planteó un plan de trabajo con varios hitos a cumplir, en cada uno debía entregarse una documentación que reflejara las acciones realizadas y los resultados obtenidos. Se pidió a los grupos que adoptaran un formato común, en todos los documentos, que contuviera al menos: autor, revisor, fecha, proyecto, asunto, versión. Muchos adoptaron el formato empleado en Ingeniería de Software II.

La documentación entregada fue revisada por los profesores notificando a los alumnos una valoración positiva o negativa sobre la marcha del proyecto, la valoración no acumulaba para la nota final. El único objetivo era aportar una evaluación formativa [19] que ayudara al desarrollo del proyecto. El plan de trabajo propuesto a los alumnos fue el siguiente:

1. Entrega de la documentación referente al proyecto elegido y las responsabilidades de cada uno de los integrantes del grupo.
2. Entrega de documentación y presentación pública de cada grupo de la especificación de requerimientos, la planificación del proyecto y las herramientas y tecnologías elegidas para el desarrollo con su correspondiente justificación.
3. Entrega de documentación con el análisis, diseño preliminar y un prototipo.
4. Entrega de la documentación final y presentación pública del proyecto.

Para cumplir este plan de trabajo los alumnos contaban con cuatro horas semanales por asignatura. Al principio del cuatrimestre los profesores emplearon algunas horas para aportar contenidos fundamentales, básicamente introducir la materia en cuestión y suministrar algunas fuentes de información. Pasado ese periodo los alumnos debían emplear su tiempo en: trabajo personal, reuniones grupales con los profesores, reuniones de grupo, entrevistas personales con los profesores y presentaciones públicas.

Respecto a la evaluación, cada alumno obtendría una nota individual por asignatura. Para llegar a ella se asignaba una calificación global al proyecto basada en el resultado final; modulada por la documentación final entregada, las presentaciones públicas y la adecuación a la planificación presentada. Esta nota podía variar posteriormente por alumno y asignatura al analizar los profesores el trabajo realizado por cada alumno en las materias donde su asignatura era competente. Las funciones representadas por los profesores fueron las siguientes:

a. Proponer el problema a resolver.
b. Aportar inicialmente conocimientos básicos de las materias.
c. Representar el papel del cliente que encarga el proyecto.
d. Valorar críticamente la documentación entregada y las presentaciones públicas.
e. Realizar entrevistas semanales, con los siguientes objetivos: observar el correcto desarrollo de las diferentes fases de cada proyecto; comprobar el reparto de tareas dentro de los grupos; facilitar el aprendizaje de los alumnos a través del planteamiento de preguntas o alternativas, intentando que éstos encontraran por sí mismos posibles soluciones.

Para realizar estas funciones los profesores contaban con las cuatro horas semanales de docencia reglada y parte de las seis horas semanales de tutorías a que obliga nuestra Universidad; en este caso decimos parte porque los profesores imparten cada uno otra asignatura además de la que participa en la experiencia.

El segundo curso la forma de trabajo fue muy similar; los proyectos planteados fueron: aplicación de fichaje a través del DNI digital, aplicación segura de intercambio de ficheros para dispositivos inalámbricos a través de la tecnología Wifi, aplicación segura para la comunicación sobre dispositivos inalámbricos a través de la tecnología Bluetooth y un gestor de proyectos.

Durante este curso 2004/2005 se han producido algunos cambios significativos. El más

importante es la sustitución de la asignatura de Bases de Datos II por la de Estrategias y Sistemas de Información del Departamento de Economía y Dirección de Empresas. Esta asignatura es de 6 créditos y se imparte en el mismo cuatrimestre y curso que las otras dos. Otra modificación significativa es el aumento del tamaño de los grupos pasando de tres a cinco miembros, hay un total de 34 alumnos organizados en siete grupos. La composición de los grupos la han realizado los profesores, intentando su equilibrio en base a la matriculación por asignatura. La decisión motivó algunas críticas por parte de los alumnos. Por otro lado, se ha comenzado a utilizar, tanto por parte de los profesores como del alumnado, la herramienta para docencia virtual disponible en nuestra Universidad, WebCT. Además de poder dejar material para los alumnos ésta permite que los grupos de trabajo dispongan de repositorios para los documentos que generan, y medios de comunicación para los miembros de cada grupo como chat, foros y correo.

Se han aumentado el número de exposiciones públicas y se emplea un día a la semana fijo para que todos los grupos realicen sus exposiciones, así como las reuniones grupales. Los tres profesores acuden a dichas reuniones y actúan como observadores y/o clientes. Finalmente este curso hay un proyecto distinto para cada grupo y ha sido asignado por los profesores, los proyectos son: portal de turismo rural, portal empresarial, compañía de seguros on-line, compañía de servicios de software, periódico virtual, software de diseño de tiendas on-line y portal del motor.

2.1. Resultados de la experiencia

Para poder valorar la experiencia se realizó a los alumnos a una encuesta. Así, los alumnos consideraban que las principales capacidades desarrolladas fueron el autoaprendizaje (84,1%), el trabajo en grupo (42,0 %) y la iniciativa (31,5 %). Un número significativo creía que no se disponía al principio de los conocimientos necesarios (63,1 %), siendo adquiridos durante el curso (42 %). Respecto del funcionamiento de los grupos se consideraba que las opiniones eran escuchadas (63,1 %) y todos los miembros participaban de forma constructiva (57,8 %). Por el contrario, una amplia mayoría (78,9 %) requeriría una mayor ayuda por parte de los profesores. La falta de tiempo también fue indicada de forma clara (78,0 %). Finalmente un porcentaje amplio consideraba que el proyecto le había permitido integrar los conocimientos adquiridos durante la carrera (63,1 %), así como fomentado la discusión y el entendimiento del problema entre los miembros del grupo (42,0 %).

De los resultados se observa que PBL implica un esfuerzo mayor por parte de los alumnos. Esto puede ser debido a la falta de costumbre de los alumnos y a están habituados a trabajos más estructurados, con menos libertad de acción y responsabilidad. El cambio en la forma de tutorización fue entendido por algunos de los alumnos como un desentendimiento por parte de los profesores. Esto se comprueba en la demanda de una mayor ayuda. La forma de tutoría frustraba al alumno, el profesor únicamente pretendía aportar orientaciones, referencias o bibliografía, buscando que el propio alumno obtuviera las respuestas. Esta frustración es indicada también en otras experiencias similares [18]. Creemos que ese sentimiento debería disminuir con una mayor práctica de los profesores, y con un mayor uso de PBL a lo largo de los estudios.

No había un tutor concreto por grupo, los tres profesores se encargaban de tutorizar a todos los grupos. Como es lógico, es necesario que haya una buena coordinación entre ellos, ya que los alumnos en ocasiones tendían a preguntar repetidamente a todos los profesores sobre los mismos temas, buscando aquél que les facilitara más su labor. Las dudas referentes a los aspectos más técnicos de las materias eran resueltas en su caso por el profesor de la asignatura relacionada.

También se comprueba en algunos casos que los conocimientos aportados en anteriores asignaturas no estaban suficientemente asentados, o existía una falta de madurez para utilizarlos y avanzar en determinadas cuestiones. Creemos que los planes de estudio actuales pueden fomentar estas limitaciones, ya que en ocasiones los contenidos de una asignatura no se aplican en otras, ofreciendo una visión de las asignaturas como compartimentos cerrados.

Se apreció que los alumnos se angustiaban ante la idea de no poder realizar el proyecto. Creemos que esto se debió en parte a la necesidad de plasmar muchos de los conocimientos estudiados y sobre todo desarrollar nuevos por sí mismos. La mayoría de los alumnos vieron el proyecto como el primer trabajo de cierta magnitud que realizaban, lo que provocaba una

natural inseguridad, pudiendo estar además en juego hasta tres asignaturas (un total de 18 créditos).

Otro de los problemas mencionados fue la falta de tiempo. En muchos de los grupos se retrasó especialmente al principio la realización del proyecto, bien por no estimar correctamente el tamaño del proyecto y realizar una mala programación temporal, bien por no cumplir la programación hecha. A pesar de eso los proyectos realizados consiguieron en general las expectativas planteadas (una media del 85 % de los alumnos superaron sus asignaturas el primer curso, y 88 % el segundo), aunque algunos de ellos no lo hicieran en la primera convocatoria posible. El hecho de que las asignaturas sean cuatrimestrales hace que haya poco tiempo de reacción si el grupo no lleva un ritmo de trabajo adecuado. En muchos grupos la finalización del proyecto mejoró la autoestima, viéndose recompensado el esfuerzo realizado.

Respecto del trabajo en grupo los resultados son positivos; en la mayoría de ellos el ambiente de trabajo ha sido bueno y eso queda reflejado en las respuestas a las encuestas.

3. Algunas consideraciones sobre la aplicación de PBL en proyectos interdisciplinares

A la vista de la experiencia durante dos cursos y medio nos parece adecuado aportar algunas consideraciones.

En primer lugar creemos que el hecho de participar varias asignaturas en el mismo trabajo hace más fácil la aplicación de PBL, ya que en una asignatura cuatrimestral puede que no haya el suficiente tiempo de abordar problemas complejos con la profundidad necesaria [1].

Uno de los aspectos más interesantes para los alumnos ha sido la integración de conocimientos. Creemos que se ha contribuido a dejar de ver las asignaturas y las materias que se imparten como mundos estancos, situación muy alejada de la realidad profesional, donde las cuestiones o situaciones planteadas son habitualmente de naturaleza interdisciplinar. En este sentido la experiencia realizada se convierte en un paso previo al proyecto final de carrera que en ocasiones se deja como la única experiencia de integración de materias.

PBL facilita a los alumnos ejercitar en muchos casos por primera vez el autoaprendizaje, el trabajo en grupo, la gestión del tiempo, la comunicación oral y escrita. Además, creemos que obliga, en mayor medida que la enseñanza tradicional, a repartir el trabajo durante todo el cuatrimestre, liberándose de las habituales apreturas finales. Los alumnos también valoran la posibilidad de superar con un único trabajo varias asignaturas a la vez, y el aprovechar más su tiempo al disminuir la dispersión que normalmente produce cursar las asignaturas por separado. Además pensamos que, con un adecuado reparto de responsabilidades, los alumnos no tienen necesariamente que estar matriculados en todas las asignaturas. Una dificultad para éstos es la natural resistencia a todo cambio, así como la necesidad de trabajar desde el principio del cuatrimestre de manera constante.

Respecto de los profesores, estimamos que es un buen paso en la adaptación de los métodos docentes a los nuevos tiempos. Igualmente es positivo el trabajo en grupo en tareas de docencia, muy importante si además una de las cosas que queremos enseñar a los alumnos es la capacidad de trabajo en grupo. Los buenos resultados académicos obtenidos también animan a tener una valoración positiva.

La carga de trabajo inicial es superior, especialmente si no se tiene experiencia en el método PBL, como ocurre en este caso. Una vez en marcha la asignatura la carga no es excesiva; pensemos que se emplean las horas de clase regladas, fijadas por el centro, y parte de las horas de tutorías que normalmente en la enseñanza tradicional los alumnos infrautilizan.

Un factor muy importante es la coordinación de los profesores, especialmente si, como es nuestro caso, no hay un profesor-tutor por grupo. Los alumnos no deben encontrar incoherencias en las respuestas, orientaciones o exigencias. Para lograrlo es necesario realizar reuniones semanales para repasar uno a uno el desarrollo de los proyectos y el desempeño de los grupos.

Durante los dos primeros cursos las sesiones han tenido una duración de dos horas, creemos que es más adecuado mantener sesiones más largas con el objeto de que los alumnos estén inmersos en el problema un tiempo mayor. Así se está haciendo durante este curso, con sesiones de hasta cuatro horas. Creemos que ello conlleva un

mayor acercamiento al mundo profesional, donde habitualmente no se salta a cada hora de una materia a otra. En este sentido, nuestras asignaturas actualmente tienen sesiones de prácticas de dos horas semanales por grupo de asignatura. Para este curso se solicitó a la dirección del centro la adecuación de los horarios para poder realizar sesiones más largas de lo habitual. Evidentemente no se pretende que los alumnos realicen una única tarea durante todo el tiempo, sino que se alternen reuniones grupales, trabajo individual, entrevistas con los profesores y exposiciones públicas. Previamente a cada sesión los profesores deben programar un horario de actividades.

Otro aspecto importante en este sentido son los espacios utilizados [13]. Para este tipo de experiencias pensamos que son muy útiles las aulas con mobiliario flexible, que nos permitan, por ejemplo realizar reuniones grupales; es importante indicar que estimamos incorrecto que los alumnos celebren todas ellas en sus domicilios particulares. Lógicamente, al ser necesario utilizar equipos informáticos en parte de las tareas, será imprescindible que, o bien las clases tengan ordenadores portátiles, o bien el centro asigne simultáneamente, para dichos proyectos, salas con ordenadores y aulas lectivas, lo que puede ser una dificultad si el centro no dispone de recursos suficientes. En nuestro caso el centro nos facilitó un aula con mesas móviles y a la vez de un laboratorio con ordenadores, los que nos permitía trabajar en una u otra dependiendo de la característica de la actividad a realizar. Para un futuro el centro está pensando habilitar una nueva aula de este tipo y dotar a las dos con equipos informáticos portátiles.

Es importante medir adecuadamente la carga de trabajo que llevan consigo los proyectos para no afectar a otras asignaturas y para comprobar que la carga de trabajo se reparte equitativamente entre las materias de las tres asignaturas implicadas. Puede pasar que los alumnos tiendan a centrarse tanto en éstos que olviden otras responsabilidades. A modo de ejemplo, en muchos casos, en la evaluación de requisitos realizada hubo que frenar las excesivas pretensiones iniciales que algunos de los grupos habían planteado.

También creemos adecuado que los alumnos sean conscientes de la importancia de las reuniones, por ello se les debe pedir convocatorias de cada reunión con un orden de día, así como actas de los asuntos tratados. Uno de los alumnos realizará el papel de secretario. Las convocatorias y las actas deben formar parte de la documentación entregada. Entendemos que los profesores pueden estar presentes en las reuniones grupales para observar como son llevadas a cabo, aunque ello pueda afectar a su desarrollo, por el hecho de sentirse observado.

Finalmente, respecto del tamaño de los grupos, los dos primeros cursos han estado compuestos por tres alumnos, pero consideramos que grupos más grandes permiten una mejor distribución del trabajo y están más en línea con otras experiencias de PBL. Además del secretario se asignó un papel de coordinador [20] a uno de los miembros del grupo con la responsabilidad de moderar las reuniones, integrar resultados, coordinar áreas y controlar la marcha del proyecto. Esto debería mejorar el cumplimiento del plan de trabajo. Este curso el papel de secretario y el de coordinador son rotatorios, el orden es elegido por los alumnos contando para la calificación final. También, como se ha comentado previamente, se ha comenzado a utilizar el WebCT, herramienta que permite a los alumnos tener un lugar donde almacenar toda la información generada, así como disponer de medios de comunicación para el grupo. Pensamos que es acertada la decisión de que los profesores decidan la distribución de los grupos, habitualmente en otras asignaturas éstos los forman dos o tres personas unidas por afinidad. Como se indicó a los alumnos, romper esa rutina y aprender a trabajar con personas fuera del círculo habitual es bueno, ya que normalmente uno puede elegir los amigos pero no los compañeros de trabajo. En cualquier caso hay un ambiente normal de trabajo en los grupos, no habiéndose detectado hasta el momento ninguna situación problemática.

4. Conclusiones

Se está desarrollando una experiencia interdisciplinar de PBL en el centro. Ésta ha partido del interés de varios profesores en utilizar métodos de aprendizaje que permitan abordar los nuevos retos de la formación universitaria.

Las dificultades surgidas, sobre todo inicialmente, han sido debidas principalmente a la falta de hábito en alumnos y profesores; la

organización de nuestros planes de estudios actuales, con unas asignaturas compartimentadas y en muchos casos de duración cuatrimestral; y al sistema de convocatorias para la evaluación de nuestros alumnos.

Creemos que este momento, de inminente proceso de renovación completa de titulaciones y planes de estudio, es una buena oportunidad para realizar diseños curriculares que permitan emplear este tipo de métodos. En España ya existen experiencias de planes de estudio adaptados a PBL [1]. A pesar de las dificultades el resultado final está siendo satisfactorio.

Este trabajo ha sido incluido dentro de las Convocatorias de Proyectos de Innovación Docente de la Universidad de Zaragoza de los años 2003 y 2004.

Referencias

[1] Alcober, J., Ruiz, S., Valero, M., *Evaluación de la implantación del aprendizaje basado en proyectos en la EPSC (2001-2003)*. XI Congreso Universitario de Innovación Educativa en las Enseñanzas Técnicas, Vilanova i la Geltrú, julio 2003

[2] Barrows, H.S., Tamblyn, R.M., *Problem-Based Learning: An Approach to Medical Education*. New York, Springer Publishing Company, 1980

[3] Buck Institute for Education. *A overview of Project Based Learning*. 2002 http://www.bie.org/pbl/

[4] Carmiña, C., Ballester E., Coll C., García E., *Mitos y realidades de la innovación educativa*. XI Congreso Universitario de Innovación Educativa en las Enseñanzas Técnicas, Vilanova i la Geltrú, julio 2003

[5] Comunique of the meeting of European Ministers in charge of Higher Education. *Towards the European Higer Education Area.* Praga, 19 de mayo 2001

[6] Conferencia de Rectores de las Universidades Españolas. *La Declaración de Bolonia y su repercusión en la estructura de las titulaciones en España.* 8 de julio de 2002.

[7] Dirección de Investigación y Desarrollo Educativo, Vicerrectoría Académica, Instituto Tecnológico y de Estudios Superiores de Monterrey, *El Aprendizaje Basado en Problemas como técnica didactica*

[8] Joint declaration of the European Ministres of Education. *The European Higher Education Area.* Bolonia, 19 de junio 1999

[9] Larsen, L.B., Andersen, S.K., Fink, F., Granum, E., *Teaching HCI to Engineering Students Using Problem Based Learning.* Interact Workshop of IFIP WG 13.1, Zurich (Suiza), septiembre 2003

[10] Oliver, J., El futuro de la formación de los profesionales informáticos. IX Jornadas de Enseñanza Universitaria de la Informática. 2003

[11] Problem-Based Learning Initiative. *Generic Problem-Based Learning Essentials*. http://www.pbli.org/pbl/generic_pbl.htm

[12] Rodon, A., *Metodología del Caso y Aprendizaje Basado en Problemas.* Institut de Ciències d l'Educacio, Universitat Autónoma de Barcelona

[13] San Diego State University, *The Learning Tree, Problem Based Learning*. *http://edweb.sdsu.edu/clrit/learningtree/Ltree.html*

[14] Sánchez, F., Sancho, M., *Repercusiones el futuro espacio europeo de educación superior sobre las titulaciones universitarias de Informática en España*. IX Jornadas de Enseñanza Universitaria de la Informática. 2003

[15] Striege, A., Rover, D.T., *Problem-Based Learning in an Introductory Computer Enineering Course*. 32nd ASEE/IEEE Frontiers in Education Conference, Boston (USA), noviembre 2002

[16] The American Acreditation Board for Engineering and Technology (ABET). http://www.abet.org

[17] The Joint Task Force on Computing Curricula ACM/IEEE Computer Society, *Computing Curricula*. Diciembre 2001

[18] Uden, L., Dix, A., *Life long learning for software engineers.* ICEUT 2000, IFIP Word Computer Congress, Beijing (China), agosto 2000

[19] Valero, M., Díaz, L.M., *Evaluación continuada a un coste razonable*. IX Jornadas de Enseñanza Universitaria de la Informática. 2003

[20] Woods, D.R., Felder, R.M., Rugarcia, A., Stice, J.E., *The future of engineering education. Developing Critical Skills*. Chem. Engr. Education, 34(2), 108-117 (2000).

Experimento docente en primero de informática: aprendizaje y evaluación centrados en el alumno

Juan José Escribano Otero, Estrella Gómez Fernández
Maria Teresa Villalba de Benito,
Dpto. de Sistemas Informáticos
Escuela Superior Politécnica
Universidad Europea de Madrid
Manuel Ortega Ortiz de Apodaca
Dpt. de Sistemas Informáticos y Programación.
Facultad de Informática
Universidad Complutense de Madrid

E:mail: juanjose.escribano@uem.es,
estrella.gomez@uem.es,
maite.villalba@uem.es,
manu.ortega@fdi.ucm.es

Resumen

La convergencia en la universidad europea, iniciada con el Tratado de Bolonia, está provocando en la universidad española la revisión de los planes de estudios ofertados tanto en el caso de pre-grado como en el de post-grado. El acercamiento al desarrollo profesional de los planes de estudio, llevó a la Escuela Superior de Informática de la Universidad Europea de Madrid (actualmente convertida en el Área de Informática de la Escuela Superior Politécnica de la misma Universidad) a incluir una asignatura llamada *Introducción a la informática en Red* donde, además de desarrollar conceptos fundamentales de las redes de ordenadores (y de Internet muy en especial), se pudiera experimentar con modificaciones en la metodología docente y en el sistema de evaluación, con el objetivo de hacer parte activa al alumno en ambos aspectos.

1. Introducción

La declaración firmada en Bolonia por 29 ministros de Educación, el 19 de junio de 1999 [3], supone el punto de partida de una de las revoluciones más trascendentales de la educación superior en Europa. Basándose en el informe "Trends in Learning Structures in Higher Education" realizado por Guy Haug y Jette Kirstein [7] para la confederación de conferencias de rectores de la UE, los ministros reunidos emiten un documento [3] en el que se definen las líneas estratégicas, con el objetivo de conseguir un espacio educativo superior único para toda al UE, en el año 2010.

En la declaración se fijan seis objetivos estratégicos que afectan fundamentalmente a la estructura de las titulaciones, y a la movilidad de pre y post graduados. En resumen, se trata de potenciar al máximo la posibilidad de que los estudiantes puedan cambiar de estado durante los estudios de pre-grado y post-grado, y que los títulos así obtenidos tengan validez en toda la UE. Como requisito indispensable para ello, hay que unificar la forma de medir la carga lectiva que lleva aparejada una titulación. La propia declaración señala la necesidad de establecer un sistema de créditos, similar a los ECTS de la iniciativa Erasmus.

Por mandato de la CRUE, el grupo de trabajo presidido por Domingo Docampo, Rector de la Universidad de Vigo, elabora el informe titulado "El crédito Europeo y el sistema educativo Español"[6], donde se analizan las consecuencias de la implantación de la nueva métrica en los planes de estudios españoles. Lo que queda claro en el informe citado es que la nueva métrica (pervive el nombre de crédito) debe valorar no sólo, como ocurre actualmente, el número de horas que el alumno emplea en asistir a clase, ya sea de teoría o de prácticas, sino que es necesario computar también aspectos como el número de horas que el alumno emplea en estudiar, en

confeccionar trabajos y, de forma general, en todas aquellas actividades relacionadas con su actividad formativa (algunos reivindican, incluso, el tiempo empleado en realizar fotocopias). Se incorporan también otro tipo de actividades encaminadas a que el alumno no sólo adquiera conocimientos sino que también adquiera determinadas destrezas (competencias es la palabra de moda) que le sean de utilidad en su posterior trayectoria profesional.

Es evidente que la nueva métrica y la limitación por curso de créditos ECTS tienen consecuencias importantes en la tarea del profesor. Ahora, el acento se pone en el trabajo del estudiante y no sólo en las horas de clase que recibe. Lo importante es, por tanto, que "el alumno aprenda" y no que "el profesor enseñe". El profesor pasa de ser un transmisor y evaluador de conocimientos a ser la persona que "conduce y asiste" el aprendizaje del alumno. Es necesario, por tanto, cambiar la dinámica de las clases, de forma que éstas no sean meros actos de transmisión/recepción de información, convirtiéndolas en actos en los que el alumno tiene un protagonismo activo.

En la Universidad Europea de Madrid hemos considerado que la asignatura *Introducción a la Informática en Red*, por sus especiales características, constituye un banco de pruebas perfecto para trabajar en experiencias innovadoras dentro del área de Informática que nos permitan reorientar nuestra práctica docente y adaptarla a las nuevas exigencias emanadas del compromiso de Bolonia.

2. Asignatura de Introducción a la Informática en Red

A menudo, los alumnos que acceden al primer curso de una de las titulaciones de informática aportan una visión extraña de lo que supone o debe suponer el estudio de una ingeniería. Probablemente el hecho de que la misma herramienta (el ordenador) que usan para sus juegos, sus comunicaciones y su búsqueda de información coincida con la herramienta de la Informática por antonomasia produce una distorsión en el nuevo universitario sobre la profundidad y rigor de la computación como disciplina.

Por otra parte el primer curso de cualquier ingeniería es, probablemente, el que encierra mayor carga de contenidos teóricos en casi todas las titulaciones. Por ejemplo, en la Universidad Europea de Madrid, en los planes de estudio de 1996, en el conjunto de las tres carreras impartidas por el área de Informática – Ingeniero en Informática, Ingeniero Técnico en Informática de Sistemas e Ingeniero Técnico en Informática de Gestión- la media de créditos impartidos en un laboratorio suponía un 14% sobre el total de ese curso. Esta circunstancia no está exenta de lógica ya que dichos conocimientos deben ser los pilares sobre los que se sustenten las posteriores prácticas.

Sin embargo esta situación provoca, en primer lugar, que el alumno se encuentre con mucha carga teórica en su primer año, justo cuando más motivado está para experimentar con ordenadores. Además, el alumno de informática ve cómo los compañeros que cursan otras carreras utilizan el ordenador más que él y por consiguiente "saben más informática que él".

Todo esto motiva (junto con otros factores, naturalmente) que la tasa de abandono del primer año supere con creces a la de cualquier otro curso.

Brandt en su artículo "Constructivism: Teaching for Understanding of the Internet" [12] sugiere el camino a seguir para solucionar ambos problemas. Primero, utilizar la búsqueda de información en Internet como objeto de estudio para familiarizarse con la herramienta que es la computadora y segundo, utilizar metodologías docentes basadas en el constructivismo [13].

El constructivismo defiende que el aprendizaje se produce en virtud de que los "aprendices" construyen el conocimiento dando sentido a experiencias que realizan, en términos de algo que ya conocen. Para ello, los modelos mentales ya conocidos sirven para transformar la información en conocimiento ampliando a su vez el ámbito de los propios modelos. En este contexto *"los modelos mentales son representaciones cognitivas de elementos junto con relaciones entre esos elementos"* [14][15].

Por ello, en el área de Informática de la UEM se introdujo como asignatura obligatoria de las tres titulaciones ofertadas la asignatura *Introducción a la Informática en Red* en la revisión del Plan de Estudios que se publicó en el año 2000. Con esta inclusión, el número de créditos impartidos en el laboratorio en este primer curso representa un 24% sobre el total.

La introducción de esta asignatura nos permitió suavizar los contenidos teóricos del primer curso y, a la vez, ensayar nuevas metodologías docentes que, por razones obvias,

no nos atrevíamos a realizar en asignaturas como la Matemática Discreta o las estructuras de datos. No en vano, los ingenieros son profesionales que "construyen" soluciones a los problemas del ámbito de su ingeniería (los navales consiguen que trozos de metal floten y los aeronáuticos que vuelen).

Es sobre esta asignatura, su temario, su necesidad de coordinación entre profesores, su metodología docente, su sistema de evaluación y, sobre todo, sobre los objetivos tanto de contenidos como extra-académicos sobre lo que versa este artículo, confiando en que lo en él expuesto pueda servir de, al menos, herramienta de reflexión y punto de partida para la confección e impartición de otras asignaturas de primeros cursos con contenidos fuertemente prácticos, sin que por ello se descuide el necesario nivel académico de una enseñanza superior.

Como se puede consultar en los planes de estudios [2], esta asignatura es obligatoria, de primer curso, primer semestre, 4H/semana (6 créditos) para las tres titulaciones de informática actualmente ofertadas por la Escuela de Informática. Complementa un año cargado de matemáticas (Matemática discreta, Álgebra, Análisis), Física, asignaturas introductorias de Programación (Introducción a la Programación) y otros laboratorios (Laboratorio de programación, Laboratorio de Computadores).

2.1. Objetivos y dinámica de la asignatura

Con el propósito de centrar el contexto en el que se han aplicado las diversas ideas innovadoras, este epígrafe pretende presentar de forma esquemática las directrices de la asignatura.

Los objetivos que se persiguen se pueden resumir en:

1.-Familiarizar al alumno con el trabajo en red, introduciendo los conceptos generales sobre redes y muy particularmente sobre Internet

2.-Enseñar al alumno a utilizar los servicios de Internet para extraer información

3.-Permitir el desarrollo de ciertas competencias personales en el alumno [6]. Dichas competencias representan un subconjunto de las propuestas por la UEM en su informe sobre el particular [7]: Iniciativa, trabajo en equipo, innovación y creatividad, confianza en sí mismo, habilidades comunicativas, responsabilidad, flexibilidad, conciencia de los valores éticos, planificación

Para conseguirlos, la metodología de la asignatura se divide en las siguientes fases:

FASE INICIAL: ejecución, corrección y discusión en clase de un TEST de conocimientos previos. Duración de la fase inicial: una semana (4 horas de clase)

FASE I (Transmisión de conocimientos): clases magistrales explicando conceptos teóricos, junto con pequeñas prácticas para familiarizarse con los programas clientes y con los servicios de Internet, presentaciones en clase y distintas prácticas de grupo en las que se analizan distintas técnicas de búsqueda de información, validación de contenidos, organización y planificación. Duración de la Fase I: 6 semanas.

FASE II (Trabajo en equipo y búsqueda de información): confección de grupos, prácticas de búsqueda de información acerca de temas relacionados con las redes e Internet. Duración: 5 semanas.

FASE III (Exposición y evaluación de trabajos): exposiciones de trabajos en el aula, evaluación de los trabajos y de la exposición por parte del grupo. Duración: 3 semanas.

2.2. Ejercicios intergrupales

Además del trabajo propio de cada profesor con su grupo de clase, se decidió introducir algunas actividades propias de la asignatura que trascendieran el ámbito del aula. Estas actividades pretenden dar coherencia horizontal a la asignatura. Dichas actividades son:

- Cybergymkhana periódica: consiste en un juego de búsqueda de información en Internet, en el que el alumno va superando pruebas mediante búsquedas en la Red [8].
- Competición grupal: pruebas que se resuelven en equipo. Como ejemplo de una de estas pruebas cabe destacar la simulación de una red de ordenadores. Este ejercicio consiste en proponer a cada alumno cumplir el papel de algún elemento de red: un alumno hará de DNS, otro de cable, otro de router, otro de servidor... Una vez montada dicha red humana, se simula una petición de red y se calcula el tiempo de respuesta. Gana la prueba aquel equipo que menos tiempo tarda en servir la petición.

Los objetivos de este tipo de actividades son múltiples:

1.- Fomentar el trabajo en equipo

2.- Fomentar la competitividad de dicho equipo. Cada grupo de alumnos forman un equipo, que compite con las demás clases (tanto de su titulación como de las otras dos ofertadas). Este hecho facilita la cohesión en el aula y la comunicación entre alumnos de todos los planes de estudio de Informática, para potenciar sinergias entre conocimientos e intereses de futuros profesionales afines

3.- Aumentar la responsabilidad del alumno, ya que estas actividades se realizan fuera del horario reglado de clases

2.3 Competencias personales: lo que piden las empresas

Como parte del trabajo desarrollado por el Departamento de Calidad de la Universidad, se realizó un ejercicio que consistía en preguntar a empresas del sector lo que exigían a los titulados para contratarlos. Fruto de este ejercicio (realizado anualmente, desde hace ya tres años) se recogió la siguiente información:

1.- Los titulados en informática, en general, salen bien preparados técnicamente de las universidades.

2.- Se echa en falta en los titulados informáticos el desarrollo de competencias personales muy útiles (e incluso imprescindibles, en la mayoría de los casos) para el cumplimiento de las funciones que se les encomiendan [7].

Se consideró que favorecer el desarrollo de dichas competencias era una necesidad ya desde los primeros cursos. Debido a la ya citada carga teórica de la mayor parte de las asignaturas de los primeros cursos, se estimó oportuno que la asignatura de *Introducción a la Informática en Red* era muy adecuada para experimentar técnicas encaminadas en dicha dirección.

Para cumplir con el objetivo de permitir el desarrollo de competencias en los alumnos, se llevaron a cabo las actividades ya comentadas y se arbitraron dinámicas específicas (explicadas con detalle en este mismo artículo) tanto para la adquisición de conocimientos como para la evaluación de los mismos.

2.4 Temario

Un grave riesgo cuando se experimenta con este tipo de técnicas es la pérdida de rigor o profundidad en los contenidos. De hecho, para poder integrar este tipo de actividades donde el alumno adquiere un mayor protagonismo en su propio aprendizaje [3], conviene diseñar un temario lo suficientemente flexible y dinámico como para que se vaya enriqueciendo con las aportaciones de los propios alumnos sin perder en el camino los contenidos teóricos necesarios.

Parte del problema, aunque no todo, se resuelve con la división de la asignatura en las distintas fases. De hecho, la realización de un test inicial el primer día de clase, antes incluso de especificar el temario completo, permite al profesor ajustar el momento de inicio de cada módulo del temario a los conocimientos demostrados por los alumnos.

Así pues, de manera general, la asignatura tiene el siguiente temario:

Módulo A: Redes de ordenadores. En este módulo se pretende presentar la terminología básica de redes, topologías, tecnologías, etc.

Módulo B: Fundamentos de Internet. En este módulo se introduce la terminología y los conceptos fundamentales directamente relacionados con Internet, tanto desde el punto de vista técnico como de uso (historia, arquitectura cliente/servidor, protocolos, etc)

Módulo C: Servicios básicos de Internet. Se presentan los servicios básicos accesibles a través de Internet, así como los programas cliente necesarios para su utilización. (correo electrónico, FTP, telnet, IRC ...)

Módulo D: Servicios elaborados y de búsqueda. Se presentan someramente los diversos servicios (www, gopher, motores de búsqueda, índices y metaíndices, listas de distribución, news...)

Módulo E: Servicios Web de valor añadido. En este módulo, se hace hincapié en el uso de Internet *para algo* (buscadores, portales, mensajería instantánea, comunidades virtuales...)

Módulo F: Servicios a través de Internet. Se pretende presentar al alumno las posibilidades de Internet como elemento dinamizador y moldeador de la Sociedad (juegos on-line, servicios de traducción, diccionarios, teletrabajo y teleenseñanza, administraciones del Estado ...)

Durante la FASE I de la asignatura, el profesor presenta cada uno de los módulos para permitir a los alumnos posteriormente desarrollar por equipos cada uno de ellos en mayor profundidad.

3. Metodología de trabajo propuesta a los alumnos

Como se ha pretendido explicar en este artículo, el flujo de información en esta asignatura sufre un cambio importante, haciendo al alumno elemento clave en su propia adquisición de conocimiento. Esta modificación supone un cambio en el enfoque que el profesor tiene sobre su papel en el proceso de aprendizaje y en el reparto de tiempo de dedicación al mismo.

3.1 Apuntes de la asignatura

Los apuntes considerados como básicos por parte de los profesores (no hay que olvidar que son varios los implicados en la impartición de esta asignatura que pertenece a tres planes de estudio) deben estar disponibles para los alumnos desde la segunda semana de clase. Esto es importante para permitir al alumno que organice sus lecturas desde casi el principio del curso.

Estos apuntes no pueden estar disponibles desde el primer día porque conviene revisar primero los resultados del test inicial por si conviene ampliar o suprimir algún tema, dependiendo del punto de partida que el profesor estime conveniente.

Como parte de los apuntes de la asignatura, se incluyen aquellos trabajos de años anteriores que se estimen oportunos, cumpliendo este material una triple función:

1.-Demostrar al alumno la importancia que se le otorga al trabajo que deberá realizar, ya que puede formar parte de los contenidos ofrecidos a los alumnos venideros

2.- Mostrar mediante ejemplos qué se considera un buen trabajo. Para cumplir correctamente este objetivo, conviene poner a disposición de los alumnos más de un trabajo de años anteriores con diferentes características e intentar evitar así la aparición de clones en cuanto a estructuras de los mismos.

3.- Permitir al profesor una mayor flexibilidad sin necesidad de redactar a última hora nuevos contenidos.

3.2. Confección del trabajo

Los trabajos de los alumnos son la pieza clave tanto de la metodología docente como del sistema de evaluación y por lo tanto se pone especial énfasis en su realización. Las características más destacables de dichos trabajos son:

1.- Se realiza obligatoriamente en equipos compuestos por un mínimo de tres personas y un máximo de 5. De esta manera se potencia la necesidad de coordinación de esfuerzos entre los alumnos y su responsabilidad

2.- Cada grupo elige un tema distinto, todos relacionados con el temario. De esta manera cada grupo trabaja en una dirección distinta completando los apuntes dados al principio del curso en uno de sus aspectos que luego expondrá a toda la clase.

El proceso se considera tan importante como el producto. Para ello, el trabajo consta de cinco puntos, tres de ellos representan el *producto* de dicho trabajo (texto del trabajo, direcciones favoritas comentadas, direcciones visitadas), otro es la *herramienta de comunicación* (el guión de la presentación, generalmente realizada en un programa para presentaciones gráficas) y el último comenta y explica el *proceso* ("cómo se hizo", documento que recoge las búsquedas e impresiones sobre las mismas de los alumnos).

3.3. Ejecución de la presentación

Otro de los elementos clave para el desarrollo de las competencias de carácter personal en los alumnos de primero es la necesidad de exponer su trabajo a todo el grupo. Para ello es para lo que se les ha pedido, como parte de su trabajo, la realización de una presentación gráfica. Dicha presentación supone, en el momento de su redacción, un interesante ejercicio de síntesis, ya que los trabajos suelen ser extensos debido a la gran cantidad de información que en Internet se encuentra sobre sí misma.

Una vez redactada dicha presentación, cada grupo dispone de media hora del horario de la asignatura para mostrarla a sus compañeros.

Al auditorio de estas charlas (los compañeros de clase de los ponentes) se le entrega un pequeño formulario donde pueden evaluar de manera anónima diversos aspectos de la presentación (claridad de la exposición, calidad de la presentación, calidad del material entregado para seguir la ponencia, interés despertado por la misma, etc.) dichas evaluaciones las recoge el profesor y, tras un vistazo a los resultados (y la correspondiente toma de las notas que estime oportunas) se las entrega al grupo de ponentes para su estudio.

Este feedback (saber qué opinan sus compañeros sobre la presentación que les acaban de hacer) es de indudable interés pedagógico y dota al alumno de un mayor conocimiento sobre qué aspectos debe reforzar en el futuro cuando se encuentre en situaciones análogas. De hecho, los comentarios recogidos por los profesores son altamente positivos, teniendo de paso un efecto secundario interesante: el alumno valora con conocimiento de causa el esfuerzo que supone la preparación de las clases por parte de sus profesores. Ni que decir tiene que algunos de los comentarios recibidos provocan una sonrisa en su profesor (creí que esto de dar clase era más fácil; los del fondo no me han hecho ningún caso). Esta *evaluación entre iguales* (rápida y anónima) ofrece al alumno una visión clara de sus puntos fuertes y débiles como orador, así como qué aspectos deberá cuidar más en el futuro para preparar una presentación pública. Después de cada exposición y a la vista de los comentarios hechos por sus compañeros de clase, los alumnos que acaban de exponer su trabajo inician un pequeño debate sobre el mismo.

3.4 Confección de batería de preguntas para el examen final

Con las acciones expuestas hasta este momento, se pretende involucrar al alumno de manera directa en su proceso de adquisición de conocimiento. No obstante, se juzgó interesante hacerle partícipe también en su proceso de evaluación...

Con este objetivo, se les propuso a los alumnos, como un ejercicio más de la asignatura la confección de un total de 30 preguntas de tipo test, con cuatro alternativas y una sola respuesta correcta, divididas en dos grupos: 20 preguntas relativas al temario de la asignatura y 10 preguntas más que versaran sobre su propio trabajo.

El compendio de todas las preguntas así conseguidas se sometió a un proceso de revisión por parte del cuadro docente. Fruto del mismo, se consiguió una batería de más de 700 preguntas de tipo test válidas. De entre estas preguntas, cada profesor elegiría 30 preguntas que conformarían el examen final de la asignatura. Además, este fichero con todas las preguntas se puso a disposición de los alumnos una semana antes de la realización del examen. Con esta acción se pretende dirigir el estudio del alumno a aquellos temas que se consideran relevantes, pero apoyándose en sus propias propuestas.

Con esta práctica se pretendía, además de hacer parte al alumno como co-autor de su propio examen final, reforzar varias habilidades útiles (responsabilidad, capacidad de síntesis y comprensión de textos largos, entre otras) para los alumnos, así como asegurar (en la medida de lo posible) una *segunda vuelta* al estudio de los apuntes de la asignatura.

4. Sistema de evaluación

El sistema de evaluación propuesto para esta asignatura tiene en cuenta, como es lógico, todas las actividades desarrolladas por los alumnos, aunque de manera ponderada.

Algunas de dichas actividades se realizaron en grupos, mientras que otras son de carácter individual. Esta dualidad provoca que el sistema de evaluación debe contemplar diversos aspectos.

El trabajo hecho en equipo y la presentación del mismo son las piezas claves del sistema de evaluación de la asignatura, junto con el examen final. El resto de las actividades evaluadas (prácticas de búsqueda, confección de preguntas de tipo test, pruebas objetivas intermedias, asistencia y participación en clase) sirven para *personalizar* la nota de cada alumno.

Uno de los riesgos más comunes en un sistema de evaluación basado en actividades de equipos de alumnos (como es el caso), es la posibilidad de que un alumno supere la asignatura sin haber aprendido los contenidos mínimos necesarios, simplemente *parasitando* a sus compañeros de equipo.

Para minimizar este riesgo, se diseñaron una serie de prácticas y pruebas intermedias, de carácter obligatorio, que obligan al alumno a llevar al día la asignatura, adaptándola a su nivel de conocimientos o expectativas, a hacer pequeñas prácticas de trabajo en grupo y a través de ellas conocer al resto de los compañeros y cómo trabajan y también a trabajar con la información a menudo sesgada o errónea que aparece en Internet y a partir de ella crear su propio conocimiento.

Como ejemplo, durante la Fase I de la asignatura, todas las sesiones debían terminar con un ejercicio de búsqueda de las palabras clave de la explicación del profesor. El resultado de dicha búsqueda debía enviarse por correo electrónico a la dirección del profesor. Dichas definiciones, junto con el resto de la materia explicada en clase,

formaba parte del contenido de la asignatura y complementaba los apuntes entregados en las primeras clases del curso.

Con las notas así conseguidas (trabajos en grupo, prácticas individuales, exámenes), cada profesor calcula la nota final, bajo su propio criterio, pero respetando los siguientes principios:

1.- El trabajo es el eje central de la evaluación

2.- La entrega de todas las prácticas es obligatoria

3.- La exposición en clase del trabajo, así como las preguntas redactadas por cada uno, aportan información sobre la participación de cada alumno en su grupo de trabajo.

5. Metodología de trabajo de los profesores

Uno de los principales problemas de una asignatura de estas características es la necesidad de coordinación entre todos los profesores implicados. Este problema, común a todas las asignaturas que tienen más de un profesor, se ve agravado por los cambios introducidos en el flujo de información docente-alumno. Como, además, dichos cambios tienen un evidente carácter experimental, convenía automatizar al máximo dichos procedimientos (coordinación y transmisión de información) para facilitar la exportación de métodos y herramientas a otras asignaturas.

Para todo ello, se ha diseñado un documento Web propio de la asignatura [1] que sirve como portal de entrada tanto a profesores como a alumnos a la documentación asociada, y como principal medio de comunicación entre los alumnos y sus profesores.

Además, existe un servicio de FTP anónimo donde los profesores de la asignatura dejan todo el material que estiman oportuno y los alumnos pueden recoger con facilidad

No obstante, existen cinco grupos distintos (y por lo tanto cinco profesores) que, además de la información común a todos ellos, pueden requerir información específica (cada grupo de alumnos debe poder acceder a los contenidos de los trabajos que se presentaron en su clase), por lo que además de un directorio en el FTP anónimo de la asignatura para cada uno de los grupos, se creó otro directorio, con acceso restringido a los profesores de la asignatura, donde poder hospedar toda la información común, permitiendo a cada profesor colgar su material específico en el directorio de su grupo y en este otro (llamado coordinación) aquellos documentos o referencias de carácter común.

Para facilitar la publicación de este material común, se desarrollaron dos sencillos programas shell-script que permiten publicar de una sola vez en todos los directorios un archivo alojado en coordinación, mientras que el otro elimina los enlaces de dichos directorios, dejando el archivo original accesible para los profesores para su posterior uso.

Como mecanismo de coordinación entre los profesores se creó una lista de distribución, llamada convenientemente iir, y se establecieron reuniones periódicas para compartir información y experiencias.

6. Evaluación de resultados

Debido al carácter experimental del método, los profesores de la asignatura pidieron ayuda al departamento de Calidad de la universidad. El objetivo era realizar preguntas específicas, en los cuestionarios de satisfacción que anualmente se reparten entre todos los alumnos de la universidad, sobre el sistema de evaluación y la percepción sobre la cantidad y calidad de aprendizaje conseguido.

Como se puede observar en [11], los datos obtenidos parecen indicar que los alumnos de Introducción a la Informática en Red se sienten satisfechos con la cantidad aprendizaje (3.72 de 5) y la calidad del mismo (3.89). En cuanto al sistema de calificación empleado los resultados también parecen expresar la conformidad de los alumnos (4.00)

Por último, el índice de aprobados es muy elevado (87% de entre los presentados) mientras que el de abandonos es sensiblemente inferior al de otras asignaturas del mismo curso (8% frente a un 13%,como media en primer curso).

7. Conclusiones y trabajos futuros

El principal problema que los autores pueden encontrar en la aplicación de este tipo de técnicas activas es la sobrecarga de trabajo a la que se somete al docente. La profusión de prácticas, trabajos en equipo y actividades dentro y fuera del aula, aumentan la necesidad de labores de *gestión docente.* Para ayudar a paliar este sobreesfuerzo, se plantea -en la ESP- desarrollar para el próximo curso varias herramientas informáticas,

entre las que cabe destacar: un generador aleatorio de test y un sistema de apoyo a la gestión del avance de las prácticas a través del correo electrónico. El generador de test estará a disposición de los alumnos como herramienta de autoevaluación. De este modo conseguimos dos cosas, por una parte aprovechar la numerosa colección de preguntas tipo test que hemos ido recopilando y por otra motivar al alumno a repasar los contenidos teóricos de la asignatura. La segunda herramienta facilitara la transmisión de enunciados de prácticas, la solución de las mismas una vez cumplido el plazo de entrega y una personalización de dichos plazos para adecuarse a los distintos ritmos de estudio de los alumnos.

Aunque, como se ha comentado, el porcentaje de aprobados es muy elevado, nos faltan otros indicadores objetivos que nos permitan evaluar el éxito o no de la experiencia. Por el momento, sólo podemos certificar que los alumnos están muy satisfechos con la dinámica de la asignatura y que el porcentaje de aprobados es altísimo. Esto que, en sí mismo, puede ser un buen indicador de calidad, también puede ser rebatido argumentando que, en definitiva, la asignatura es una "tradicional María".

Como carecemos de grupos de control que nos permitan comparar el rendimiento de los alumnos siguiendo esta metodología, y otra "mas tradicional" es necesario encontrar y desarrollar métodos e indicadores que nos permitan dar respuesta a lo anterior y a algunas preguntas que, a nuestro juicio, están todavía sin contestar.

1)- ¿Es aplicable esta metodología a otras asignaturas?

2)- ¿En que forma modificaría el perfil de nuestros titulados la introducción generalizada de esta metodología en otras asignaturas del grado?

3)- ¿Sería el perfil detectado en 2) "mas Boloñés" que el actual?

Referencias

[1] http://www.esi.uem.es/asignaturas/IIR.htm *Documento Web de la asignatura*

[2] *http://www.uem.es/esp/ Documento Web de la Escuela Superior Politécnica de la UEM*

[3] The Bologna declaration on the European space for higher education: an explanation. http://europa.eu.int/comm/education/socrates/erasmus/bologna.pdf

[4] Brandt Scott, Constructivism: Teaching for Understanding of the Internet. Communications of the ACM. Octubre 1997.

[5] Bricall , Joseph M. *Universidad 2 mil*. Conferencia de Rectores de las Universidades Españolas(CRUE). 2000.

[6] Docampo Amoedo, Domingo, et al. *El crédito Europeo y el sistema educativo Español*. http://www.crue.org/espaeuro/encuentros/credito-vigo2002_.pdf

[7] Iñigo Alvarez, et al. *Plan de desarrollo de competencias en el alumnado de la Universidad Europea – CEES*, documento interno, 2002.

[8] Pedro José Lara et al. *Nuevas técnicas de aprendizaje: Cybergymkhana*, JENUI 2002

[9] Peter J. Denning and Robert Dunham.The Profession of IT COMMUNICATIONS OF THE ACM November 2001/Vol.44, No.11.

[10] Haugh, G. y Kirstein, J. Trends in Learning Structures in Higher Education. http://www.rektorkollegiet.dk/sider/publikationer/english/edutrends.htm

[11] Juan José Escribano Otero et al. *El proceso de aprendizaje: herramienta para el desarrollo de competencias profesionales en primero de informática,* JENUI 2003.

[12] Brandt, D.S. Constructivism: Teaching for Understanding of the Internet". Comm. ACM, Vol 40, nº 10, Octubre 1997. pp 113-117.

[13] Tobin, K., and Tippins, D. "Constructivism as a referent for teaching and learning". K. Tobin, Ed. The Practice of Constructivism in Science Education. AAAS Press, Washington, D.C., 1993

[14] Belanger, F. and Van Slyke, C. "Abuse or Learning?"Comm. ACM, Vol. 45, n.º 1, Enero 2002.

[15] Ansari, y. And Simon, H. "The Theory of Learning by Doing". Psych. Rev. 86, 2 (1979), 124 –140.

CEDI 2005

Adaptación al EEES y actividades docentes relacionadas

Adaptación al sistema ECTS: resultados de una experiencia

Rafael Molina Carmona, Juan Antonio Puchol García
Dpto. Ciencia de la Computación e Inteligencia Artificial
Universidad de Alicante
Apdo. Correos 99
03080-Alicante
{rmolina, puchol}@dccia.ua.es

Resumen

Presentamos una propuesta docente adaptada al Espacio Europeo de Educación Superior (EEES) para una asignatura ya implantada en el plan de estudios de Ingeniería Informática en nuestra Universidad. Para ajustarnos al EEES proponemos un conjunto de competencias que los alumnos deberían adquirir y una serie de actividades cuyo objetivo es proporcionarles estas competencias. Nos adaptamos a la nueva filosofía de los créditos ECTS, presentando la distribución de tiempos para cada actividad. Presentamos, además, los resultados y la evaluación de la implantación, y proponemos un esquema de adaptación que pueda servir como referencia para otras asignaturas que se encuentren en ese proceso.

1. Introducción

En 1998 se inició el proceso de convergencia en Europa entre los sistemas de educación superior de los distintos países de la Unión. En las Declaraciones de la Sorbona [8] y de Bolonia [9], los estados miembros se comprometían a coordinar las políticas educativas para establecer el llamado Espacio Europeo de Educación Superior (EEES). Entre sus objetivos destacan la adopción de un sistema de titulaciones comprensible y comparable, su estructuración en dos ciclos, el establecimiento de un sistema de créditos común (ECTS), y la promoción de la movilidad y de la cooperación europea.

Estos objetivos han sido refrendados por diversos organismos españoles (Conferencia de Rectores [2, 3, 4] Ministerio de Educación [7], etc.). La convergencia europea supone, sin duda, el cambio principal al que se ha visto sometido el sistema universitario en los últimos decenios.

Con la vista puesta en la adaptación al nuevo contexto europeo, presentamos en las X Jornadas de Enseñanza Universitaria de la Informática [10] un plan para la adaptación de la asignatura Gráficos Avanzados y Animación del programa de estudios de Ingeniería en Informática. Presentamos ahora los resultados de la implantación de aquella propuesta y evaluamos este proceso, con el ánimo de servir como ejemplo de la problemática a la que nos enfrentamos los docentes ante los cambios que se nos presentan.

El documento se organiza de la siguiente manera: En el apartado 2 recordamos nuestra propuesta de métodos didácticos para los gráficos por ordenador adaptado al sistema de créditos ECTS. En el apartado 3 se ofrecen los resultados de la implantación y qué valoraciones nos sugiere este proceso. Por último, presentamos conclusiones y expectativas futuras.

2. Propuesta de adaptación al sistema de créditos ECTS

El Sistema Europeo de Transferencia de Créditos (ECTS) establece un sistema de equivalencias y reconocimiento de estudios, que garantizará la transparencia y el reconocimiento académico [12]. Los principios en los que se basa son:

- Los créditos ECTS representan el volumen de trabajo efectivo del estudiante y el rendimiento con calificaciones comparables.
- La información sobre los programas de estudios y los resultados se ofrece a través de documentos con formato normalizado.

El sistema educativo español posee una estructura de créditos no coincidentes, en su filosofía, con los ECTS: los créditos se asocian a horas docentes o de contacto con el profesor, lo que implica un importante cambio para los docentes al valorar no sólo el trabajo que se realiza en la preparación de las clases, en la evaluación de los conocimientos de los alumnos, en su atención durante las tutorías, etc., sino también el volumen de trabajo requerido para que el estudiante alcance el nivel de conocimientos y

competencias que le permitan superar la materia. Debe incluir, por lo tanto, lecciones magistrales, trabajos prácticos, seminarios, períodos de prácticas, trabajo de campo, trabajo personal, así como los exámenes u otras formas de evaluación.

Diversos estudios [7, 12] establecen en 60 créditos el volumen de trabajo de un estudiante a tiempo completo durante un curso (30 créditos por semestre). Considerando una actividad académica de 40 semanas al año y una carga de 40 horas semanales, se establece para el crédito europeo un volumen de trabajo de 25 a 30 horas, de las que 10 o 12 horas deben ser de contacto con el profesor.

Aunque la adopción del sistema de créditos ECTS es uno de los aspectos del EEES que más puede afectar a la labor cotidiana del docente y al desarrollo de las clases, otros elementos también afectan a la forma en que debemos plantearnos la actividad educativa: diversidad de los programas y su flexibilidad, desarrollo de conocimientos, aptitudes, habilidades y técnicas que preparen al estudiante para analizar, comprender y asimilar conocimientos futuros, aptitudes y habilidades para la comunicación, trabajo en equipo, etc.

Teniendo en mente estos objetivos, proponemos introducir en la docencia de los Gráficos nuevas actividades que tengan en cuenta este espíritu. En concreto, presentamos un plan para adaptar la asignatura de Gráficos Avanzados y Animación, optativa de 6 créditos (3 de teoría y 3 de prácticas), concebida como continuación de otra obligatoria de 4,5 créditos (Gráficos por Computador) y con características adecuadas para esta experiencia: número limitado de alumnos (no más de 50), generalmente de los últimos cursos y contenidos establecidos pero avanzados.

2.1. Propuesta de actividades

Proponemos una serie de actividades junto con las competencias que están relacionadas con ellas, con los objetivos de motivar al alumno, hacerles entender mejor los contenidos de la asignatura, propiciar el pensamiento creativo y enseñar a aprender por encima de enseñar conocimientos. Proponemos también una evaluación continua, proporcionarles una formación que les permita profundizar por ellos mismos y orientar la enseñanza hacia la interdisciplinariedad [6, 11].

Para llegar a cumplir estos objetivos, proponemos realizar una serie de actividades que complementen las clásicas de clases teóricas y prácticas [1, 5]. Estas actividades son:

- Lección magistral: su función principal es la transmisión de conceptos y la consecución de los objetivos conceptuales, más teóricos que metodológicos. Para paliar los problemas de pasividad debemos complementarla con otras actividades e incorporar mayor interacción.
- Prácticas con el ordenador: En disciplinas con alto contenido tecnológico tienen un papel central. Sus objetivos son la aplicación de los principios teóricos al diseño, implementación y prueba del software y el hardware, la introducción de métodos experimentales, y procesos que llevan a un buen saber hacer en la computación.
- Resolución de problemas: Permite comprobar la solidez de los conocimientos teóricos y si han sido comprendidos. Se enfatizan los aspectos de instrumentación y uso y fomentan la participación, la integración y el pensamiento crítico.
- Búsqueda bibliográfica: Propicia el desarrollo de habilidades como la lectura de artículos de diferentes fuentes, su comprensión y síntesis, y despierta el interés del alumno por temas que no pueden desarrollarse durante las sesiones normales de clase y por la investigación en ese campo.
- Trabajos en grupo: Complementan los conocimientos teóricos o prácticos. Se pretende que el estudiante aporte sus propias ideas sobre la resolución del trabajo. Fomentan el trabajo en equipo y la coordinación de su desarrollo.
- Prueba de evaluación: Afecta tanto al alumno como al profesor. Tiene como objetivos la calificación y la retroalimentación. Las pruebas de evaluación deben realizarse a lo largo todo el proceso, para ver el progreso, vigilar la adquisición del conocimiento y adecuar el proceso de enseñanza.

Con estas actividades pretendemos desarrollar en los alumnos nuevas competencias que les permitan enfrentarse al mundo postacadémico con plenas garantías. Entre ellas destacamos

1. Adquisición de conocimientos básicos.
2. Mejora de la capacidad de comunicación.
3. Fomentar capacidades de análisis y síntesis.
4. Mejora de la capacidad de abstracción.
5. Mejora del análisis y la jerarquización de la información.

6. Trabajo en equipo.
7. Planificación de la resolución de problemas, en función de los recursos disponibles.
8. Fomento de las competencias instrumentales y tecnológicas.
9. Mejora del conocimiento transcultural e interdisciplinar.
10. Retroalimentación y valoración de los propios conocimientos.

	Competencias									
Actividades	1	2	3	4	5	6	7	8	9	10
Lección magistral	●	•	•	•	•				•	
Prácticas en ordenador	•		●	●			●	●	•	•
Resolución problemas	•		●	●			●	•	•	●
Búsqueda bibliográfica	•	●	●	•	●				●	
Trabajos en grupo	•	●		•	•	●	●	•	●	•
Prueba de evaluación					•		•			●

Tabla 1. Relación entre competencias y actividades.

Cada una de las actividades reseñadas tiene como objetivo mejorar una o varias de estas capacidades. En la tabla 1 presentamos cómo, desde nuestro punto de vista, cada tipo de actividad afecta a la mejora de cada competencia. Un círculo grande indica que es una actividad dirigida expresamente a mejorar esa capacidad. Un círculo pequeño indica también una relación actividad-competencia, aunque de menor entidad. En las celdas en las que no hay círculo, la relación es más débil, aunque evidentemente puede existir.

2.2. Desarrollo temporal de la asignatura

En experiencias de este tipo, partimos de la distribución de créditos actual, impuesta por el plan de estudios vigente. Se dispone de 3 créditos de teoría y otros 3 de prácticas que, tomando como referencia el sistema ECTS, se corresponden con entre 150 y 180 horas de trabajo, de las cuales entre 60 y 72 horas son de contacto con el profesor. Nos plantearemos un volumen de 150 horas totales y 60 horas de contacto con el profesor. Las restantes 90 horas, se dedicarían al estudio y otras actividades planteadas.

Para cada tema, los objetivos y las actividades propuestas y su volumen de trabajo se muestran en la tabla 2. Una programación más detallada puede encontrarse en [10]

	Titulo tema	Objetivos	Actividades	Horas
1	Introducción: el contexto de	• Contextualizar asignatura	**Lección**: Presentación. Conceptos básicos	1
			Estudio y prueba de autoevaluación	5
2	La luz en los gráficos	• Superficies. • Teoría de la luz. • Modelos de Phong, Blinn y Cook-Torrance. • Ilum local-global	**Lección**: Superficies e iluminación. Reflexión. Modelos Phong, Blinn y Cook-Torrance. Ilum local y global	3
			Práctica con ordenador: Comparación de los tres modelos estudiados	2
3	Introducción al problema del aliasing	• Problema del aliasing. • Soluciones propuestas.	**Lección**: Muestreo de señales. Aliasing. Supermuestreo, multimuestreo y muestro estocástico.	3
			Búsqueda bibliográfica: Antialiasing en HW.	6
4	Texturas	• Técnicas de mapeado de texturas. • Métodos de generación de texturas.	**Lección**: Mapeado. Magnificar y minimizar. Captura y compresión. Generación.	3
			Práctica: Diferentes mapeados en OpenGL	4
			Estudio y prueba de evaluación temas 1-4.	13
5	Trazado de Rayos	• Conocer con detalle el Trazado de Rayos. • Determinar las posibilidades de mejora.	**Lección**: Algoritmo básico. Intersecciones. Antialiasing. Definición recursiva y algoritmo práctico. Eficiencia.	4
			Práctica: Implentación de trazador de rayos básico.	6
			Resolución de problemas: Intersecciones varias.	6
6	Radiosidad	• Método básico de Radiosidad. • Identific. escenas adecuadas • Combinar con Trazado de Rayos.	**Lección**: Método de Radiosidad. Formulación de la matriz y de los factores de forma. Combinar con Trazado de Rayos.	4
			Práctica: Prueba de SW.	2
7	Métodos volumétricos de visualización	• Características de la visualización científica. • Método básico de Marching Cubes	**Lección**: Datos científicos. Algoritmo Marching Cubes.	2
			Trabajo en grupo: Algorit. práctico Marching Cubes.	6
			Estudio y prueba de evaluación temas 5-7	16
8	Introducción a la animación	• Fundamentos de la animación por ordenador. • Comparar con la animación clásica. • Características de cada técnica.	**Lección**: Keyframing, Morphing. Animar modelos. Movimiento a bajo nivel.	2
			Búsqueda bibliográfica: Películas de animación	6
			Trabajo en grupo: Buscar reportajes sobre animación y visionar en clase	6
9	Animación de estructuras articuladas	• Conocer qué es una estructura articulada. • Diferencias entre cinemática directa e inversa.	**Lección**: Fig. articuladas. Cinemát. inversa y directa. DOF. Figuras con piernas.	4
			Práctica: Prueba de SW. Animar figura articulada.	6
			Resolución problemas: cinemática directa e inversa de sistema articulado DH.	6
10	Animación de objetos blandos	• Tipos de deformaciones	**Lección**: Objetos poligonales, Bézier y BSpline. Mallas 2D y 3D. Animar caras.	2

10	Animación de objetos blandos	• Animación: deformación a lo largo del tiempo	**Práctica**: Prueba de SW: vestir la figura articulada.	5
11	Animación procedimental	• Animación de fenómenos físicos • Identificar qué podemos hacer • Modelos y técnicas de animación emergentes.	**Lección**: Animar sistemas de partículas, guiadas por el comportamiento y modelos analíticos.	2
			Práctica: Prueba de SW. Animación procedimental	5
			Estudio y prueba de evaluación temas 8-11	16

Tabla 2. Propuesta de temas, objetivos, actividades y distribución temporal.

A todas estas actividades hay que añadir 4 horas de tutorías. En la figura 1 se representa la distribución de tiempos por actividad y por competencia (tablas 1 y 2). Se han considerado sólo las capacidades relacionadas con cada actividad mediante un círculo grande (tabla 1).

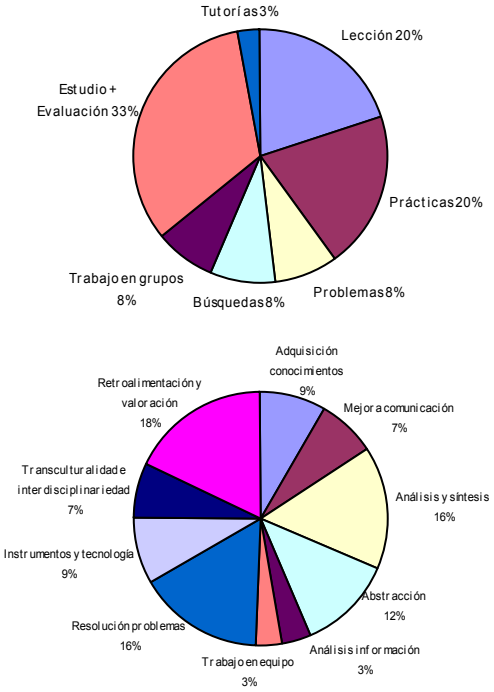


Figura 1. Distribución porcentual de tiempos por actividad (arriba) y por competencia (abajo)

3. Implementación de la propuesta

Durante los cursos 2003-04 y 2004-05 se ha implementado la propuesta anterior. Los resultados los presentamos a continuación.

3.1. Curso 2003-04

Durante el primer curso se implementó la propuesta tal y como se ha presentado, aunque no fue posible desarrollar todos los temas ni todas las actividades inicialmente propuestas: el tema 7 no se impartió y los temas 10 y 11 se esbozaron en una única sesión. Además, algunas de las actividades se propusieron de forma opcional.

Esta experiencia se ha aprovechado para llevar a cabo un amplio estudio que nos ha permitido detectar desajustes y errores de cálculo [13]. Este estudio ha consistido en la realización de un conjunto de encuestas a los alumnos. En concreto se ha optado por incluir dos tipos de cuestionarios: uno cuantitativo cuyo objetivo es medir los tiempos empleados en cada actividad y su dificultad, y otro cualitativo, que permita profundizar en la opinión personal de los alumnos.

Para no sobrecargar a los alumnos con largos cuestionarios después de cada tema, se ha optado por realizarlo sólo para el tema 2, cuyo contenido resulta representativo del resto del temario.

Cuestionario cuantitativo

El cuestionario cuantitativo ha tratado dos aspectos de la asignatura: los créditos teóricos y los créditos prácticos. En cada caso se ha preguntado por el tiempo empleado en la realización de cada actividad y por la dificultad encontrada, que debía valorarse gradualmente con valores entre 1 (poca dificultad) y 5 (dificultad máxima). Entre las actividades por las que se ha preguntado en cada caso destacan: explicación del profesor, estudio de los apuntes, investigación de contenidos, diversos tipos de tutorías, trabajo de laboratorio y preparación y realización de la prueba de evaluación en el caso de la teoría.

En la tabla 3 se muestran los principales estadísticos para el tiempo total empleado en el tema 2, incluyendo teoría y prácticas. Debemos señalar en este punto varias consideraciones:

- La muestra (22 cuestionarios) es relativamente pequeña, pero podemos considerarla válida para una población total de 50 alumnos.
- En la implementación real finalmente se sustituyeron las prácticas de los temas 2, 4, 5 y 6 por una sola práctica más amplia y que es la que se valora en este cuestionario.
- En la propuesta inicial, el tiempo de estudio y la evaluación incluía todo el primer bloque

(temas del 2, 3 y 4). Para el tema 2 podemos estimar como válido una tercera parte del tiempo (4 horas)

Media		32,30
Desv. típ.		19,71
Percentiles	25	18,93
	50	34,97
	75	50,31

Tabla 3. Tiempo total del tema 2

Comparando estos resultados con la previsión inicial podemos comprobar que existe una desviación importante: se preveía un tiempo total de 21 horas (3 horas de lección + 4 horas de estudio y evaluación + 14 horas de las prácticas de los temas 2, 4, 5 y 6), pero los estudiantes han necesitado una media de 32,3 horas.

Si nos fijamos en el alto valor de la desviación típica, detectamos una gran diversidad en los valores. Este hecho lo corrobora también el valor de los percentiles (un 25% del alumnado ha necesitado menos de 19 horas). Estos resultados se explican por la diversidad de conocimientos previos que presentan los estudiantes: al tratarse de una asignatura optativa, los cursos de procedencia son muy variados, de manera que muchos no poseen conocimientos previos afianzados sobre la materia, el lenguaje de programación, etc., mientras que otros han cursado ya materias afines. Además, estas diferencias deben ser más acusadas en estos primeros temas (recordemos que el tema 1 es un repaso de la asignatura obligatoria Gráficos por Computador y no debe suponer esfuerzo a los alumnos que ya la han cursado).

		Teoría Tiempo total	Prácticas Tiempo total	Tiempo Total Tema 2
Media		15,05	17,25	32,30
Desv. típ.		10,67	11,80	19,71
Percentiles	25	8,75	6,75	18,93
	50	15,32	18,00	34,97
	75	18,06	23,25	50,31

Tabla 4. Tiempo del tema 2 en teoría y prácticas

En la tabla 4 se muestran los resultados desglosados en teoría y prácticas, donde vemos que la mayor parte de la desviación se produce en la teoría: habíamos previsto 7 horas y la mayoría de alumnos necesita más del doble. En el caso de las prácticas la diferencia no es tan acusada: frente a las 14 horas previstas, la media es de 17 horas.

Si analizamos los resultados de una forma más pormenorizada podemos intuir dónde se encuentran las diferencias (tabla 5): la lección magistral se alargó algo más de lo previsto (4 horas frente a 3 previstas) y los alumnos han necesitado más de 4 horas para preparar la prueba de evaluación (estudio de apuntes, investigación de contenidos y preparación para la prueba).

Los tiempos empleados en tutorías no han sido incluidos porque son muy poco representativos. Esto confirma que los alumnos hacen muy poco uso de las tutorías, y no aprovechan su alta utilidad. En nuestra opinión, cuando el proceso de adaptación esté más avanzado, quizás los alumnos perciban de una manera más clara la utilidad de este recurso. En cualquier caso debemos reflexionar sobre ello.

		Teoría Explicación profesor	**Teoría** Estudio apuntes clase	**Teoría** Investigación contenidos fuera clase	**Teoría** Preparación prueba evaluación	**Práctica** Explicación profesor	**Práctica** Trabajo laboratorio	**Práctica** Realización trabajo individual
Media		3,93	2,81	1,51	3,01	3,39	5,55	5,84
Desv. típ.		3,34	2,39	1,45	2,36	3,95	5,91	6,07
Percentiles	25	,34	1,00	,75	,79	,00	,00	,00
	50	4,00	3,00	1,00	3,00	2,00	5,50	5,00
	75	6,25	4,00	2,00	5,00	5,25	8,00	10,00

Tabla 5. Tiempo del tema 2 desglosado en actividades

La encuesta también contempla un cuestionario sobre las actividades que suponen una mayor dificultad para los estudiantes. En el ámbito de la teoría las cinco actividades con mayor dificultad media son:

- Lección por parte del profesor (2,45)
- Estudio de los apuntes tomados en clase (2,45)
- Investigación de contenidos (2,55)
- Preparación exclusiva para la prueba (2,36)
- Realización de la prueba (2,18)

De estas actividades, resulta curioso que los alumnos consideran que la realización de la

prueba de evaluación es la que menos dificultad tiene, reconociendo que, una vez comprendidos los conceptos, la prueba no es difícil.

A continuación se presentan las actividades prácticas que suponen una mayor dificultad. Son, con mucha diferencia, las siguientes:

- Explicación por parte del profesor de cuestiones metodológicas y aplicación (2,77)
- Trabajo en el laboratorio (2,91)
- Trabajo práctico individual (3,09)
- Aprendizaje y tareas on-line (2,59)

De los datos de dificultad obtenidos (en muchos casos cercanos o incluso superiores a 3), se sigue que, en general, las prácticas superan a la teoría en dificultad. Los alumnos encuentran especialmente complicado el trabajo fuera del aula, al no tener la asistencia del profesor.

Cuestionario cualitativo

El cuestionario cualitativo es más flexible (el estudiante expresa con sus propias opiniones) pero tiene un tratamiento analítico más complejo. Se ha preparado una encuesta con preguntas abiertas para las cuatro actividades siguientes:

- Lección del profesor en clase (teoría)
- Preparación material, recopilación y estudio.
- Lección del profesor en clase (prácticas)
- Realización de la práctica (en clase y fuera de ella, consulta de material, aprendizaje del lenguaje de programación, etc).

Para cada una de estas actividades, se han planteado cuatro preguntas abiertas:

- Dificultad encontrada y sus causas
- Tiempo y esfuerzo empleado en el aprendizaje
- Rendimiento previsible (relación esfuerzo/calificación)
- Motivos de satisfacción (vivencias positivas o negativas)

Al no tratarse de una encuesta con preguntas cerradas, la diversidad de respuestas ha sido muy amplia. No obstante, hay ciertos patrones que se repiten, por lo que hemos podido clasificar las respuestas en un conjunto discreto de casos. Los resultados se presentan a continuación. Un estudio más extenso puede consultarse en [13].

Para la actividad 1 (Lección del profesor en clase –teoría-) podemos destacar que una buena parte de los alumnos encuentran pocas o ninguna dificultad (casi un 60%) aunque a más de un 30% les resulta difícil debido a la formulación matemática o a la escasez de conocimientos previos. En cuanto al tiempo y esfuerzo empleado en el aprendizaje una mayoría muy importante considera que son normales (50%) o escasos (31%). La mitad de los estudiantes desconocen cuál será su rendimiento, aunque buena parte de ellos se aventuran a suponer cómo va a ser y casi todos son optimistas. En cuanto a los motivos de satisfacción, una inmensa mayoría (más del 54%) valora el aprender cosas nuevas e interesantes seguidos de los que se encuentran satisfechos por haber comprendido el proceso de los gráficos. Curiosamente nadie tiene opiniones negativas.

En cuanto a la segunda actividad (preparar el material, recopilar información y estudiar el tema), los alumnos encuentran las mismas dificultades (en general escasas), y los mismos motivos de satisfacción (el aprendizaje de nuevos conceptos interesantes) que habíamos constatado en el caso de las lecciones de teoría.

Las actividades 3 y 4, relacionadas con las prácticas, les suponen una mayor dificultad, y esto queda reflejado muy claramente en los datos tanto del cuestionario cuantitativo como del cualitativo.

En este sentido son más los alumnos que encuentran dificultades importantes en las explicaciones de las prácticas (más de un 70%), que los que no encuentran ninguna (poco más del 27%). Parte de ellos achacan estas dificultades a las explicaciones inadecuadas del profesor (lo que nos debe hacer reflexionar), aunque la mayoría encuentran otras razones muy diversas. En este mismo sentido el tiempo y el esfuerzo empleado les parece mucho a la mayoría, y ya no son tan optimistas en cuanto al rendimiento, si bien es cierto que muy pocos (el 9%) creen que su rendimiento será escaso. En cuanto a los motivos de satisfacción, siguen siendo positivos: se sienten satisfechos por haber aprendido cosas nuevas e interesantes (17%), o haber comprendido el proceso de generación de los gráficos (13%). No obstante, un 18% no tiene motivos de satisfacción.

La actividad 4, como era previsible, es la que ha supuesto un mayor tiempo y esfuerzo, aunque también les ha reportado muchas satisfacciones. Más de un 40% de los estudiantes consideran que la práctica era demasiado compleja, aunque también es cierto que una parte importante del alumnado (más del 27%) no ha encontrado dificultades especiales. La mitad consideran que el esfuerzo y el tiempo han sido muchos y, aunque la mayoría valora su rendimiento como alto o

normal, la posibilidad de que éste sea bajo la prevén casi un 10%, más del doble que en el caso de la teoría. No obstante, a pesar de las dificultades, los motivos de satisfacción son muchos, especialmente haber aprendido cosas nuevas y haber conseguido que la práctica funcione (objetivo que a muchos les parecía al principio del curso difícil de alcanzar).

Por último, se les pidió a los alumnos que valoraran la asignatura positiva o negativamente y que justificaran esta respuesta. Más de un 77% la valoran positivamente y destacan, como motivos para esta valoración, el haber aprendido cosas nuevas e interesantes y el haber comprendido el proceso que siguen los gráficos por computador. Como aspectos negativos, destacan la complejidad de la asignatura y en particular de las prácticas.

De estos cuestionarios podemos extraer los puntos fuertes y débiles de la asignatura. Entre los primeros, está el hecho de que se trata de una asignatura que les suele resultar interesante y atractiva (buena parte de los estudiantes han valorado muy positivamente el conocer conceptos nuevos e interesantes). En la parte negativa, los alumnos consideran que las prácticas son complejas y descompensadas con la teoría. Debemos destacar, no obstante, que la práctica de este tema ha ocupado más de la mitad del tiempo de prácticas de la asignatura. Puede parecer desproporcionada con el tema 2 (momento de la encuesta) pero no tanto si se compara con el total de la asignatura.

La encuesta cuantitativa realizada nos puede permitir estimar convenientemente los tiempos y la dificultad de cada actividad con el objetivo de la implantación de los créditos ECTS. El cuestionario cualitativo la complementa, profundizando no tanto en los tiempos necesarios, como en las causas de las dificultades que los estudiantes encuentran en la materia.

3.2. Curso 2004-05

La experiencia del curso anterior nos ha permitido ajustar mejor los tiempos previstos. En concreto se han realizado cambios en tres aspectos principales: teoría, prácticas y trabajos en grupo.

En la teoría, el tema 7 ha desaparecido del temario (se ofrece como trabajo en grupo), los temas 10 y 11 conforman una única unidad y se han ajustado las horas y el orden de los temas.

En las prácticas, fuente principal de dificultad, hemos reducido su complejidad pero hemos aumentado su diversidad: las prácticas de implementación han sido simplificadas y se han añadido pruebas de software (comparación de resultados utilizando software ya existente).

Por otro lado, se ha llegado a un compromiso en cuanto a los trabajos adicionales: todos se plantean en grupo para fomentar las relaciones y a cada grupo se le asigna un trabajo distinto que deben exponer en clase. El trabajo puede consistir en una búsqueda bibliográfica, el desarrollo de un tema, un trabajo práctico, la resolución de problemas, etc., tratando un tema propuesto por el profesor o por el alumno. Así no se aumenta tanto la carga de trabajo y se fomentan también las capacidades de expresión de los estudiantes.

En la tabla 6 se resume la distribución del temario por semanas, junto con las prácticas y los trabajos fuera del aula (complemento a las prácticas, trabajos en grupo, estudio, etc). Cada semana supone dos horas de clase en aula de teoría y otras dos en laboratorio.

<table>
<tr><th>Semana</th><th>Teoría</th><th>Prácticas</th><th colspan="2">No presencial</th></tr>
<tr><td>1</td><td>Tema 1. Introducción: el contexto de los gráficos</td><td>Organización prácticas</td><td></td><td>Organización grupos</td></tr>
<tr><td>2</td><td rowspan="2">Tema 2. La luz en los gráficos</td><td rowspan="5">Práctica 1. Implementa-ción modelos de iluminación</td><td rowspan="4">Práctica 1</td><td rowspan="13">Estudio
+
Trabajos en grupo</td></tr>
<tr><td>3</td></tr>
<tr><td>4</td><td rowspan="2">Tema 3. Trazado de rayos</td></tr>
<tr><td>5</td></tr>
<tr><td>6</td><td rowspan="2">Tema 4. Radiosidad</td></tr>
<tr><td>7</td><td rowspan="3">Práctica 2. Implementa-ción trazador de rayos</td><td rowspan="3">Práctica 2</td></tr>
<tr><td>8</td><td>Tema 5. Introducción al problema del aliasing</td></tr>
<tr><td>9</td><td>Tema 6. Texturas</td></tr>
<tr><td>10</td><td>Tema 7. Introducción a la animación</td><td rowspan="2">Práctica 3. Prueba SW: Trazado vs. Radiosidad</td><td rowspan="2">Práctica 3</td></tr>
<tr><td>11</td><td>Tema 8. Animación de estructuras articuladas</td></tr>
<tr><td>12</td><td rowspan="3"></td><td rowspan="4">Práctica 4. Prueba SW: Animación estruct. artic., obj. blandos y procedimental</td><td rowspan="4">Práctica 4</td></tr>
<tr><td>13</td><td rowspan="2">Tema 9. Animación obj. blandos y procedimental</td></tr>
<tr><td>14</td></tr>
<tr><td>15</td><td>Presentación trabajos y videos</td><td></td></tr>
</table>

Tabla 6. Distribución final de actividades por semanas

Aunque en estos momentos no hemos terminado este segundo curso, la experiencia del anterior y el desarrollo actual nos permiten augurar un ajuste en cuanto a los créditos y la dificultad de la materia más acorde con las previsiones iniciales.

4. Conclusiones

Hemos presentado una propuesta de adaptación de una asignatura al EEES, incorporando la filosofía de los créditos ECTS, así como un estudio que nos permite evaluar los resultados. Se ha conseguido una adaptación que todavía requiere de mejoras, pero que nos ha supuesto una experiencia muy válida para procesos posteriores.

Consideramos que para abordar este proceso con éxito es imprescindible plantear una propuesta inicial que nos sirva de punto de partida. La propuesta debe basarse básicamente en la experiencia del docente y es difícil dar una serie de reglas que sean válidas para todas las situaciones. En cualquier caso, consideramos que es importante proporcionar a los estudiantes competencias más allá de las clásicas de transferencia del conocimiento. Esto, junto con el objetivo de conseguir su motivación, aconseja la programación de una gran variedad de actividades, así como su carácter interdisciplinar.

Aun realizando una propuesta inicial consistente, equilibrada y variada, es muy difícil que esta previsión sea adecuada si no se completa con un proceso de revisión continua que nos permita ajustar los tiempos y las dificultades, a la vez que actualizar los contenidos (por otro lado inevitable, dada la evolución constante de este tipo de materias). Durante la adaptación, el uso de encuestas es muy útil para identificar desajustes.

Además, debemos tener en cuenta que todavía seguimos arrastrando la estructura clásica de teoría y prácticas. En un futuro, cuando el marco sea más flexible, podremos mejorar la adaptación. En definitiva, no podemos pretender una adaptación de un curso para otro, pero estas experiencias nos acercan al modelo europeo.

Consideramos que la experiencia puede ser válida para otras materias, especialmente si nos fijamos en el proceso (más que en la propuesta concreta), en el sistema de encuestas y en las ideas sobre actividades que se pueden plantear.

En un futuro, nos proponemos seguir mejorando la propuesta. También pensamos en plantear nuevas actividades, siguiendo un esquema menos rígido: puesto que todavía debemos seguir con la distribución de créditos actual, nos proponemos desvincular algunas horas de teoría y de prácticas para realizar actividades más participativas: resolución de problemas, casos prácticos, visitas a empresas, conferencias de profesionales, campeonatos de juegos, actividades conjuntas con otras asignaturas e incluso con otras carreras, para fomentar el carácter interdisciplinar, etc. También creemos necesario, como ya hemos apuntado, fomentar la utilización de las tutorias.

Referencias

[1] Castillejo, J.L. *Pedagogía técnica*. CEAC, 1987.

[2] Conferencia de Rectores de las Universidades Españolas. *La declaración de Bolonia y su repercusión en la estructura de las titulaciones en España*. Julio 2002

[3] Conferencia de Rectores de las Universidades Españolas. *Declaración sobre el Espacio Europeo de Educación Superior*. Sept. 2003.

[4] Conferencia de Rectores de las Universidades Españolas. *Declaración sobre el Espacio Europeo de Educación Superior*. Oct. 2003.

[5] De La Orden, A. *La evaluación de la enseñanza universitaria*. Ponencias Proyecto de Innovación Educativa, 1988.

[6] Michavila, F. y Calvo, B. *La Universidad Española Hoy*. Editorial Síntesis, 1998.

[7] Ministerio de Educación, Cultura y Deporte. *Documento-Marco: La integración del Sistema Universitario Español en el Espacio Europeo de Educación Superior*. Feb. 2003

[8] Ministros de Educación de Francia, Alemania, Italia y Reino Unido. *Declaración conjunta para la armonización del diseño del EEES (Declaración de la Sorbona)*. 1998

[9] Ministros Europeos de Educación. *Declaración de Bolonia sobre el EEES*. 1999

[10] Molina, R. y Puchol, J.A. *Propuesta docente para gráficos por ordenador en el nuevo contexto europeo de educación*. X Jornadas de Enseñanza Universitaria de la Informática (JENUI 2004). Julio 2004.

[11] Novak, J.D. *Constructivismo humano: Un consenso emergente*. Enseñanza de las ciencias, 6(3), 1988.

[12] Pagani, Raffaella. *El crédito europeo y el Sistema Educativo Español*. Septiembre 2002.

[13] Rizo R., Pujol M., Molina R., Compañ P., Arques P., Puchol J.A., Colomina O., Satorre R., Villagrá C. *Implantación ECTS en FIA-GAA-MFAC. Estudio de valoración del trabajo de los alumnos*. Investigar el Espacio Europeo de Educación Superior. ICE Universidad de Alicante, 2004.

Utilización de un Simulador de Fútbol como Plataforma de Prácticas de Inteligencia Artificial

B. López, M. Montaner, J.L. de la Rosa
Dept. d'Electrònica, Informàtica i Automàtica
Universitat de Girona
Av. Lluís Santaló s/n
17071 Girona
{blopez,mmontane,peplluis}@eia.udg.es

Resumen

Observando las ventajas y los inconvenientes de los sistemas actuales de enseñanza de inteligencia artificial (IA) a ingenieros informáticos, proponemos una metodología que consiste en la utilización de un simulador de fútbol (JavaSoccer) para fomentar la comparación e integración de las técnicas de IA. Concretamente en las prácticas, los alumnos deben desarrollar un equipo de jugadores de fútbol que decidan las acciones a realizar basándose en varias de las técnicas de IA. Al final del curso, se realiza una competición entre los diferentes equipos, donde los estudiantes pueden evaluar y comparar objetivamente los resultados. La motivación que adquieren al competir entre ellos en partidos reales es clave para despertar su interés hacia la IA. Esta metodología se está probando desde hace cinco años en la Universidad de Girona (UdG) y los resultados nos avalan.

1. Introducción

La mayoría de los planes de estudios de informática incluyen temas de inteligencia artificial (AI), tal y como recomiendan los planes de estudios en informática de ACM/IEEE-CS [7]. Y a nuestro parecer, en las nuevas titulaciones que surjan como consecuencia de las reformas causadas por el nuevo espacio europeo de enseñanza, la IA, en cualquiera de sus diferentes formas (sistemas de soporte a la decisión, minería de datos, etc.), seguirá siendo una pieza clave en la formación de ingenieros informáticos.

Los cursos actuales de IA se basan en la enseñanza de una colección de técnicas de manera aislada. Por ejemplo, una clase teórica basada en la heurística es seguida de una de prácticas donde un ejercicio de juego es utilizado para consolidar los conocimientos. No se dedica ningún esfuerzo a integrar o comparar diversas técnicas, tema clave en una formación en ingeniería.

La práctica proporciona el camino para colocar a los alumnos ante un problema de ingeniería. Pero, las prácticas suelen estar dirigidas hacia técnicas específicas, de tal manera que los alumnos suelen terminar aprendiendo sólo el problema/técnica específico discutido. Así pues, el alumno no adquiere aptitudes hacia la integración o comparación de diferentes técnicas, un elemento crucial en la formación de un ingeniero.

La reciente incorporación de técnicas distribuidas, como los sistemas multiagente, en los temarios de IA, como consecuencia de su adaptación a los avances en la disciplina, han favorecido el desarrollo de prácticas de integración de técnicas. Sin embargo, esta integración aunque pueda resolver el problema de la complementariedad, no trata en la mayoría de las ocasiones la comparación de diferentes técnicas.

Esta debilidad en la enseñanza está siendo afrontada por algunos profesores mediante el uso de aplicaciones apropiadas que permiten la combinación de varias técnicas de IA para solucionar un problema a la vez que comparar los sistemas resultantes. Las más populares son las subastas [17] y el fútbol [4]. Este último ha recibido una especial atención desde la fundación de la RoboCup, puesto que ha fijado varios desafíos en la comunidad de la IA. El fútbol, así como otras competiciones de ámbito internacional (ver por ejemplo [19]), se están estableciendo como un marco de referencia para la comparación

de sistemas inteligentes complejos que difícilmente pueden ser evaluados de otra manera.

En esta ponencia mostramos la experiencia de cinco cursos en el uso de la simulación del fútbol en los cursos de IA de la Universidad de Girona (UdG). La construcción de un equipo de fútbol requiere de un gran número de habilidades que integren diversas capacidades para alcanzar un software inteligente y adaptativo, que interactúe continuamente con un mundo cambiante y no determinista [11]. Además, la simulación de fútbol presenta un marco competitivo en la que diferentes sistemas construidos por los alumnos pueden ser comparados a través de los resultados experimentales: ganar o perder un partido.

Desde el punto de vista docente, los resultados obtenidos mediante el uso de un entorno competitivo como es el fútbol, han sido muy satisfactorios. Los alumnos se interesan en la IA de tal manera que piden participar en nuestros proyectos de investigación para hacer su proyecto final de carrera o proyecto de tesis doctoral.

Esta ponencia está organizada de la siguiente manera. Primero describimos en la sección 2 la metodología de enseñanza de IA llevada a cabo en la UdG. A continuación, en la sección 3, proporcionamos las diversas prácticas desarrolladas y continuamos en la sección 4 discutiendo las diversas consecuencias de esta metodología después de cinco años de su aplicación. Finalmente, concluimos en la sección 5 con varias observaciones.

2. Enseñanza de la IA en la UdG

La Ingeniería Informática en la Universidad de Girona (UdG) es una titulación de segundo ciclo. Por ello, la mayor parte de los estudiantes que cursan estos estudios han obtenido con anterioridad la titulación de Ingeniero Técnico en Informática de Sistemas (ITIS) o de Gestión (ITIG). En estas titulaciones de primer ciclo, los estudiantes tienen acceso a una asignatura optativa sobre Inteligencia Artificial Aplicada (ITIS) o Introducción a la Inteligencia Artificial (ITIG). Atendiendo que existe una materia troncal en Ingeniería Informática sobre Inteligencia Artificial, se ha realizado un gran esfuerzo por parte del profesorado para evitar duplicar contenidos entre las asignaturas de las titulaciones técnicas y las superiores. En este sentido, las titulaciones técnicas se basan en el aprendizaje de herramientas de IA, mientras que en la titulación superior el énfasis se realiza en el desarrollo e implementación de técnicas. En este trabajo nos centraremos en la titulación superior.

El perfil de la formación de ingenieros informáticos en la UdG se orienta hacia informáticos industriales. Bajo esta perspectiva, los planes de estudios de IA se basan en dos cursos: el primero es troncal y se llama "Inteligencia Artificial: Técnicas y Métodos" (IATM) y el segundo es opcional y se llama "Diseño de Sistemas de Supervisión" (DSS). La primera asignatura hace más énfasis en los fundamentos de la IA, mientras que la segunda en las técnicas y métodos útiles para el desarrollo de sistemas inteligentes en entornos industriales. Consecuentemente, la primera asignatura cuenta con dos horas semanales de clases teóricas por una de prácticas, mientras que la segunda está más equilibrada y cuenta con una clase teórica semanal y una de prácticas.

Esta organización nos permite combinar la enseñanza teórica, que pone énfasis en el contenido que debe ser enseñado, con enseñanza práctica, que proporciona las aptitudes que deben ser adquiridas y los problemas a los que debe hacer frente un ingeniero. Ambas estrategias de docencia forman una metodología de enseñanza que consideramos conveniente para los ingenieros, coincidiendo con [2] y [12].

La peculiaridad de nuestra metodología es la organización de las prácticas. En lugar de consistir en ejercicios aislados, un problema para cada técnica, el fútbol proporciona un marco de trabajo común donde las diversas técnicas de IA se utilizan para solucionar el mismo problema. Además, el juego de fútbol tiene la ventaja de ser conocido y atractivo para los alumnos. De hecho, a los estudiantes les fascina poder implementar un equipo de jugadores que actúan sin la interacción humana y preparan a los jugadores virtuales como si fueran auténticos entrenadores profesionales.

2.1. Contenidos teóricos

Aunque algunos de los estudiantes llegan a los cursos de IA con algunos conocimientos adquiridos en las titulaciones técnicas, para la

mayoría de ellos la asignatura IATM significa su primer contacto con la disciplina. Por ello, las clases teóricas, donde el contenido de los temas es proporcionado a los estudiantes, resultan indispensables. Las clases teóricas tienen una organización secuencial del contenido y una estructura lógica del tema. El profesor posee el conocimiento y lo enseña mientras los alumnos permanecen pasivos. El feedback en las clases se proporciona en el examen al final de los cuatrimestres.

Respecto al contenido, el primer curso, IATM está pensado para proporcionar la base de IA, mientras que DSS, está concebido para poner en práctica los temas aprendidos en IATM al mismo tiempo que varias técnicas son repasadas con profundidad. Durante los cinco cursos que han servido de experiencia para este trabajo, los temarios han sido modificados susceptiblemente de acuerdo a la experiencia. La tabla 1 muestra el programa desarrollado para el curso actual, 2004-05.

Tabla 1. Materias de IATM y DSS

IATM	DSS
• Agentes racionales • Resolución de problemas • Juegos • Redes neuronales • Algoritmos genéticos • Sistemas Expertos • Sistemas difusos • Técnicas de aprendizaje automático • Sistemas multiagente	• Técnicas de Planificación • Técnicas de Scheduling • Minería de datos • Sistemas multiagente • Planificación basada en casos

Obsérvese que IATM hace énfasis en los aspectos fundamentales de la asignatura mientras que DSS en la aplicación de técnicas de IA dentro de la pirámide CIM (*Computer Information Manufacturing*, ver figura 1), de acuerdo con el perfil requerido en el plan de estudios. Asimismo es interesante observar que ambas asignaturas abordan el desarrollo de sistemas como agentes inteligentes que colaboran entre ellos para resolver problemas y/o están integrados con otras aplicaciones convencionales de IA en entornos abiertos. Precisamente la utilización del concepto de agente inteligente nos permite desarrollar nuestra estrategia docente sobre la integración y comparación de técnicas, como se verá en la sección siguiente.

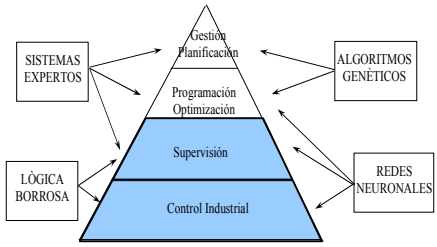


Figura 1. Técnicas de IA aplicables en las tareas que componen la pirámide CIM

2.2. Contenidos prácticos

La idea básica de las clases prácticas es enseñar a los alumnos a concretar problemas de la ingeniería que requieren el conocimiento de la disciplina, poniéndolos en la situación de los ingenieros [2]. El punto clave reside en la selección del problema. En nuestra tentativa de buscar un problema lo suficientemente rico, donde varias técnicas de la IA fueran aplicadas, hemos seleccionado el diseño de un equipo de fútbol. La idea de usar el fútbol como base del problema nace de la RoboCup. La RoboCup es un proyecto común internacional para promover la IA, la robótica y los campos relacionados. El objetivo básico es proporcionar un problema estándar donde varias tecnologías puedan ser integradas y examinadas [15],[6]. De acuerdo a [6], la construcción de un equipo de fútbol robótico necesita de los siguientes conocimientos y habilidades:

- Desarrollo de jugadores: agentes autónomos, robótica, visión, fusión de información de diferentes sensores en tiempo real.
- Trabajo en equipo: colaboración multiagente, reconocimiento de contextos
- Comprensión de la competición: modelización cognitiva
- Habilidad para desarrollar y ejecutar estrategias y juegos en tiempo real:

adquisición de estrategias, razonamiento en tiempo real, planificación y comportamiento reactivo
- Entrenamiento pre y post-juego: aprendizaje automático.

La RoboCup tiene diferentes ligas de competición:
- Simulación, donde se realizan partidos de fútbol simulado
- De robots de tamaño pequeño, también llamada liga F-180, donde compiten dos equipos de 5 jugadores robots cuyas dimensiones no sobrepasan 180 mm de diámetro por 15 cm de altura.
- De robots de tamaño mediano, donde compiten dos equipos de 6 jugadores, cada uno de ellos midiendo un máximo en superficie de 50x50 cm y 80 kg. de peso, incluyendo todos los sensores *on-board*.
- De robots con 4 patas, donde dos equipos de 4 robots tipo AIBO rivalizan.
- Humanoide, donde robots de diferentes tamaños (entre 18 cm y 2 m) compiten en el campo de fútbol. Ésta última liga está todavía en fase experimental.

Las dos primeras competiciones se realizan en entornos reales, con robots de diferentes tamaños. Para poder realizar prácticas sobre estas plataformas se requeriría de una infraestructura docente que, por ahora, resulta prohibitiva desde el punto de vista económico de nuestra universidad. Además, la utilización de plataformas de robots requiere de conocimientos de control de sistemas. Por ello, sí se han realizado algunas prácticas con robots en asignaturas especializadas y con pocos alumnos en las ingenierías industriales (véase [13]), pero no se ha considerado conveniente en las asignaturas de IA de los ingenieros informáticos.

En cuanto al entorno de simulación de Robocup, éste requiere de un servidor y terminales bastante complejos. Por ello, se ha optado por utilizar Javasoccer [8]. Javasoccer es un programa que simula el dinamismo y las dimensiones de la liga de robots pequeños de la RoboCup. Dos equipos de jugadores compiten en un campo empujando y chutando la pelota hacia la portería del equipo rival (véase la fig. 2).

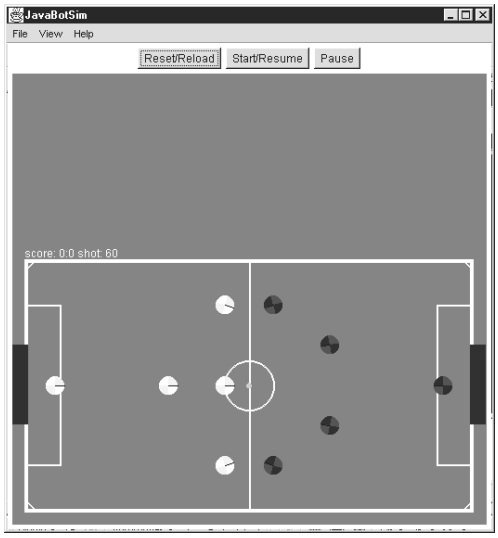


Figura 2. Interfaz gráfica de Javasoccer

El entorno es parametrizable en cuanto al número de jugadores por cada equipo, el tiempo de ejecución, las faltas permitidas, etc. Los alumnos sólo deben diseñar los jugadores del equipo a partir de un conjunto de bibliotecas predefinidas. Estas bibliotecas contienen las funciones básicas de los jugadores (sus movimientos en el campo,…) permitiendo así que los alumnos concentren sus esfuerzos de programación en el proceso de decisión de cada jugador. Las estrategias del equipo se pueden implementar con cualquier software, y con técnicas de IA que faciliten la manipulación de situaciones dinámicas como las que surgen en el fútbol.

El problema que los alumnos deben resolver es el de construir un equipo de Javasoccer que derrote a los otros equipos en un partido de fútbol usando diferentes técnicas de IA. Los alumnos organizados en parejas, trabajan el problema y seleccionan los métodos de IA que consideran más convenientes. El profesor supervisa su trabajo y se asegura que los alumnos aprendan diversas técnicas con profundidad. Finalmente, la competición permite a los alumnos comparar las diferentes técnicas aplicadas.

3. Prácticas basadas en la simulación de fútbol

Hemos desarrollado dos conjuntos de prácticas que corresponden a las dos asignaturas comentadas, IATM y DSS. El primero está pensado para poner en contacto a los alumnos con las técnicas de IA, mientras que el segundo sirve para consolidar su uso ya que los alumnos tienen adquirida la base.

Todas las prácticas están pensadas para realizarse en grupo de, como máximo dos estudiantes. De esta manera se favorece el trabajo en grupo, una de las características transversales que forman parte de la formación universitaria.

3.1. Toma de contacto

La toma de contacto de los estudiantes con la práctica de las técnicas de IA se organiza en dos partes. En la primera parte, las prácticas se relacionan con los problemas típicos de IA. Siendo éste el primer curso en IA, es importante que los alumnos conozcan los problemas canónicos de esta disciplina. Hay un total de cinco trabajos a realizar para superar el curso. Durante los cinco últimos cursos se han ido variando los enunciados. Un resumen de ellos se muestra en la tabla 2.

Tabla 2. Prácticas de IATM

1. Introducción al Prolog Desarrollo de un árbol genealógico Desarrollo de una calculadora de expresiones boleanas Implementación de un juego de aventuras de tipo laberinto.
2. Resolución de problemas y búsqueda heurística El mundo de los bloques El problema de las tilas.
3. Juegos Nine-men Morris Othello
4. Sistemas basados en las reglas Diagnóstico de una avería Control de temperatura
5. Aprendizaje automático Implementación de ID3.

Los lenguajes utilizados son, además de Prolog, C++ o Java. En cuanto a los entornos de desarrollo de sistemas expertos y difusos, se viene utilizando durante muchos cursos JESS [9] y FuzzyJess [5].

Es en la segunda parte de este conjunto de prácticas cuando se introduce la plataforma de fútbol simulado. Se distinguen dos prácticas:

1. Diseño de un jugador de fútbol. El alumno debe definir el problema de tomar una decisión sobre cómo debe de actuar el jugador de acuerdo a la información conocida. El método de toma decisiones se debe de basar en lógica borrosa, esto es, con la utilización de FuzzyJess [5]. Esto conlleva que el estudiante deba practicar con la integración de Javasoccer y FuzzyJess. Por un lado, Javassoccer proporciona un entorno multiagente, en que cada jugador toma una decisión individual. Por otro lado, FuzzyJess proporciona los mecanismos de decisión difusos. La información proporcionada por Javassocer es: posición de la pelota, posición del jugador, posición de los compañeros, posición de los contrarios, etc. Ésta información es numérica y debe de ser interpretada por los estudiantes para definir los correspondientes conjuntos difusos (pelota cerca, compañero lejos, etc.) sobre los cuales desarrollar las reglas de toma de decisión. Las acciones posibles a realizar por cada jugador son libres.
2. Diseño e implementación de mecanismo de coordinación para un equipo de fútbol. En esta práctica se trata de utilizar la cooperación entre los jugadores, mediante la comunicación de las jugadas a realizar de manera que se puedan coordinar los equipos. Por ejemplo, si un jugador decide pasar la pelota a un compañero, éste último tiene que coordinarse para que la recepción de la pelota se lleve a cabo satisfactoriamente.

Al final del cuatrimestre, hay una sesión especial de prácticas donde los equipos compiten en un campeonato. Ya que todo alumno debe implementar el mismo conjunto técnicas, la competición de fútbol muestra la capacidad de cada equipo para combinar las técnicas, la exactitud de su implementación y su capacidad de proporcionar al sistema el conocimiento apropiado.

El profesor no necesita hacer nada especial para motivar a los alumnos, dado que el desafío de ser el mejor equipo los estimula más de lo esperado.

3.2. Consolidación de conocimientos

En el curso de DSS, todas las prácticas tienen dos partes: en la primera se aplica la técnica a un ejercicio de supervisión y en la segunda se aplica al fútbol. Las prácticas realizadas durante el último curso se muestran en la tabla 3.

Tabla 3. Prácticas de DSS

1. Revisión de los conceptos básicos de resolución de problemas mediante heurísticas 1.1 Exploración en dominios clásicos 1.2 Utilización en el proceso de decisión de un jugador
2. Planificación 2.1 Planificación basada en casos para resolver un problema de ensamblaje de piezas 2.2 Planificación de la actividad de un jugador
3. Scheduling 3.1 Scheduling de las prácticas de un estudiante durante un curso escolar 3.2 Scheduling de la actividad de un jugador
4. Sistemas multiagente 4.1 Coordinación de un conjunto de robots en el mundo de la basura. 4.2 Coordinación de un conjunto de jugadores para ganar un partido de fútbol.
5. Minería de datos 5.1 Aprendizaje de árboles de decisión para el diagnóstico de fallos eléctricos 5.2 Aprendizaje *off-line* (fuera de partido) de las jugadas coordinadas de un equipo de jugadores de fútbol.

Cada una de las prácticas, en su aplicación al fútbol, se amplía respecto a la versión anterior. Por tanto es un proceso de mejora continuado:

- La planificación proporciona una estrategia de juego a largo término
- La coordinación multiagente procura una estructura de cooperación para la toma de decisiones en equipo
- Las técnicas de aprendizaje automático permiten el entrenamiento del equipo antes (*off-line*) y durante (*on-line)* un partido.

Cada grupo de estudiantes se especializa en un conjunto de técnicas de IA. Durante estos tres años las técnicas sugeridas e implementadas por los alumnos han sido las siguientes:

- Lógica difusa: implementación normal y neuronal
- Redes neuronales off-line y on-line
- Algoritmos genéticos: off-line y on-line
- Razonamiento basado en casos
- Planificación basada en casos
- Agentes que se comunican con comportamiento adaptativo

La última sesión se dedica a los partidos de fútbol, como en el curso anterior. La importancia de la competición en este curso permanece ya que los equipos se construyen con diversas técnicas, con más grado de complejidad y con más dedicación. La motivación añadida de competir contra los compañeros de clase hace que los resultados sean realmente fabulosos.

3.3. Resultados

Según la organización del curso, podemos hablar sobre resultados en las clases teóricas y en las prácticas. Con respecto a las clases teóricas, debemos observar que todos los alumnos que han realizado los trabajos suelen superar los exámenes. Esto es una indicación de que consolidaron el contenido de los cursos. Después, los alumnos adquieren las aptitudes para evaluar y seleccionar técnicas a través de las prácticas. Las prácticas sirven como base para entender los recursos y la complejidad para implementar las técnicas de IA. La asimilación de estos conocimientos permite a los alumnos poseer una mayor objetividad para enfrentarse a las aplicaciones del mundo real.

Uno de los resultados más interesantes se obtiene en la última clase dedicada a la competición. Primero porque el día de la competición es una fecha límite para los alumnos, como cualquier plazo de un proyecto de ingeniería en el mundo real. Y segundo, porque el equipo debe funcionar en un entorno real, es decir, no entregan una demo. Además, el equipo debe jugar el partido contra otro que no ha visto previamente. Esto conduce a un cierto grado de imprevisión que los alumnos deben tener en cuenta en el proceso de desarrollo de su equipo de jugadores.

Sin embargo, no se puede confundir el hecho de ganar un partido con el de que la técnica utilizada sea la mejor. Esta es otra lección que los

alumnos aprenden: una buena técnica no funciona necesariamente bien si no se la implementa apropiadamente. Por ejemplo, un alumno puede construir un sistema que razona basado en casos para cada jugador, y después descubrir que el sistema no es suficientemente bueno, si la accesibilidad a la base de casos no es rápida. La técnica depende de la implementación concreta i la ocasión particular en que se usa.

4. Impacto de la metodología

En esta sección discutimos algunos de los impactos de la metodología aplicada en los cursos de IA de ingeniería en el dominio académico, de la investigación y de la transferencia tecnológica.

4.1. Impacto académico

Primero debemos hablar sobre la reacción de los alumnos. Su grado de participación ha ido más allá de nuestras expectativas. El fútbol es una aplicación clave para su motivación, puesto que convierte la rivalidad deportiva en interés científico-técnico. El ser el ganador de los partidos hace que los alumnos inviertan más horas en el desarrollo de su equipo que el tiempo planeado por los profesores, provocando un mayor aprendizaje. Además, los alumnos valoran positivamente el conocimiento aprendido; concretamente, aprecian el esfuerzo realizado en la integración de varias técnicas en un solo sistema. Estos resultados han sido experimentados ya por otras universidades, como por ejemplo [3].

También podemos mencionar que ha habido un interés creciente de los estudiantes en la realización del proyecto fin de carrera en IA.

Finalmente, hay que considerar que los programas de IA desarrollados a partir de la simulación del fútbol, tienen unos buenos resultados como método de la enseñanza. Es importante observar que, para tratar el problema de proporcionar habilidades a los alumnos de informática, algunas metodologías educativas caen en enseñanzas obsoletas como el a veces llamado aprender por hacer. Esta propuesta puede provocar que el alumno acabe su enseñanza con un buen conocimiento sobre una técnica específica pero también con varias lagunas en su conocimiento general. Sin embargo, nuestra metodología a combinar la enseñanza teórica con las prácticas, siguiendo los resultados acreditados por [2] y [12], previene tales desventajas. El uso de la simulación del fútbol, un problema suficientemente complejo para integrar varias técnicas de IA [10], refuerza una metodología para aprender de forma integrada. Que los resultados actuales en la investigación del fútbol se estén aplicando a otros problemas complejos del mundo real, a la organización del trabajo en equipo [14] y al rescate [16], prueba la validez de la aplicación como plataforma para la enseñanza a ingenieros informáticos.

4.2. Impacto en la investigación

El fútbol ha sido el marco de trabajo en la investigación del grupo *Agents Research Lab* dentro del Instituto de Informática y Aplicaciones de la UdG desde 1995 (véase [18]). Podemos constatar que la metodología de la enseñanza explicada, ha acercado a muchos estudiantes a nuestro laboratorio de investigación para conocer cuáles son los proyectos que se están llevando a cabo relacionados con la IA. Por ello, la utilización del simulador de fútbol nos proporciona una muy buena presentación para poder incorporar a nuestro grupo de investigación estudiantes cualificados en la disciplina objeto de nuestro estudio.

4.3. Impacto en la transferencia de conocimiento

El fútbol también ha proporcionado un prestigio a la universidad para el público en general, gracias a la publicidad hecha al participar en las competiciones y eventos científicos importantes. Como consecuencia, las industrias más cercanas a nuestro entorno se han acercado para proponer proyectos de colaboración y para contratar alumnos. Todo este proceso ha sido consolidado gracias a la fundación de una *spin-off* llamada *Agents Inside* (véase [1]).

5. Conclusiones

La metodología de prácticas basada en la simulación del fútbol proporciona una formación *holística* en la disciplina de la IA. El fútbol nos proporciona el marco de trabajo ideal para la

integración y comparación de varias técnicas de IA, que es clave para los ingenieros a fin de facilitarles la selección de la técnica apropiada en un problema real. La metodología expuesta se ha utilizado durante cinco cursos consecutivos en la universidad de Girona, experiencia con la que hemos obtenido resultados académicos exitosos, así como impactos positivos en los dominios de la investigación y de transferencia de tecnología.

Los resultados obtenidos nos han animado a iniciar otros métodos de prácticas en entornos competitivos como los juegos. En particular, en los dos últimos cursos hemos realizado competiciones con Othello y Nine-Men Morris. Sin embargo, los problemas de juegos no nos proporcionan tantas facilidades para la integración de diferentes técnicas, como el caso de la simulación de fútbol. Otra posibilidad que estamos estudiando es establecer un campeonato inter-universitario de fútbol simulado.

Finalmente, pensamos que la simulación de fútbol nos puede permitir profundizar en las habilidades de trabajo de equipo de los estudiantes, constituyendo grupos de prácticas muti-disciplinares, en los que estudiantes de ingeniería industrial colaboren con estudiantes de ingeniería informática. Esperamos implementar esta estrategia docente en el próximo curso.

Agradecimientos

Queremos expresar nuestro agradecimiento a todos los profesores que nos han ayudado en los laboratorios de las prácticas de fútbol, especialmente Bianca Innocenti, Silvana Aciar, Eduard Muntaner, Josefina López-Herrera, Josep Antoni Ramon, Israel Muñoz y todos los miembros del Rogiteam.

Referencias

[1] Agents Inside, http://www.agentsinside.com/.

[2] Azevedo da Silveira, M., Scavarda-do-Carmo, L.C. "Sequential and Concurrent Teaching: Structuring Hands-On Methodology", IEEE Transactions on Education, vol. 42, no. 2, pp. 103-108, 1999.

[3] Coradeschi, S. Malec, J. "The use of RoboCup (soccer simulation) for an IA programming course", Journal of Robotic Society of Japan, Special issue on Robotics & Education, 1998.

[4] de la Rosa, JL., Montaner, M. "Docència de tècniques d'IA mitjançant Javasoccer", 2n Congrés Català d'Intel.ligència Artificial, Girona, Spain, 1999.

[5] FuzzyJess.http://www.iit.nrc.ca/IR_public/fuzzy/fuzzyJToolkit.html.

[6] Hedberg, S. "Robots playing soccer? RoboCup poses a new set of IA research challenges", IEEE Expert, September/ October, 1997, pp. 5-9.

[7] IEEE-ACM. Computing Curricula. Volume II: Computer Science. The Joint Task Force on Computing Curricula, IEEE computer Society, Association for Computing Machinery, 2001.

[8] Javasoccer, http://www.cc.gatech.edu/grads/b/Tucker.Balch/JavaBots/JavaSoccer/docs/index.html

[9] JESS, Java Expert System Shell, http://herzberg.ca.sandia.gov/jess/.

[10] Kitano, H., Suzuki, S., Akita, J. "RoboCup Jr.: RoboCup for Edutainment", Proc. of Int. Conf. On Robotics and Automation 2000, IEEE Press, NY, 2000.

[11] Kitano, H., Tambe, M., Stone, P., Veloso, M., Coradeschi, S., Osawa, E., Matsubara, H., Noda, I., Asada, M.. "The RoboCup Synthetic Agent Challenge 97".

[12] Lund, H.H. "Robot Soccer in Education", Advanced Robotics Journal, 1999.

[13] Luo, N., de la Rosa, J.L., Muñoz, I. "Micro mobile robots for advanced control course teaching". WFEO/ASEE e-Conference on Engineering Education, 2003.

[14] Raines, T., Tambe, M., Marsella, S. "Automated Assistants to Aid Humans in Understanding Team Behaviours", Proc. Autonomous Agents, Barcelona, Spain, 2000.

[15] RoboCup, http://www.RoboCup.org/.

[16] RoboCup-Rescue, http://robomec.cs.kobe-u.ac.jp/robocup-rescue/.

[17] Rodriguez-Aguilar, J.A. Martin, F.J., Noriega, P., Garcia, P., Sierra, C. "Towards a Test-bed for Trading Agents in Electronic Auction Markets", April, 1998, IA Communications Journal. CO;PROVAR

[18] RogiTeam experimental platform, Agents Research lab, http://eia.udg.es/arl/research.html.

[19] Trade Agents competition, IJCAI 2005, http://www.sics.se/tac/.

La Accesibilidad: materia obligatoria en los planes de estudio de Ingeniería Informática

Julia González
Departamento de Informática.
Escuela Politécnica
Universidad de Extremadura.
10071 Cáceres
juliagon@unex.es

Mercedes Macías
Departamento de Informática.
Facultad de Ciencias del Deporte
Universidad de Extremadura.
10071 Cáceres
mmaciasg@unex.es

Fernando Sánchez
Departamento de Informática.
Escuela Politécnica
Universidad de Extremadura.
10071 Cáceres
fernando@unex.es

Resumen

La disponibilidad y la inmediata difusión de la información que proporciona la Web es la mejor cualidad que ofrece Internet a sus usuarios. Estas características de la red se han revelado especialmente valiosas en el caso de las personas que presentan ciertas discapacidades, ya que han visto favorecido su acceso a la educación y al mercado laboral. Pero la disponibilidad de la información no implica necesariamente su accesibilidad y en ocasiones es casi imposible poder obtener dicha información.

La accesibilidad de la información y de las nuevas tecnologías es una característica deseable que los sistemas deberían tener para evitar las barreras y afortunadamente, en los últimos años se están desarrollando legislaciones que apoyan una sociedad de la información sin exclusiones evitando en lo posible la existencia de barreras, incluidas las digitales.

En este trabajo, se verá cuál es el marco normativo en el que se está desarrollando la sociedad de la información actual, y además se reflexionará sobre el problema de la *accesibilidad* a la información publicada en la Web. Con este propósito, recorreremos las diferentes barreras que suelen encontrar las personas con discapacidad a la hora de alcanzar la información existente en la red. Podemos decir que el tratamiento de la accesibilidad a la web es un concepto novedoso, pero eso no debe impedir que los profesionales de las nuevas tecnologías posean conocimientos que permitan evitar o paliar las posibles barreras, y conozcan los fundamentos del "diseño para todos".

El nuevo Espacio Europeo de Educación Superior nos proporciona el marco ideal para modificar nuestros planes de estudio e incluir asignaturas o itinerarios relacionados con este tópico en el nuevo título de grado de Ingeniería Informática, y por tanto, hacer que este concepto esté presente en todo el proceso de diseño de los futuros sistemas informáticos.

1. Introducción

Con el auge de las nuevas formas de comunicación y de la disponibilidad de la información a través de Internet, las personas con discapacidades se han visto enormemente respaldadas. Así, un individuo con discapacidad motriz, es capaz de escribir un libro mediante el uso de dispositivos de reconocimiento de voz y publicarlo posteriormente en la red. Un sujeto con discapacidad visual podrá obtener los contenidos de ese libro mediante un sintetizador de voz al tiempo que otro con discapacidad visual y auditiva accederá al mismo con un dispositivo Braille.

El número total de personas con discapacidades permanentes en España es de 3'5 millones de personas, lo que supone un 9% de la población [1]. Las discapacidades pueden ser muy variadas: visión, oído, comunicación, aprendizaje, utilización de brazos o manos, etc. Y de diverso grado, por ejemplo en el caso de la visión, puede ir desde la discapacidad para recibir cualquier imagen hasta la discapacidad para tareas visuales de detalle. Pero a esta cantidad deben sumárseles 4'4 millones de personas con edad avanzada que ven sus capacidades motrices y sensoriales disminuidas y también a aquellos que por circunstancias transitorias, tienen sus capacidades limitadas. Se puede cifrar este grupo en 8'1 millones de personas. Se tiene pues en total unos

16 millones de personas [1], que son beneficiarios potenciales de los aportes proporcionados por la accesibilidad en general. Hablamos de aproximadamente el 40% de la población total, lo que nos obliga a tomar las medidas adecuadas para no discriminar a un grupo de población tan extenso.

Las nuevas tecnologías supusieron y siguen suponiendo un gran avance en nuestra sociedad y un medio para la integración social y laboral de las personas con discapacidad. Permiten reducir considerablemente y en muchos casos eliminar las minusvalías, pero lamentablemente la nueva sociedad de la información también se ha convertido en una causa de exclusión, se ha abierto lo que ha dado en denominarse *brecha digital*. La falta de formación en nuevas tecnologías, la inaccesibilidad al hardware o al software son algunos de los motivos de la aparición de estas barreras.

En lo relativo a la disponibilidad de la información publicada en Internet, es evidente que no todo el mundo puede acceder a ella en la misma medida o de la misma forma. Las personas que presentan alguna discapacidad, en muchos casos deben franquear un sinfín de obstáculos de accesibilidad de distinto tipo, y todo ello para conseguir situarse en el mismo punto que el resto de usuarios.

Puesto que los problemas de acceso a la información no son exclusivos de los afectados, la Administración ha comenzado a tomar algunas medidas al respecto. De aquellas relativas a la accesibilidad a la Web se hará un repaso en la siguiente sección. En cuanto a las diferentes barreras de la accesibilidad con que se encuentran los usuarios con algún tipo de discapacidad se hablará en la sección 3, profundizándose en el problema concreto de la accesibilidad a la Web en la sección 4. Todo ello en correspondencia con las diversas áreas de conocimiento y materias de estudio que pueden verse implicadas en las distintas titulaciones universitarias en Informática. Finalmente se reunirán una serie de reflexiones a modo de conclusión.

2. Marco normativo para el acceso global en la sociedad de la sociedad de la información

Actualmente estamos inmersos en la Sociedad de la Información. Los procesos más habituales se desarrollan utilizando como soporte Internet y nuevas tecnologías, pero este avance genera, en ocasiones, barreras insalvables para determinados grupos de población, debidas a una falta de conocimiento o de previsión en el diseño y desarrollo de estos procesos. Por todo esto, es necesario que existan leyes y normativas que aseguren que el avance tecnológico no limite la accesibilidad a dichos procesos o a la propia información.

Como primera medida, en Diciembre de 1999 la Comisión Europea crea la iniciativa *eEurope: una Sociedad de la Información para todos* [2], para dar soporte a la entrada de Europa en la era digital, acelerando su desarrollo y estimulando la creación de nuevos servicios y actividades económicas, con el fin de promocionar la competitividad y la creación de empleo. Entre sus objetivos se cuentan:

- Conseguir que todos los ciudadanos, hogares, escuelas, empresas y administraciones estén conectados a la red.
- Crear en Europa una cultura y un espíritu empresarial abierto a la cultura digital invirtiendo en las personas y en la formación.
- Garantizar que la sociedad de la información no se traduzca en exclusión social.

Para lograr estos objetivos se consideraron una serie de áreas prioritarias [3]. De todas ellas, concretamente la número siete hacía referencia a *la participación de los discapacitados en la cultura electrónica* [4]. Las nuevas tecnologías ofrecen oportunidades a las personas con discapacidad para superar los obstáculos socioeconómicos, geográficos y culturales con que se enfrentan, permitiéndoles participar en la vida social y laboral en igualdad de posibilidades.

Esta iniciativa pretendía incentivar a la industria europea para que explotase plenamente el potencial de mercado de los productos y servicios concebidos para discapacidades que con frecuencia se pueden desarrollar con escasos costes adicionales, utilizando el principio del *diseño para todos* o diseño universal, que implica tener en cuenta las necesidades específicas desde el proceso de diseño. También se dedica un especial esfuerzo a la mejora de las oportunidades en educación y formación y a garantizar la plena

participación de los individuos con discapacidad en la sociedad.

El Gobierno español, lanzó a principios de Enero de 2.001 el Plan de acción INFOXXI [5] para llevar a cabos las acciones propuestas en la iniciativa eEurope. Una de las acciones de este plan, la cuarta, habla de *la accesibilidad y alfabetización digital* y pretende *facilitar el acceso a la Sociedad de la Información y el uso intensivo de las nuevas tecnologías a los discapacitados con el fin de conseguir la igualdad de oportunidades*.

Trascurridos los plazos de acción propuestos en la iniciativa comunitaria y en el plan INFOXXI, aparece un nuevo plan de acción europeo: eEurope 2005: Una sociedad de la información para todos [6]. Este plan nace como sucesor del plan eEurope 2002, y busca estimular el desarrollo de servicios, aplicaciones y contenidos en Internet, acelerando al mismo tiempo el despliegue de un acceso seguro.

El nuevo plan, aprobado por el Consejo Europeo de Sevilla en junio de 2002, sucede al Plan 2002 y con él se pretende extender la conectividad a Internet en Europa. Para ello, se plantea la consecución de una infraestructura de banda ancha segura y disponible para la mayoría, que se traduzca en un aumento de la productividad económica y una mejora de la calidad y la accesibilidad de los servicios en favor del conjunto de los ciudadanos europeos. Las líneas de acción prioritarias están encaminadas a conseguir:

- servicios públicos en línea modernos
- negocios electrónicos dinámicos
- una infraestructura de información segura
- disponibilidad masiva de un acceso de banda ancha a precios competitivos
- evaluación comparativa y difusión de las buenas prácticas

Existe además un objetivo transversal, general para todo el plan, de acceso para todos, con el fin de luchar contra la exclusión social, ya sea vinculada a necesidades especiales, a una minusvalía, a la edad o a la enfermedad.

Dentro de las acciones propuestas en este plan está la de mejorar el acceso de las personas con discapacidad a los sitios Web públicos, por ello en el DOC 86 DEL 10.04.2002, se declara que los Estados miembros deben acelerar la implantación de las directrices de Accesibilidad para todas sus páginas Web.

La adopción del plan por los estados miembros lleva aparejada la creación de leyes y planes de actuación para su consecución. En España, la primera ley que incluye los principios de este plan es la Ley de Igualdad de Oportunidades, no Discriminación y Accesibilidad Universal de las Personas con Discapacidad, LIONDAU [7]. Esta ley, actualiza los contenidos dedicados a las personas con minusvalía, hasta entonces recogidos en la Ley de Integración Social de los Minusválidos, LISM [8]. Y establece el marco legal para tratar la accesibilidad en general. Un año antes, nace la Ley de Servicios de Sociedad de la Información y de Comercio Electrónico, [9], donde se van a tratar temas concretos de la Accesibilidad en las TICs.

En la disposición final séptima de la LIONDAU, se establece que antes de diciembre de 2005 el Gobierno debía aprobar unas condiciones básicas de accesibilidad y no discriminación para el acceso y utilización de las tecnologías, productos y servicios relacionados con la sociedad de la información. Estas condiciones serán obligatorias entre el 2007 y el 2009 para todos los productos y servicios nuevos, y entre el 2011 y 3l 2013 para todos aquellos productos existentes que sean susceptibles de ser ajustados razonablemente. Una de las disposiciones más importantes, como personal docente de una Universidad Española, es la disposición final décima, en la que se declara que antes del fin de 2005, en todos los programas educativos, incluidos los universitarios, el Gobierno desarrollará el currículo formativo en "diseño para todos".

Cabe destacar la disposición adicional quinta de la LSSI, en la que se establece de forma explícita que antes del 31 de diciembre de 2005 los sitios Web de la administración pública deberán ser accesibles con los criterios de accesibilidad generalmente reconocidos, para personas con discapacidad o bien personas de edad avanzada. También se establece que las administraciones podrán requerir estos mismos criterios de accesibilidad a aquellos sitios web que financien.

Con el objetivo de realizar las acciones necesarias para que se lleven a cabo las propuestas

de las leyes anteriores, se ponen en marcha los planes de acción siguientes:

Para la accesibilidad en general, dependientes del Ministerio de Trabajo y Asuntos Sociales:

1. *II Plan de acción para las personas con discapacidad 2003-2007* [10]. Las estrategias más importantes de este plan son:
    - La atención a personas con graves discapacidades.
    - Las políticas activas de inserción laboral de las personas con discapacidad.
    - La promoción de la accesibilidad de entornos, productos y servicios. Dentro de esta área se incluye la creación de master y asignaturas que formen a los profesionales en el "diseño para todos". Lamentablemente, se recogen estas acciones para Arquitectos, Ingenieros de Caminos e Ingenieros en Telecomunicaciones, excluyéndose los Ingenieros en Informática.
    - La cohesión de los Servicios Sociales para personas con discapacidad.
2. *I Plan Nacional de Accesibilidad 2004-2007* [11]. Uno de sus objetivos, el quinto, es la promoción de la accesibilidad en las nuevas tecnologías. Dentro de la línea de actuación de *Concienciación y formación,* se establece como estrategia la inserción del "diseño para todos" en los estudios universitarios. Para ello se financia la creación de asignaturas específicas, la asistencia a congresos, la cofinanciación de cursillos y actividades sobre aspectos técnicos del diseño accesible y el desarrollo de material didáctico y técnico.

Para la accesibilidad en TICs, dependientes del Ministerio de Industria, Turismo y Comercio y del Ministerio de Administraciones Públicas:

1. *Plan de choque para el impulso de la administración electrónica en España [12].* A través de la medida 7 de este plan, la administración general del estado se compromete a que todos los sitios Web de la Administración, cumplan con los requisitos establecidos por la WAI del W3C[13].
2. *España.es* [14]. Dentro de este plan existen diferentes actuaciones, todas centradas en fomentar la accesibilidad de los discapacitados a sitios Web a través de la formación, estudios de la situación actual de la ley o el impulso de la adopción de estándares para la accesibilidad.

Desde el año 1999 desde las diferentes administraciones han existido intentos de transmitir a los ciudadanos la necesidad de la accesibilidad global, incluyendo la sociedad de la información. Sin embargo, en ninguna de las sucesivas directrices y leyes que han ido apareciendo a lo largo de estos años, aparece una descripción clara de lo que debe ser la accesibilidad a las nuevas tecnologías. Rara vez se trata este tema en profundidad, y aunque se ha querido potenciar la inclusión del "diseño para todos" en el currículo profesional a través de las Universidades, no encontramos que estas iniciativas hayan penetrado suficientemente.

Veamos en el siguiente apartado diferentes aspectos de la accesibilidad, y su posible estudio en el plan de estudio de los futuros profesionales de las Nuevas Tecnologías, los Ingenieros en Informática.

3. La accesibilidad

El principio de la Accesibilidad se basa en la filosofía del *diseño para todos*. La idea general es que los productos deberían ser usables por el rango más grande de población posible. Esto beneficiaría tanto al individuo con alguna discapacidad como al que no la tiene.

A continuación analizaremos los distintos niveles de accesibilidad tanto hardware como software, que un usuario con discapacidad debe salvar *antes* de alcanzar la información que pudiera estar publicada en la Web. En general las soluciones hardware son menos asequibles económicamente que las software, debido a la construcción de series limitadas por parte de empresas de poca expansión mundial.

3.1. Accesibilidad al hardware

Es fácil imaginar soluciones a problemas concretos de discapacidad, Sin embargo, es evidente que no se puede fabricar un hardware universal que sirva para todo tipo de discapacidad. En ocasiones se hace necesario utilizar ayudas técnicas adicionales que permitan al usuario utilizar un ordenador de forma no convencional.

Existen diversos dispositivos de Entrada / Salida específicos [15] como por ejemplo:

- Conmutadores, que permiten la selección por barrido. La persona activa la exposición de distintas opciones y escoge con un pulsador la opción que desea.
- Teclados Braille, que constan únicamente de seis u ocho teclas, cada una de ellas se corresponde con uno de los puntos que configura un carácter en Braille.
- Dispositivos sensibles a los cambios de potencial eléctrico muscular. Pueden activarse con mínimos cambios en la posición de los labios, de la mandíbula e incluso oculares.
- Dispositivos de entrada por voz.
- Terminales Braille, que permiten enviar la información de la pantalla hasta el usuario utilizando caracteres Braille dispuestos en una línea anexa al teclado, desde 20 hasta 80 ocurrencias según modelo. Se trata del único método de acceso al ordenador para personas sordociegas, unas 2.000 en España.
- Impresoras Braille. Las hay de una o dos caras y con posibilidad de realizar gráficos en relieve.

3.2. Accesibilidad al software

Para que un usuario con una discapacidad específica pueda utilizar el software instalado en su ordenador, puede disponer de una serie de ayudas software específicas, como pueden ser:

- Magnificadores de pantalla, que amplían la información a visualizar en un monitor. Algunos magnificadores permiten por ejemplo, ampliar únicamente una zona de la pantalla a modo de lupa.
- Lectores de pantalla. En este caso la información llega al usuario mediante mensajes hablados con voces total o parcialmente sintetizadas
- Reconocimiento de textos impresos OCR (Optical Character Recognition) parlante. Se trata de un método para que las personas con discapacidad visual puedan acceder a la información escrita en papel. La herramienta escanea el texto a leer, lo pasa del formato gráfico al formato texto y lo lee en voz alta.

Consideremos el caso de una persona con problemas de desplazamiento. Aunque puede ayudarse de una silla de ruedas y manejarla con destreza, en ocasiones las ciudades no resultan accesibles y presentan obstáculos insalvables que la silla no podrá franquear.

De igual modo, existen numerosos usuarios con discapacidades de cualquier tipo que son perfectamente capaces de manejar una aplicación informática y que disponen de las ayudas técnicas precisas. Pero, en ocasiones, el software no está diseñado del modo adecuado, resultando inaccesible en sí mismo.

Gracias a las demandas de algunas instituciones y de determinadas organizaciones, que exigen a los fabricantes de software el desarrollo de aplicaciones más accesibles, muchos de ellos empiezan a incluir algunas características de accesibilidad en los mismos. Por ejemplo, el Sistema Windows en las últimas versiones ofrece una opción en el Panel de Control para instalar y personalizar ciertas alternativas de accesibilidad. Al permitir cambios de tamaños de tipografía o de contraste de colores, los usuarios con resto visual pueden verse beneficiados y según el caso puede que no requieran software específico adicional para utilizar el ordenador. Sin embargo, estas facilidades para configurar el entorno aún son muy escasas.

3.3. El World Wide Web

La accesibilidad de un usuario a una página o sitio Web es la capacidad del mismo para conseguir el objetivo con que el autor ha desarrollado dicha página o sitio Web [16] generalmente el intercambio de información.

El avance de las nuevas formas de telecomunicación y en concreto de Internet, ha hecho posible que un usuario pueda conectarse con cualquier punto del planeta en cualquier instante. Así, un bien tan preciado como la información, está disponible en cualquier lugar del mundo en todo momento. Pero, que la información esté disponible no significa que sea accesible para todos. A veces, descifrar la información publicada en Internet es tan complicado que obliga a desistir al interesado de su empeño. Un usuario debe enfrentarse al navegador y a la propia página Web, si desea acceder a los contenidos, analicémoslos:

3.3.1. Accesibilidad de los navegadores

En el mercado actual existen diversos navegadores para acceder a Internet, aunque son sólo unos pocos los más utilizados, y en general no los más adecuados para ciertas deficiencias.

Algunos de los más utilizados, en sus últimas versiones se han visto forzados por los usuarios a incorporar opciones de accesibilidad que permiten configurar algunas de sus características de visualización. Sin embargo, dado el carácter eminentemente gráfico que la mayoría de navegadores ofrece, a veces no pueden ser utilizados ni manipulando estas opciones, por lo que interesan navegadores especializados, como aquellos que sólo muestran el texto de las páginas visitadas, o los que ofrecen los contenidos interpretándolos y produciendo una salida con inflexiones de voz.

3.3.2. Accesibilidad de las páginas Web

El autor de una página Web debería generar el contenido del mismo, pero no la presentación. De ésta última se encargarán el navegador y las opciones marcadas por el usuario. El usuario puede configurar el navegador escogiendo tipografías, tamaños o colores, puede usar un navegador no gráfico, con lo que obtendrá las descripciones en vez de las imágenes, puede escuchar el documento Web con un sintetizador de voz o leerlo con un dispositivo Braille. En todos los casos, la estructura del documento es la misma, pero la entrega o recepción la debería personalizar el usuario [18].

Cada vez es más frecuente la utilización de elementos no textuales en las páginas Web. Se incluyen imágenes, sonidos e incluso sensaciones táctiles. Naturalmente, no todos los usuarios están en disposición de poder captar todos estos medios de presentación, con lo que en ocasiones, los contenidos pueden resultar inaccesibles. Por ejemplo, si para incluir una cabecera atractiva o el directorio de la página Web se utilizan textos en formato gráfico, un lector de pantalla no encontrará texto que verbalizar y la persona con discapacidad visual no sabrá qué hay en la página.

Ante la enorme diversidad de páginas Web y multitud de diseños, la sección Web Accessibility Initiative (WAI) del World Wide Web Consortium (W3C) ha recopilado una serie de pautas de accesibilidad [13], que se han convertido en las guías a seguir, incluyendo a la propia legislación española, y ha elaborado un conjunto de guías destinadas a los creadores de páginas Web y a los desarrolladores de herramientas de autor [19]. Su objetivo final es que las páginas Web creadas sean más accesibles. Pero aunque estas guías constituyen un excelente recurso que todo autor debiera conocer, lo cierto es que no se puede obligar a nadie a seguir ninguna directriz al diseñar sus páginas.

Por otra parte, la sección WAI ha creado un grupo de trabajo denominado Evaluación y Reparación encargado de recopilar información sobre herramientas de evaluación, reparación, filtrado y transformación de páginas Web [20] para conseguir que entre creadores y usuarios sea posible que la Web sea más accesible. Algunas de estas herramientas están destinadas a los diseñadores, como las de evaluación y reparación facilitándoles el proceso de creación al detectar y corregir los posibles fallos de accesibilidad existentes. Otras están destinadas a los usuarios y su objetivo es dejarles participar de forma activa en el proceso de filtrado o transformación de las páginas que visitan permitiéndoles escoger, de forma limitada, qué elementos de la página desean obtener y de qué forma quieren percibirlos.

4. Perfil del Ingeniero en Informática

El perfil profesional del Ingeniero en Informática le hace protagonista de la sociedad de la información, y por tanto necesita que en su currículo formativo aparezcan las claves que han sido comentadas anteriormente. Un futuro profesional de las nuevas tecnologías necesita incorporar el concepto de la accesibilidad de manera global. Por ello en el plan de estudios, todos y cada uno de los subapartados tratados en el punto anterior deben ser considerados e incorporados al currículo.

En el libro blanco de la titulación de Ingeniero en Informática [21], adaptada al nuevo Espacio Europeo de Educación Superior, EEES, se ha estudiado la necesidad de tratar tres perfiles profesionales, pero con un único título de grado de carácter generalista, en el que se propone que en los contenidos específicos del título se traten contenidos agrupados en cuatro categorías:

- Programación
- Ingeniería del Software, Sistemas de Información y Sistemas Inteligentes,

- Sistemas Operativos, Sistemas Distribuidos y Redes
- Ingeniería de Computadores

Dentro de estas subcategorías deberían estar definidos los contenidos que se engloban en cada uno de los aspectos de la accesibilidad tratados. Respecto a la **accesibilidad del hardware** un informático debiera conocer la existencia de los diferentes dispositivos de Entrada / Salida y ser capaz de asesorar acerca de estos productos. Estos contenidos podrían estar recogidos en la subcategoría de *Ingeniería de Computadores*.

Teniendo esta formación, sería más difícil que se dieran hechos curiosos como que no sea posible imprimir Braille desde un procesador de textos como Word por ejemplo, ya que no existen drivers en España de impresoras Braille para Windows, o que una línea Braille resulte enormemente cara (pensemos que únicamente representa unos 80 caracteres de una sola vez, y cuesta más de treinta mil euros) porque utiliza una tecnología algo anticuada. Tal vez fuera posible abaratar costes si se dedicara algún tiempo a investigar sobre su mejora o sobre otros métodos de acceso a una computadora tanto para entrada de datos como para salida de información.

La **accesibilidad al software** podría tratarse en asignaturas como *Sistemas de interacción persona-computador*, *Dispositivos físicos para la Interacción persona-computador, Tratamiento digital de la palabra*, *Procesamiento de Lenguaje Natural, Interfaces Software en Lenguaje Natural, Procesamiento de la voz, Interfaces y Periféricos*, *Reconocimiento de formas, Entornos de usuario*. Podrían incluirse dentro de la Ingeniería software.

Hay que darse cuenta por ejemplo de que en el campo del reconocimiento de caracteres impresos aún queda mucho trabajo por hacer. En realidad es poco el material que se llega a leer correctamente, ya que existen muchos documentos como textos manuscritos, prospectos con cambios de color o tipografía, que resultan complejos y hacen difícilmente identificable su contenido.

En general, se debería incorporar en las distintas fases del diseño de una aplicación software el concepto de *accesibilidad*. De esta forma, el producto final podría ser utilizado por cualquier persona, ya que las aplicaciones se adaptarían a cada cual en función de sus necesidades. Dado que el diseño, desarrollo y mantenimiento del software es competencia casi exclusiva de un informático, hay varias materias en las que las cuestiones de accesibilidad al software para las personas con discapacidad deben ser recogidas, como *Sistemas Operativos, Programación, Sistemas de Información y por supuesto Ingeniería Software.*

Ya existen contenidos relacionados con la Ingeniería Web, que sin duda integran muchos de los conocimientos adquiridos en otros campos. Es necesario que formemos a titulados que sean capaces de desarrollar **navegadores** accesibles. Estos navegadores son software, por tanto deben tratarse en las categorías mencionadas anteriormente, pero por sus características concretas además deben ser tratados en las materias relacionadas con *Redes y Sistemas Distribuidos*. Un navegador manejado con la voz de su propietario ya es una realidad, aunque no se nos escapa que en el campo de reconocimiento del lenguaje oral aún hay mucho trabajo por hacer.

Por supuesto para la elaboración de **Páginas Web** son diversas las materias que intervienen, todas las derivadas de la *Ingeniería Software*, la *Programación* y el conocimiento en ***Redes***.

Además del currículo formativo, no debemos olvidarnos de la continua evolución de las nuevas tecnologías, por ello desde la Universidad se debe favorecer la investigación y la innovación en el campo de la Accesibilidad. Con este fin fue creado hace unos años un grupo de investigación en el seno de la Universidad de Extremadura. Este grupo se mantiene y continúa desarrollando una herramienta que permita el acceso a Internet de personas con discapacidad visual, del que forman parte los autores de este artículo.

5. Conclusiones

Un 9% de la población española presenta algún tipo de discapacidad, y más del 35% de la población es beneficiario de la accesibilidad en general. Esto nos indica que favorecer el "diseño para todos", no es una cuestión de favorecer a un pequeño sector, si no a toda la población.

A través de este trabajo hemos visto los esfuerzos de las distintas administraciones, europea y estatal, para establecer un marco legislativo y planes de actuación que permitan el aumento de la accesibilidad en todos los aspectos de nuestra sociedad, incluyendo la sociedad de la información y las nuevas tecnologías.

A menudo no se ha profundizado en la accesibilidad a las nuevas tecnologías por su novedad y continuo cambio. En este artículo se ha expuesto cómo las herramientas informáticas y sobre todo Internet constituyen una importante ayuda y a la vez barrera en la integración social y laboral de personas con algún tipo de discapacidad. Como docentes de futuros profesionales de las nuevas tecnologías, Ingenieros en Informática, nos sentimos en la obligación de incorporar los contenidos relacionados con la accesibilidad al currículo formativo del futuro profesional. Estamos en el momento adecuado para realizar esta inclusión, que además del apoyo de los planes de actuación estatal, cuenta con el marco del nuevo EEES.

Así, los futuros informáticos podrán hacer que cada vez sea más frecuente la creación de equipos informáticos y aplicaciones con un mayor grado de accesibilidad para todos y que se investiguen nuevas formas de obtención de la información existente en la Web, construyendo realmente una Sociedad de la Información sin exclusiones.

Referencias

[1] "La accesibilidad en España". Observatorio de las Telecomunicaciones y de la Sociedad de la información (red.es). Ministerio de Industria, Comercio y Turismo. Noviembre 2004.

[2] La Comisión Europea. Sociedad de la Información eEurope. Objetivos. http://europa.eu.int/comm/information_society/eeurope/objectives/ index_es.htm

[3] Áreas Prioritarias. http://europa.eu.int/ comm/information_society/eeurope/objectives/10areas_ es.htm

[4] SID@R. Una sociedad de la Información para todos. Prioridad 7: e-Participación de las personas con discapacidad. http://www.sidar.org/eeurope/eurotext.htm

[5] Plan INFOXXI. http://www.setsi.mcyt.es/infor_XXI/I21/strc_f.htm

[6] La Comisión Europea. Sociedad de la Información eEurope 2005. http://europa.eu.int/information_society/eeurope/index_en.htm

[7] Ley 51/2003 de Igualdad de Oportunidades, no Discriminación y Accesibilidad Universal de las Pesonas con Dispabacidad. BOE 2.12.2003.

[8] Ley 13/1982 de Integración Social de los Minusválidos. BOE 30.04.1982

[9] Ley 34/2002 de Servicios de Sociedad de la Información y de Comercio Electrónico. BOE 11.07.2002

[10] II Plan de Acción para las personas con discapacidad 2003-2007. Aprobado en Consejo de Ministro 5.12.2003

[11] I Plan Nacional de Accesibilidad 2004-2012. www.seg-social.es/imserso

[12] Plan de choque para el Impulso de la Administración Electrónica en España. Ministerio de Ciencia y Tecnología y Ministerio de Administraciones Públicas 8.05.2003.

[13] W3C Recomendation. Pautas de Accesibilidad del Contenido en la Web 1.0. 05-05-1999. http://www.w3.org/WAI/

[14] España.es: Programa de actuaciones para el Desarrollo de la Sociedad de la Información en España. Ministerio de Ciencia y Tecnología. 11.07.2003

[15] La accesibilidad: más que una cuestión de detalle. Editorial de la revista UTLAI Revista trimestral de la Asociación cultural "Usuarios de Tiflotecnología para el Libre Acceso a la Información" Número 8. Enero 2.001.

[16] Unitat de Investigació ACCESO. Universidad de Valencia. Estudio de accesibilidad a la red. 98. http://acceso.uv.es/accesibilidad/estudio/

[17] Lebihan, Rachel. ZDNet Australia. Olympic site must race for blind acessibility. 29-08-2000. http://www.zdnet.com.au/zdnn/stories/zdnn_display/au0005194.html

[18] Steel, Warren. Hints for Web Authors. http://www.mcsr.olemiss.edu/~mudws/webhints.html

[19] W3C Recommendation. Authoring Tool Accessibility Guidelines 1.0. 03-02-2000. http://www.w3c.org/TR/ATAG10/.

[20] W3C. Web Accessibility Initiative. Evaluation, Repair, and Transformation Tools for Web Content Accessibility. http://www.w3.org/WAI/ER/existingtools.html

[21] Libro Blanco de Ingeniería Informática. (http://www.aneca.es/modal_eval/docs/libroblanco_informatica.pdf)

Una propuesta de adaptación al EEES para la Arquitectura de Computadores

José M. García y Manuel E. Acacio
Departamento de Ingeniería y Tecnología de Computadores
Universidad de Murcia
30080 Murcia (España)
e-mail: {jmgarcia,meacacio}@ditec.um.es

Resumen

Con la próxima entrada de los nuevos planes de estudio motivados por el Espacio Europeo de Educación Superior, estamos viviendo unos momentos de cambios profundos en la Universidad. En esta ocasión no se trata tan sólo de modificar más o menos los contenidos docentes que se imparten en cada una de las materias, sino de implantar una nueva metodología docente.

En este artículo presentamos las actividades que hemos estado realizando en estos últimos cuatro años dirigidas a orientar la asignatura de Arquitectura de Computadores en esta nueva metodología, cuyo enfoque principal está centrado en el aprendizaje del alumno. En general, calificaríamos nuestra experiencia de muy satisfactoria, dado que hemos podido constatar que los alumnos mejoran sus conocimientos sobre la materia, aumentando también su rendimiento académico e interés por los conceptos analizados. La opinión de los alumnos también es favorable, obteniendo una valoración positiva por el 87 % de los mismos.

1. Introducción y motivación

La enseñanza universitaria cubre un amplio abanico de materias y de formas de enseñar. En cualquiera de sus formas, el profesor universitario trata de transmitir unos conocimientos a los alumnos que tiene delante, procurando que dichos alumnos asimilen, comprendan e incluso lleguen a hacer suyos (*interioricen*) los contenidos trasmitidos. Para conseguir este objetivo, el profesor cuenta con una serie de recursos didácticos que ayudan a que el estudiante se interese por la materia, le dedique el tiempo necesario a su comprensión y aprendizaje, e incluso llegue a *disfrutar* estudiando dichos conocimientos.

Hasta ahora, la forma tradicional de enseñar estaba centrada sobre todo en las enseñanzas impartidas por el profesor, y se manifestaba en la importancia que se le atribuía a la clase magistral. Aunque existían otras actividades complementarias como las prácticas, especialmente en las carreras científicas y técnicas, la labor del alumno era meramente pasiva y se dedicaba a estudiar una serie de nuevos conocimientos y a manifestar, por medio de un examen, el grado de asimilación de dichos conocimientos. Por contra, el nuevo Espacio Europeo de Educación Superior, siguiendo un modelo de aprendizaje más anglosajón, pone el acento sobre todo en el alumno y en su proceso de aprendizaje.

El alumno es ahora el verdadero protagonista de la educación universitaria. La labor del profesor pasa a tener más un carácter de tutoría, de guiado. Es el profesor, con sus abundantes conocimientos de la asignatura y su experiencia docente, el que debe orientar a cada alumno para que consiga el nivel de conocimientos exigido en dicha materia, por medio de la realización de diversas actividades. Es decir, se busca una atención personalizada al alumno. Aunque muchos alumnos recorran caminos semejantes, en principio –o en teoría– cada uno de ellos debe tener su propio desarrollo curricular.

Esta nueva concepción del método docente es la que nos ha motivado a poner en práctica, y a experimentar, con unas actividades complementarias que permiten a los alumnos desarrollar su trabajo de forma autónoma, y a convertirse, en la medida de lo posible, en los verdaderos protagonistas de su formación académica. Para ello, hemos acudido a las universidades de los países anglosajones para ver qué metodología desarrollan. A partir de ahí hemos procurado adaptar esta metodología a nuestra realidad, teniendo sobre todo en mente la convergencia europea de los próximos años.

Al ser la Informática una disciplina reciente y en continua evolución, tanto los conocimientos impartidos como su forma de transmisión están en un cambio continuo. En el caso de la Arquitectura de Computadores quizás la gran evolución (o más bien revolución) vino de la mano de Hennessy y Patterson, dos conocidísimos profesores estadounidenses que en el año 1990 publicaron un libro [5] que revolucionó la docencia (y la investigación) en esta materia. Con su famoso "enfoque cuantitativo" pusieron de manifiesto la importancia y creciente necesidad de medir y evaluar cualquier nueva idea en Arquitectura de Computadores, por medio de una metodología correcta y que pudiera ser reproducible por otros. A partir de entonces, prácticamente cualquier libro importante en Arquitectura de Computadores está desarrollado bajo este enfoque (por ejemplo, ver [8] o [2]).

Como ayuda y complemento eficaz en la docencia de esta materia, dichos profesores empezaron a pedir a sus alumnos la realización de un mini-proyecto de investigación que les ayudara a fijar ideas y a profundizar en los conocimientos adquiridos. Asimismo, empezaron a mandar una serie de ejercicios (problemas) para que los alumnos los resolvieran en su tiempo libre, lógicamente antes de la próxima clase. Finalmente, y debido a la rápida obsolescencia de los libros utilizados, aconsejaban a sus alumnos la lectura de artículos de investigación de las mejores revistas y congresos, como medio de adquirir los últimos conocimientos que se desarrollaban en este campo. Esta metodología de trabajo pronto hizo fortuna y hoy en día un buen número de universidades ya la han adoptado. En este artículo pretendemos explicar las *bondades* que nosotros hemos encontrado a la incorporación de estos recursos a la enseñanza de la Arquitectura de Computadores, así como mostrar la experiencia que hemos obtenido en su aplicación en estos últimos años.

Tras esta introducción, el resto del artículo se estructura de la siguiente manera. En la siguiente sección ofrecemos un breve apunte de los aspectos más relevantes del Espacio Europeo de Educación Superior. En la sección 3 presentamos cada una de las actividades complementarias que hemos añadido a la asignatura, mostrando el enfoque que les hemos dado y las características más destacadas que presentan. A continuación, en la sección 4 mostramos cómo ha sido la experiencia práctica que hemos tenido durante estos años, ofreciendo en la sección siguiente los resultados obtenidos y la opinión de los alumnos. Finalmente, el artículo termina con una sección de conclusiones acerca de todo lo expuesto.

2. El Espacio Europeo de Educación Superior

El Espacio Europeo de Educación Superior supondrá una importante modificación del actual marco curricular, ya que habrá que cambiar gran parte del sistema universitario español para armonizarlo con el del resto de países de la Unión Europea.

La construcción del Espacio Europeo de Educación Superior es un proceso que se inicia con la Declaración de la Sorbona y se consolida y amplía con la Declaración de Bolonia, en las que los ministros europeos de educación instaban a los estados miembros de la Unión Europea a desarrollar e implantar en sus países una serie de actuaciones, conducentes a adoptar un sistema de titulaciones comprensible y comparable entre los diversos países, basado en una estructuración en dos niveles principales, el de grado y el de postgrado (tipo máster y/o doctorado). Para fomentar la comparabilidad de los estudios y promover la movilidad de los estudiantes y titulados se propugna es-

tablecer un sistema común de créditos.

El sistema de créditos europeos, conocido como ECTS (*European Credits Transfer System*), nace y se desarrolla con los programas de movilidad de estudiantes para dar una respuesta a la necesidad de encontrar un sistema de equivalencias y de reconocimiento de los estudios cursados en otros países. La generalización de esta unidad de medida académica para todos los estudiantes es un objetivo fundamental para la creación del Espacio Europeo de Educación Superior, de forma que el trabajo desarrollado por un estudiante en cualquiera de las universidades de los estados miembros sea fácilmente reconocible en cuanto a nivel, calidad y relevancia.

Su introducción en el sistema universitario español implica importantes diferencias con respecto al crédito vigente. Conviene subrayar, al respecto, que el crédito europeo no es una medida de duración temporal de las clases impartidas por el profesor, sino una unidad de valoración del volumen de trabajo total del alumno, expresado en horas, que incluye tanto las clases (teóricas o prácticas) como el esfuerzo dedicado al estudio y a la preparación y realización de exámenes. En resumen, esta nueva unidad de medida comporta un nuevo modelo educativo basado en el trabajo del estudiante y no en las horas de clase, o dicho de otro modo, centrado en el aprendizaje de los estudiantes, y no en la docencia de los profesores.

3. Nuevos recursos docentes en Arquitectura de Computadores

De cara a propiciar un cambio en el enfoque docente, desde hace unos años hemos empezado a introducir en la asignatura de Arquitectura de Computadores toda una serie de actividades complementarias que permitan un trabajo más personal y autónomo por parte de los alumnos. En el contexto de este artículo, entendemos por Arquitectura de Computadores los conocimientos actualmente impartidos a los alumnos de la titulación de Ingeniero en Informática en alguno de los cursos de 2º ciclo, con un carga de al menos 9 créditos. Estos estudios comprenden una serie de tópicos avanzados en Arquitectura de Computadores en dos grandes líneas: el diseño de arquitecturas que explotan el paralelismo a nivel de instrucción (ILP), como pueden ser los procesadores superescalares, VLIW o vectoriales, y el diseño de arquitecturas que explotan el paralelismo gracias a tener varios procesadores (arquitecturas multiprocesador).

En esta sección vamos a explicar los nuevos recursos docentes que hemos empleado en la asignatura: el proyecto de investigación, la lectura de artículos científicos, los boletines de ejercicios y la lista de correo de la asignatura.

3.1. El proyecto de investigación

Empezaremos nuestra propuesta por detallar *qué es* un proyecto de investigación. Entendemos por proyecto de investigación un trabajo realizado por los alumnos en el que tratan de demostrar los conocimientos adquiridos en la asignatura, por medio de *razonar* e *investigar* sobre un aspecto concreto de la misma, teniendo que relacionar diversos conocimientos que han ido asimilando a lo largo de la carrera, y debiendo redactar un documento científico en el que muestren la investigación realizada y los *hallazgos* derivados de dicha investigación.

A la hora de proponer los posibles trabajos a realizar, o de permitir que el alumno elija un cierto trabajo de su interés particular, trataremos de que los trabajos tengan una alta componente de investigación de tipo experimental, y no únicamente bibliográfica. Al final de la fase experimental, los alumnos redactan un documento explicando qué es lo que han desarrollado, qué metodología han seguido para ello y qué resultados han alcanzado. Dicho documento tiene que tener la estructura de un artículo científico, por lo que debe incluir una introducción y motivación a dicho trabajo, un repaso de qué han hecho otros autores en casos parecidos, para finalizar con unas conclusiones y, si es posible, con unas líneas de trabajo futuro. Para facilitarles dicha tarea, les proporcionamos un documento [3] que pueden usar como guía en dicho proceso.

Dentro de la asignatura, el proyecto de investigación forma parte de los créditos prácticos dedicados a ésta. En estos años, hemos

dedicado la mitad de dichos créditos a la realización de dos prácticas sencillas correspondientes a cada uno de los bloques en que dividimos la asignatura. El resto de los créditos los reservamos para que los alumnos realicen el trabajo de investigación que hayan elegido.

3.2. Lecturas recomendadas

Como complemento a la formación impartida en el aula, hemos visto conveniente que el alumno tome soltura en acostumbrarse a leer artículos de investigación. En función de los contenidos que se imparten en la asignatura, estos artículos los hemos agrupado en tres bloques: el primero se refiere a aquellos trabajos relacionados con la mejora de prestaciones en los procesadores por medio de explotar adecuadamente el paralelismo a nivel de instrucción, el segundo bloque trata de los diferentes tipos de arquitecturas multiprocesador y finalmente, el tercer bloque se dedica al problema de la consistencia de memoria y la coherencia de las cachés en los sistemas multiprocesador.

Con las lecturas de investigación conseguimos un efecto colateral, y es que el alumno se acostumbre a utilizar la bibliografía recomendada por el profesor para cada uno de los temas que se imparten, procurando vencer una natural *alergia* que parece que tienen frecuentemente los alumnos a los libros de consulta, en especial cuando están escritos en inglés.

La selección de los artículos la hemos realizado teniendo en cuenta el impacto que han tenido, el lugar de su publicación (habitualmente en congresos tipo ISCA o HPCA o revistas de IEEE o ACM), la originalidad de la idea mostrada y la legibilidad por parte de los alumnos.

Esta actividad docente cada vez está ganando más adeptos, y ya empieza a haber libros de consulta que la recomiendan, como es el caso, por ejemplo de [8] o de [7].

3.3. Boletines de ejercicios y lista de correo

Una de las razones que, a nuestro juicio, hace que los alumnos obtengan bajas calificaciones y un bajo rendimiento académico es la falta de hábitos de trabajo. Para solucionar esto, y ayudar a que la asignatura la vayan estudiando poco a poco, proponemos la realización de unos boletines de ejercicios (problemas) fuera del horario de clases.

Además de dedicar parte de las horas de teoría a explicar cómo se resuelven los problemas de la asignatura, ofrecemos a los alumnos la posibilidad de realizar 3 boletines de *"Ejercicios para casa"*. Dichos boletines consistirán en 5 ó 6 problemas acerca de la materia correspondiente. Posteriormente a la fecha de entrega, publicamos su respuesta y comentamos en clase las líneas generales de resolución de los mismos. El primero de estos boletines lo entregamos en las primeras semanas de curso, y sirve para *refrescar* los conocimientos de años anteriores, mientras que los otros dos están más relacionados con cada uno de los bloques en que hemos dividido la asignatura.

De forma adicional, estamos usando una lista de correo electrónico para el ágil manejo de la asignatura. Aprovechando que en el servidor del Departamento está instalado el software `Mailman` (de libre distribución), tenemos activada una lista de correo para todos los alumnos de la asignatura[1]. Lo primero que se les pide a los alumnos es que se den de alta en dicha lista con el correo electrónico que suelen utilizar habitualmente[2]. Una vez dados de alta, cada uno de ellos recibe todos los mensajes que cualquier otro envía a dicha lista. La lista está administrada por los profesores de la asignatura (aunque de momento no moderada), y los propios profesores forman parte de dicha lista (detalle este último que se les advierte a los alumnos).

Hemos encontrado diversos beneficios a la utilización de una lista de correo para los alumnos de la asignatura. El primero y más importante es la consulta de las diversas dudas que les surgen en la realización de los problemas, prácticas, estudiando la asignatura, etc. Dichas dudas, puestas en la lista, pueden ser contestadas por otros compañeros (cosa que de

[1] La dirección de la lista es `aic@ditec.um.es`

[2] Aunque la Universidad les proporciona una dirección de correo a todos los alumnos, en bastantes ocasiones ellos utilizan otra cuenta de correo.

hecho sucede con frecuencia) o bien por el profesor, caso de una pregunta más difícil o bien de una contestación previa errónea. Además, y de cara a facilitar las tutorías por *email*, hemos encontrado que muchas veces las consultas realizadas al profesor eran muy interesantes y la respuesta, además de enviársela al alumno que la formuló, ha sido re-enviada también a la lista, para que de esta forma todos los alumnos conozcan dicha respuesta. Por último, hemos encontrado que la lista de correo es un medio muy útil para avisar a todos los alumnos de cualquier incidencia que pueda ocurrir: cambio de una clase, retraso en la entrega de unas prácticas, *publicar* en la página Web (y dejar en la fotocopiadora) las transparencias de un tema determinado, etc.

4. Experiencia práctica

En esta sección presentamos algunos detalles concretos de cómo hemos llevado a la práctica las actividades complementarias descritas, dentro de la asignatura de Arquitectura de Computadores de 4º curso de la Ingeniería Informática en la Universidad de Murcia.

En primer lugar, decir que la idea que hemos puesto en práctica es que todas estas actividades complementarias no tengan que ser obligatorias, aunque se les dé un peso importante en la nota final de la asignatura, como después comentaremos. Eso quiere decir que un alumno puede aprobar (e incluso obtener buena calificación) sin necesidad de realizar alguna de las actividades propuestas. De esta forma, pensamos que los alumnos gozan de una mayor libertad y se les motiva más para que se lancen a la *aventura*.

4.1. El proyecto de investigación

Empezamos por unos comentarios acerca del proyecto de investigación. En cuanto al número de personas que tienen que realizar juntas el trabajo, recomendamos los grupos de 2 alumnos, pues de esa forma obtienen el beneficio adicional de trabajar de forma conjunta en la resolución de una tarea compleja. Opcionalmente, también se puede realizar el trabajo de forma individual pero, salvo casos muy excepcionales, no admitimos grupos de tres o más personas. Como *esperamos* que ambos componentes del grupo trabajen de forma equitativa, la nota obtenida en el trabajo será la misma para ambos.

Para la elección del trabajo de investigación, proponemos una serie de trabajos relacionados con cada uno de los 2 bloques que componen la asignatura. Esta propuesta la realizamos a título orientativo, pudiendo el alumno, de acuerdo con el profesor, elegir otro trabajo por el que tenga un mayor interés. Dentro de cada bloque, los trabajos se dividen entre complejidad normal y complejidad superior, de cara a orientar a los alumnos acerca de la dificultad de cada trabajo, no teniendo, en principio, ninguna relación en cuanto a la nota a obtener en él. Para lo que sí sirve dicha distinción es de cara a la posible continuidad del trabajo en un proyecto fin de carrera. Este es el caso de los trabajos de complejidad mayor, los cuales habitualmente pueden abrir las puertas a desarrollos futuros.

Con respecto al resto de detalles del proyecto de investigación, remitimos al lector interesado a la página Web[3] de la asignatura, o a la referencia dada en [4].

4.2. Las lecturas y los boletines de ejercicios

Para comprobar que los alumnos han leído los artículos sugeridos, deben entregar un resumen de dichas lecturas con una extensión máxima de un folio (por las dos caras) y manuscrito. La organización aconsejada para el resumen es la siguiente: mostrar los objetivos del autor del artículo, a continuación comentar los principales desarrollos que se exponen en el mismo, y acabar dando una opinión personal de los logros obtenidos por el autor y su importancia y relevancia en el campo de la Arquitectura de Computadores, así como cualquier otra opinión personal que se considere oportuna.

[3] http://www.ditec.um.es/arquitectura. El material utilizado está disponible a cualquier persona que lo solicite por email.

Lo que nos interesa es que el alumno lea dicho artículo de investigación y lo entienda. No se trata de hacer *simplemente* un resumen del artículo, sino de ganar en familiaridad con la forma de presentar los contenidos científicos, *deslumbrarse* por los problemas planteados y sorprenderse con las soluciones aportadas por los diversos autores. Lo que más se valora de la redacción del trabajo es el enfoque que se le da al mismo y la opinión personal que le dan los alumnos a lo leído. Se sobreentiende, al ser artículos de investigación avanzada, que puede haber detalles poco comprensibles, lo que no impide el correcto aprovechamiento de las lecturas realizadas.

Con respecto a los boletines de ejercicios, se prepara una colección de problemas seleccionados, intentando que cubran los diferentes aspectos tratados. Muchos de los problemas los hemos escogido de los propios exámenes realizados en la asignatura en años anteriores, permitiendo así a los alumnos tener una idea de lo que se les va a exigir cuando se enfrenten a su propia prueba de evaluación.

Los alumnos tienen aproximadamente dos semanas para la realización de estos boletines, teniendo que entregar en plazo fijo la resolución manuscrita de los ejercicios propuestos. Una vez pasada dicha fecha, se les proporciona a los alumnos la solución de los problemas. Al cabo de un tiempo para que hayan podido revisar las soluciones y comprobar sus resultados, se resuelven en clase prestando especial atención a aquellos puntos en los que han tenido una mayor dificultad o no ha quedado suficientemente clara la respuesta que les hemos facilitado.

Hemos comprobado que también resulta interesante para los alumnos el disponer de una colección de problemas adicionales resueltos (incluyendo, de nuevo, exámenes de otros años). Por ello, facilitamos dicho material en la página Web de la asignatura (y en la fotocopiadora).

4.3. El proceso de evaluación

Finalmente, nos resta comentar algo acerca del proceso de evaluación de todas estas actividades.

Para la evaluación del trabajo de investigación, nuestra propuesta es realizarla a partir del trabajo manuscrito que entregan los alumnos. En el momento de presentar a los alumnos el trabajo de investigación, se detallan los criterios que vamos a seguir para su evaluación. Una explicación más detallada se puede encontrar en [4].

Para la evaluación de los boletines de ejercicios, proponemos la auto-evaluación por parte de los alumnos. Al finalizar el comentario de clase a las respuestas a los problemas, los alumnos tienen que corregirse dichos ejercicios y enviar al profesor (habitualmente por *email*) la nota obtenida. Para evitar abusos, el profesor aleatoriamente corrige un 20 % de los ejercicios entregados. En caso de que la diferencia de nota sea significativa, el alumno es llamado al despacho del profesor para que explique el motivo de dicha diferencia.

Vemos dos grandes ventajas a proceder de este modo: por una parte, no sobrecargar la labor del profesor con excesivas correcciones de actividades de los alumnos; por otra parte, obligamos a que el alumno *vuelva* sobre los ejercicios entregados, los observe con espíritu crítico, descubra en qué se ha equivocado y se valore. De esta forma, los alumnos aprenden de sus errores y progresan mejor en la asignatura. Con respecto a la *bondadosa* calificación que pueden realizar los alumnos sobre ellos mismos, hemos comprobado que su influencia en la nota final de la asignatura es escasa o prácticamente nula.

Las lecturas de artículos es el apartado sobre el que la corrección es más benigna por nuestra parte, ya que somos conscientes de que los alumnos no están acostumbrados a leer un artículo, hacerse cargo de su contenido y poder ofrecer un resumen con comentario personal incluido. Por ello, el criterio que seguimos es que prácticamente por entregar cada lectura ya está aprobada con una nota de 7. En función de la redacción y el contenido, esta nota sube en numerosas ocasiones a 8, 9 ó 10. También, aunque los casos son más raros, los alumnos podrían obtener una nota más baja cuando se observe una auténtica desgana en la realización de dicha lectura.

La otra cuestión relacionada con el proceso de evaluación es determinar cuánto influyen cada una de las actividades descritas en la nota global obtenida por el alumno en la asignatura. Desde el principio teníamos claro que queríamos ofrecer a los alumnos una asignatura mucho más participativa, y con un reflejo claro en la nota final que obtienen los alumnos. Al mismo tiempo, queríamos que el alumno realizara estas actividades de forma voluntaria, por lo que debíamos permitir que cualquier alumno pudiera aprobar al realizar únicamente las actividades tradicionales (exámenes y prácticas).

Contando con todo lo dicho, proponemos calcular la nota final de la asignatura de acuerdo a las siguientes proporciones: 35 % Exámenes de la asignatura (teórico-prácticos), 15 % Prácticas, 25 % Trabajo de investigación, 10 % Boletines de ejercicios, 10 % Lecturas y 5 % Participación en clase y tutorías.

5. Resultados obtenidos

Desde nuestro punto de vista, la experiencia en estos años ha sido muy positiva. Hemos notado que los alumnos responden adecuadamente a la idea de tener que trabajar por su cuenta y organizarse el trabajo por ellos mismos. Además, y esto es lógicamente lo más importante, hemos comprobado cómo los alumnos mejoran su comprensión de los conceptos propios de la asignatura.

Por otra parte, los alumnos obtienen unos beneficios adicionales con la realización de estas actividades complementarias. En primer lugar, el alumno mejora sensiblemente su comprensión de los tópicos impartidos en la asignatura, desarrollando su intuición, entendiendo mejor las causas que han desencadenado los últimos avances en Arquitectura de Computadores y las soluciones que se han aportado hasta el momento. Asimismo, el alumno se acostumbra a leer y a redactar un artículo científico, permitiéndole esta tarea mejorar su expresión escrita y ayudándole a reflexionar sobre los conocimientos adquiridos.

Pero no debemos dar únicamente nuestra opinión. Al finalizar la asignatura, tenemos por costumbre pasar a los alumnos una encuesta anónima para recabar su opinión sobre los diversos aspectos de la asignatura.

A partir de los resultados obtenidos en estos últimos años en la encuesta, hemos extraído las siguientes conclusiones:

- Las actividades complementarias están muy bien valoradas por los alumnos: El trabajo de investigación está valorado con una puntuación de 4 ó 5 (sobre 5) por el 81 % de los alumnos que realizan la asignatura, los boletines de ejercicios por el 85 %, llegando las lecturas a gozar del favor del 93 % de los alumnos.

- El 96 % de los alumnos opina que el trabajo de investigación le ha servido para reforzar los conocimientos adquiridos en la asignatura.

- Con respecto al grado de dificultad para realizar todas estas actividades, hay una mayor diversidad de opiniones. Aunque sólo el 25 % encuentra excesivo dicho grado de dificultad, es cierto que para la mayoría (82 %) las actividades complementarias suponen un esfuerzo considerable.

- Finalmente, comentar que a la gran mayoría de los alumnos (87 %) les gusta este nuevo enfoque de la asignatura, pues les permite organizarse con más libertad y trabajar de forma continuada a lo largo de todo el cuatrimestre.

En general, los alumnos valoran positivamente la asignatura y la labor realizada por los profesores de la misma. En general, es bastante frecuente que los alumnos reconozcan que, a priori, no era ésta una de las asignaturas por la que mostraran sus preferencias, pues habitualmente están más inclinados por otras materias en las que perciben una mayor "aplicabilidad" de lo que aprenden. Sin embargo, según ellos mismos reconocen, al final del curso su opinión ha cambiado, pasando de ser la Arquitectura de Computadores una materia "puramente" teórica, a descubrir el enfoque *cuantitativo* que tiene y la utilidad que más adelante tendrá en el ejercicio de su profesión.

Quizá un último dato que refleja este cambio es el número de alumnos que realizan el proyecto fin de carrera en el área de Arquitectura, y que ha pasado de estar en torno al 8 %, a sobrepasar el 20 % de los alumnos.

6. Conclusiones

En este artículo hemos presentado una propuesta sobre cómo incorporar unas actividades adicionales a la docencia de los contenidos de Arquitectura de Computadores que se imparten en el segundo ciclo de la titulación de Ingeniero en Informática. Dichas actividades están en consonancia con la idea del Espacio Europeo de Educación Superior, en donde se le otorga un mayor énfasis al trabajo personal y autónomo del alumno. A pesar de que nuestra experiencia se centra en esta materia, nos parece que la mayoría de las ideas mostradas en este artículo se pueden trasladar a otras muchas materias del segundo ciclo.

La incorporación de éstas u otras actividades dentro de la docencia de Arquitectura de Computadores es una práctica común en las universidades americanas. En este artículo hemos mostrado cómo hemos llevado a la práctica dicha metodología docente, la experiencia que nos hemos encontrado en estos años y las diversas características que hemos aplicado para conseguir este fin.

Quizá la conclusión obtenida más importante ha sido comprobar la mejora de los alumnos en sus conocimientos en esta materia. El tener que realizar estas actividades complementarias obliga a los alumnos a tener que *pensar* y *razonar* acerca de todos los conceptos explicados en ésta y otras asignaturas de la carrera. Además, el enfrentarse a redactar un documento acerca del trabajo realizado, y a expresar por escrito sus ideas, les facilita que muchas veces descubran que hay conceptos que no tenían aún suficientemente claros y que necesitaban de un estudio y/o análisis adicional. Asimismo, la oportunidad de constatar en la práctica aquello que se ha estudiado en las clases teóricas aumenta el interés de los alumnos por la Arquitectura de Computadores.

Referencias

[1] Todd Austin, E. Larson, y D. Ernst. Simplescalar: An infrastructure for computer system modeling. *IEEE Computer*, 35(2):59–67, Febrero 2002.

[2] José Duato, S. Yalamanchili, y L. Ni. *Interconnection Networks: An Engineering Approach*. Morgan Kaufmann Publishers, San Francisco, USA, revised edition, 2002.

[3] José M. García. Guía para la preparación de informes técnicos y artículos de investigación. Technical Report DITEC-UM-2003-2, Universidad de Murcia, Marzo 2003.

[4] José M. García y Manuel E. Acacio. El Proyecto de Investigación: Un complemento eficaz en la docencia de Arquitectura de Computadores. *X Jornadas de Enseñanza Universitaria de la Informática*, Thomson Paraninfo, pp. 135-142, 2004.

[5] John L. Hennessy y David A. Patterson. *Computer Architecture. A Quantitative Approach*. Morgan Kauffmann, 1ª edition, 1990.

[6] Christopher J. Hughes, Vijay S. Pai, Parthasarathy Ranganathan, y Sarita V. Adve. Rsim: Simulating shared-memory multiprocessors with ilp processors. *IEEE Computer*, 35(2):40–49, Febrero 2002.

[7] Bruce Shriver y Bennett Smith. *The Anatomy of a High-performance Microprocessor: A Systems Perspective*. IEEE CS Press, 1998.

[8] William Stallings. *Computer Organization and Architecture: Designing for Performance*. Prentice Hall, 6ª edition, 2003.

CEDI 2005

Arquitectura de Computadores y Sistemas Operativos

Características y adecuación al aula del entorno BOOLE-DEUSTO de diseño lógico

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

Resumen

En el presente trabajo se estudian las necesidades computacionales de la clase de electrónica digital y se analiza la idoneidad de algunos de los entornos sw utilizados en el aula. De este estudio surge la necesidad de un nuevo sw, el BOOLE-DEUSTO, que es descrito y caracterizado. Seguidamente se compara el BOOLE-DEUSTO con el entorno LogicAid del profesor Roth de la Universidad de Tejas, para establecer sus ventajas y desventajas. Finalmente se presenta la experiencia de utilizar BOOLE-DEUSTO en el aula, incluyendo una encuesta a los alumnos.

1. Entornos profesionales y necesidades del aula

La figura 1 sitúa el ámbito práctico de la asignatura de electrónica digital o tecnología de los computadores, según sea la titulación. Un sistema digital puede ser implementado con transistores y puertas (nivel de bit), circuitos MSI y PLD's (nivel de palabra) o circuitos programables tipo FPGA y ASIC (nivel de sistema). Los sistemas de la asignatura son básicos, y por tanto su ámbito práctico-tecnológico no va más allá de los circuitos MSI.

La figura 1 muestra claramente que para el ámbito de diseño de nivel de bit muchas veces se usan en la asignatura entornos sw más orientados al nivel de palabra o de sistema. Es decir, se utiliza un entorno que no se adecúa a las necesidades del alumno y del aula. Pero, ¿cuáles son las necesidades del alumno y del profesor?

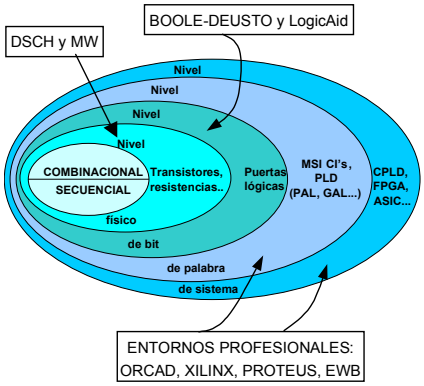


Figura 1. Niveles y tecnologías de electrónica digital

La asignatura es eminentemente práctica; cuando el alumno procede a analizar y diseñar sistemas digitales a nivel de bit (combinacionales o secuenciales) no hace otra cosa que aplicar con criterio diferentes métodos. Los métodos que el alumno debe aprender a aplicar pueden clasificarse en dos grupos ya clásicos:

- No Heurísticos. Métodos ordenados en pasos de aplicación sistemática. El resultado de aplicar un método no heurístico no depende de la experiencia del alumno, sólo depende del rigor con que se aplique el método. Por ejemplo, obtener la tabla de verdad de una expresión booleana.
- Heurísticos. No existe un método ordenado como tal, sino una serie de reglas. El resultado de aplicar un método heurístico depende de la experiencia del alumno y de lo complicado del problema. Por ejemplo,

simplificar una función booleana usando diagramas Veitch-Karnaugh.

La figura 2 ordena los métodos de análisis (flechas ascendentes discontinuas) y diseño (flechas descendentes) de sistemas combinacionales básicos a nivel de bit.

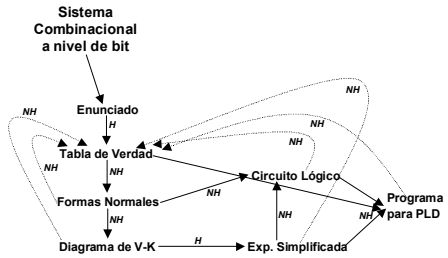


Figura 2. Análisis y diseño de sist. combinacionales

La figura 3 muestra el análisis y diseño de sistemas secuenciales a nivel de bit. La conclusión principal, compartida con los sistemas combinacionales, es que analizar y diseñar a nivel de bit no es más que pasar de una representación a otra equivalente.

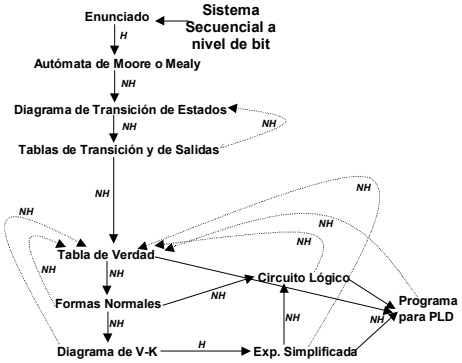


Figura 3. Análisis y diseño de sistemas secuenciales

Resumiendo, el análisis y diseño de sistemas combinacionales y secuenciales a nivel de bit consiste en la aplicación ordenada de una serie de métodos heurísticos y no heurísticos. La pregunta es ¿abordan los entornos profesionales el diseño en todos sus pasos? ¿lo hacen didácticamente? La respuesta a ambas preguntas es no. Y la razón es que a un entorno profesional le interesa el resultado, y no el proceso. El error lo comete el profesor que en un curso de iniciación utiliza entornos profesionales, que como ya se ha mostrado en la figura 1 están centrados en el diseño a nivel de palabra o sistema. El profesor debería utilizar el BOOLE-DEUSTO, o uno similar como el LogicAid.

¿Por qué es ideal el BOOLE-DEUSTO? Pues porque permite al alumno practicar con todos los métodos de una forma libre y amigable. La definición más clara del BOOLE-DEUSTO es la de *calculadora booleana*.

La tabla 2 confronta las características de los entornos profesionales y del BOOLE-DEUSTO. Se puede ver como en la mayoría de los casos las características resultan antitéticas, aunque no excluyentes.

ENTORNO BOOLE-DEUSTO	ENTORNOS PROFESIONALES
Didáctico	Profesional
Necesidades del alumno	Necesidades del profesional
Hasta nivel de bit	Hasta nivel de sistema
Proyectos sencillos	Proyectos complejos
El alumno controla a la herramienta	El usuario es dirigido por la herramienta
Interesa el proceso	Sólo interesa el resultado
Sin instalación	Instalación compleja
Es muy fácil de usar	Críptico y difícil de usar
Es gratis y de libre distribución	Coste generalmente elevado
No tiene simulación temporal	Sí tiene simulación temporal
No permite la captura gráfica de un circuito	Permite la captura gráfica de un circuito

Tabla 1. Comparación entre BOOLE-DEUSTO y los entornos profesionales

Al observar la tabla no se debe concluir erróneamente que el profesor debe optar por uno u otro, sino que el BOOLE es adecuado al principio de la formación del alumno, para trabajar en el aula y en casa (hasta el nivel de bit), y que los entornos profesionales son adecuados para trabajar en el laboratorio diseñando circuitos reales (desde el nivel de bit hasta FPGA's).

Resumiendo nuestra experiencia, podemos elegir cuatro entornos sw para abordar la

asignatura de diseño digital en toda su amplitud y complejidad:

- DSCH y MW [1]. Entorno de simulación de sistemas digitales a nivel físico MOS e incluso de layout, aunque también contempla el nivel de bit. Su uso está recomendado para los cursos superiores de electrónica donde se aborda el diseño físico de las puertas lógicas. Tiene una clara orientación didáctica.
- BOOLE-DEUSTO [2]. Entorno de análisis y diseño, y no de simulación, de sistemas digitales a nivel de bit. Útil en un primer curso de diseño lógico. Tiene una clara orientación didáctica.
- PROTEUS. Entorno de diseño y simulación de sistemas digitales de cualquier nivel. Útil para cualquier nivel de formación. Su orientación es claramente profesional, sin importarle las necesidades didácticas.
- WebPack ISE de Xilinx. Entorno de diseño y simulación de sistemas digitales implementados en VHDL y PLDs avanzados (CPLD, FPGA, etc). Exclusivamente útil en cursos de lógica programable o laboratorios. Tiene una orientación profesional con algunos recursos didácticos.

2. Descripción de BOOLE-DEUSTO

BOOLE-DEUSTO es un entorno sw de análisis y diseño de sistemas digitales básicos a nivel de bit, aunque lo mejor es definir a BOOLE-DEUSTO como una calculadora booleana. La siguiente lista recuerda someramente las posibilidades que ofrece para sistemas combinacionales y secuenciales:

- Expresiones booleanas.
- Tablas de verdad.
- Diagramas de Veitch-Karnaugh.
- Formas normales.
- Expresiones simplificadas o minimizadas.
- Expresiones NAND/NOR.
- Circuitos lógicos: AND-OR, OR-AND, NAND y NOR.
- Simplificador booleano especializado.
- Manejo gráfico de V-K para aprendizaje del alumno.
- Generador de código VHDL, ABEL y OrCAD-PLD para sistemas combinacionales.
- Captura gráfica del diagrama de transición de estados de Mealy o Moore.
- Verificación del autómata.
- Reducción o minimización de estados del autómata.
- Diseño del autómata: generación de las tablas.
- Conversión Moore-Mealy, y viceversa.
- Obtención del circuito lógico con J-K y D.
- Generador de código VHDL, ABEL y OrCAD-PLD para sistemas secuenciales.
- Simulación interactiva y batch de autómatas.

En cuanto al entorno en su conjunto, BOOLE permite:

- Salvar y cargar sistemas digitales.
- Imprimir o guardar resultados.
- Copiar al portapapeles las figuras y tablas.
- Editar un texto para cada sistema.
- Versiones en español, inglés y euskara. Se puede adaptar a más idiomas; actualmente en desarrollo el catalán y el portugués.

Genéricamente, BOOLE-DEUSTO destaca por cuatro aspectos:

- Adaptado a las necesidades del aula, tanto en su completitud como en el total control de la herramienta por parte del alumno.
- El entorno ofrece los métodos pero no los ordena en una secuencia de análisis o diseño, es el alumno el encargado de elegirlos y aplicarlos con criterio. BOOLE-DEUSTO no es un entorno dirigido, es una calculadora booleana.
- BOOLE-DEUSTO es gratuito, sólo tiene un .exe, no tiene licencias y se aprende a usar en 15 minutos. Es decir, cuando el profesor lo distribuye sabe que no tendrá que atender dudas ni problemas de instalación.
- Por último, BOOLE-DEUSTO permite al profesor generar, resolver y documentar ejercicios de forma rápida.

Veamos algunas imágenes de BOOLE-DEUSTO. En primer lugar, la figura 4 muestra el aspecto de la pantalla principal para los sistemas combinacionales, en ella se ofrecen todos los métodos disponibles.

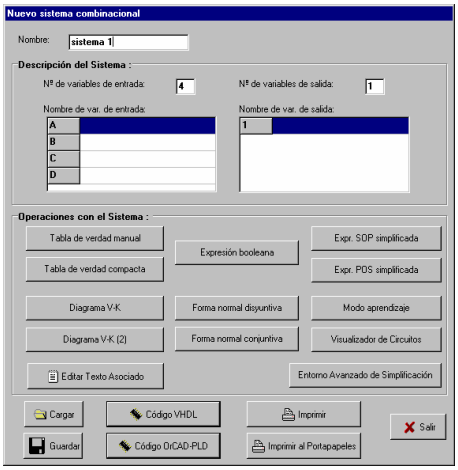


Figura 4. Menú principal de sistemas combinacionales

En la figura 5 podemos ver el aspecto de un V-K simplificado. En este caso cabe recordar que el alumno puede practicar introduciendo con simples clics de ratón la solución en forma de lazos, indicándole el BOOLE-DEUSTO la bondad de lo introducido. Ningún otro entorno conocido por este profesor permite una simplificación tan cómoda y didáctica para el alumno. El BOOLE-DEUSTO es una gran ayuda para el alumno y el profesor.

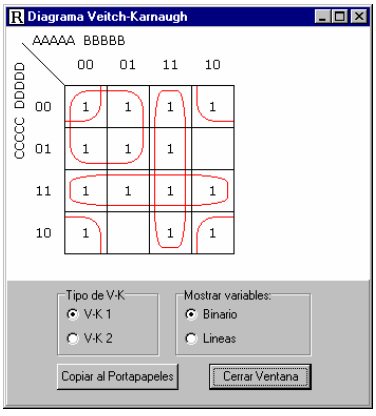


Figura 5. Visualización de lazos en el V-K

Para sistemas secuenciales el BOOLE-DEUSTO permite la captura gráfica del autómata a diseñar o simular. La figura 6 muestra el aspecto de un autómata de Mealy que podrá ser minimizado (Figura 7), simulado o diseñado.

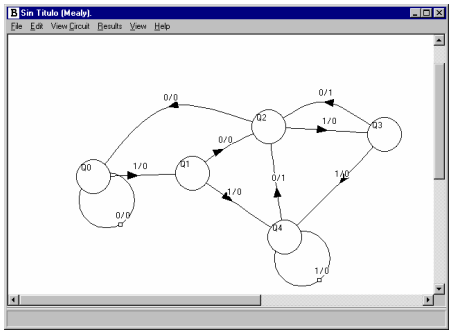


Figura 6. Autómata original de Mealy

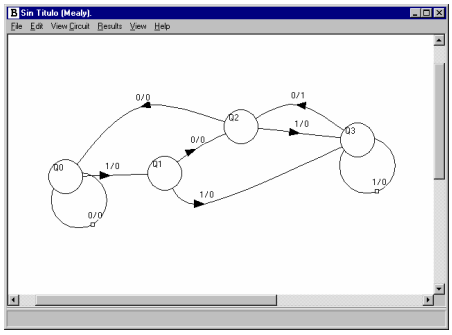


Figura 7. Autómata de Mealy reducido

BOOLE-DEUSTO también permite introducir un autómata incompletamente especificado. En este caso también permite su minimización, con las reservas propias para este tipo de algoritmos de simplificación. Las figuras 8 y 9 muestran un autómata de Moore incompleto y el correspondiente minimizado.

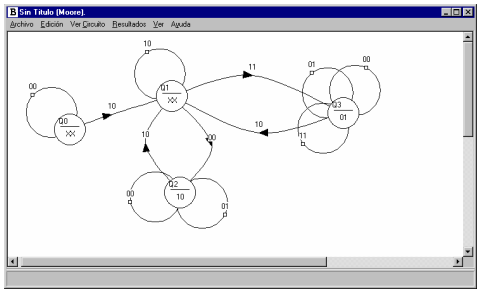


Figura 8. Autómata original de Moore

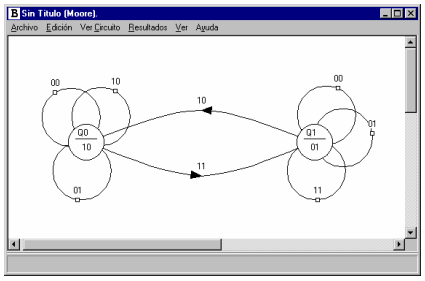


Figura 9. Autómata de Moore reducido

Finalmente BOOLE-DEUSTO permite obtener el circuito lógico con flip-flop J-K (Figura 9) o D del autómata minimizado.

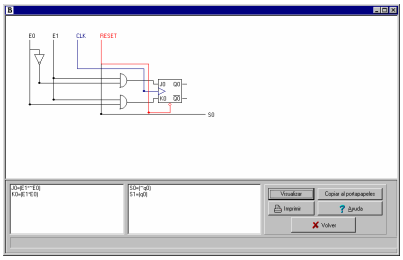


Figura 10. Circuito del autómata de Moore

Una de las últimas novedades del BOOLE-DEUSTO mejor aceptadas es la generación del código VHDL o del fichero JEDEC del sistema introducido, combinacional o secuencial. De este forma el alumno/usuario puede pasar del BOOLE-DEUSTO a un entorno profesional de diseño, obteniendo una implementación hardware tipo PLD. Esta opción es muy útil, pero aleja al BOOLE-DEUSTO de entorno original: el aula, llevándolo al laboratorio. De hecho BOOLE-DEUSTO no va a implementar ningún método "útil" más, pensamos que este es un objetivo de los entornos profesionales.

3. BOOLE-DEUSTO *versus* LogicAid

Hace tres años encontramos un sw cuyos objetivos parecían ser sorprendentemente idénticos a los del BOOLE-DEUSTO: el LogicAid y el SimUaid. El autor del sw es el profesor Charles H. Roth de la Universidad de Tejas y lo distribuye con su libro [3]. El objetivo inmediato, cuyos resultados presentamos aquí, fue comparar BOOLE-DEUSTO con LogicAid.

La tabla 2 resalta las principales diferencias entre ambos entornos. Resumamos algunas de ellas:

- BOOLE-DEUSTO ofrece al alumno un entorno de diseño libre y no guiado (calculadora booleana), mientras que LogicAid establece una secuencia forzosa, es decir, después de un método solo puede venir otro ya definido. BOOLE-DEUSTO es superior en estrategia didáctica.
- BOOLE-DEUSTO es más amigable en su uso que el LogicAid. BOOLE-DEUSTO utiliza gran cantidad de formatos gráficos, mientras que LogicAid utiliza más el modo texto.
- BOOLE-DEUSTO permite introducir cualquier tipo de expresión booleana; LogicAid, no.
- BOOLE-DEUSTO es superior en el uso de diagramas V-K: más cómodos y sin límite de número de variables.
- BOOLE-DEUSTO permite obtener expresiones NAND/NOR; LogicAid, no.
- BOOLE-DEUSTO permite obtener circuitos lógicos; LogicAid, no.
- BOOLE-DEUSTO permite la simulación gráfica de autómatas; LogicAid, no.
- BOOLE-DEUSTO ofrece un mayor número de métodos para autómatas que el LogicAid.
- LogicAid tiene implementado un mejor método de simplificación booleana que el BOOLE-DEUSTO.
- BOOLE-DEUSTO permite la minimización de autómatas incompletamente especificados; LogicAid, no.
- LogicAid permite transiciones alfanuméricas en los autómatas; BOOLE-DEUSTO, no.
- BOOLE permite la exportación del sistema a OrCAD-PLD, VHDL y JEDEC, mientras que LogicAid solo exporta a JEDEC.
- LogicAid va acompañado del SimUaid que permite la simulación de circuitos lógicos, aspecto en absoluto cubierto por el BOOLE.
- BOOLE-DEUSTO es gratis y de instalación directa, LogicAid necesita de licencia y no está clara su gratuidad.

Generalizando, BOOLE-DEUSTO es superior en métodos, enfoque didáctico y uso al LogicAid desarrollado en EE.UU.

Característica	BOOLE-DEUSTO	LogicAid
Estrategia general	Calculadora booleana	Estructura fija
Uso	Modo gráfico	Modo texto
Métodos	Muchos	Pocos
Algoritmos internos	Normal	Excelente
Formatos de exportación	JEDEC, PLD, VHDL	JEDEC
Nivel de complejidad	A nivel de bit	A nivel de bit
Instalación	Gratis y directa	Con licencia

Tabla 2. Resumen de la comparación entre BOOLE-DEUSTO y LogicAid

Tema	BOOLE-DEUSTO
Álgebra de Boole	Comprobar Teoremas por inducción perfecta
Manipulación básica booleana	Obtención de la tabla de verdad de una función booleana Obtención del circuito lógico de una función booleana tipo SOP, POS, NAND o NOR Obtención de las formas normales de una tabla de verdad Obtención de un diagrama de Veitch-Karnaugh Simplificación de un diagrama de V-K Comprobación de una simplificación del alumno No obtiene la tabla de verdad de un circuito lógico. No simula
Diseño de sistemas combinacionales a nivel de bit	Diseño siguiendo los pasos: determinar e/s, tabla de verdad, formas normales, diagramas de V-K, simplificación e implementación Obtención de código para PLD: VHDL, OrCAD-PLD Generación del fichero JEDEC Simplificación avanzada: mayor número de métodos (heurísticos y no heurísticos) y estimación de coste
Diseño de autómatas	Captura gráfica de un DTE de Moore o Mealy Comprobación del autómata Conversión Moore-Mealy Minimización de estados Simulación gráfica del autómata Obtención de las tablas de diseño Obtención del circuito lógico con f-f tipo J-K o D No obtiene el DTE de un circuito lógico

Tabla 3. Utilización de BOOLE-DEUSTO en Diseño Lógico

4. Desarrollo y experiencia en el aula

Los primeros desarrollos del BOOLE-DEUSTO son del año 1995, la primera versión del 2001 [4, 5] y la última de 2005 [2]. En este largo periodo BOOLE-DEUSTO ha ido incorporando nuevas habilidades booleanas, hasta llegar al punto actual en el que se da por concluido. La razón para dar por terminado el BOOLE-DEUSTO es que las últimas modificaciones e incorporaciones tenían como referente las necesidades profesionales, o por lo menos buscaban que el BOOLE-DEUSTO pudiera ser usado para implementar sistemas: VHDL, generación de JEDEC, etc. Esta evolución saca al BOOLE-DEUSTO de su hábitat natural: el aula, y por tanto no tiene interés académico seguir con su desarrollo. Solo aparecerá una última versión de BOOLE-DEUSTO, aquella que recoja las comentarios de los usuarios a la última versión.

BOOLE-DEUSTO es usado en la Facultad de Ingeniería de la Universidad de Deusto en las titulaciones de Ingeniería Informática e Ingeniería en Electrónica y en Telecomunicaciones. Más específicamente las asignaturas son: Electrónica Digital y Tecnología de Computadores. Ambas asignaturas son del primer curso, la primera del

primer semestre, y la segunda, del segundo. Cerca de 600 alumnos lo usan por año, 300 en informática y 300 en electrónica. Los alumnos pueden obtener el BOOLE-DEUSTO de la página WEB del profesor [6], y actualmente se les entrega gratuitamente con el libro de ejercicios del curso por deferencia del Servicio de Publicaciones de la UD.

La didáctica del uso de BOOLE-DEUSTO está reflejada en la tabla 3 y en [7], y bien puede ser la siguiente. El profesor muestra la aplicación de un método, hace un par de ejemplos y manda completar a los alumnos una serie de ellos. El BOOLE-DEUSTO sirve al alumno en este último paso para comprobar si lo obtenido por él es correcto. Lo que no tiene sentido es utilizar el BOOLE-DEUSTO para obtener directamente los resultados. Es como cuando aprendimos a dividir y teníamos una calculadora al lado.

Para el profesor el BOOLE-DEUSTO cumple una doble función. Primero le libera de ser el referente en la aplicación de un método, este es un trabajo que puede resultar muy cansado y aburrido, ya que el alumno está aprendiendo desde cero. Pero hay un segundo uso, el profesor puede usar BOOLE-DEUSTO para plantear y resolver nuevos ejercicios, a él no le hace falta resolver ejercicios "a mano" para seguir aprendiendo. BOOLE-DEUSTO le permite copiar, imprimir y guardar los ejercicios resueltos.

¿Ayuda BOOLE-DEUSTO a la implantación del nuevo modelo universitario alentado por la declaración de Bolonia? Claramente la respuesta es sí, ya que fomenta el aprendizaje autónomo del alumno y ayuda al profesor y al alumno a trabajar de forma separada. Actualmente parece claro que se van a reducir las horas de aula -aunque no las de estudio y los contenidos-; en este ámbito el BOOLE-DEUSTO se puede convertir en el referente o conexión entre los alumnos, el profesor y los contenidos o métodos de electrónica digital. Este aspecto es reconocido como fundamental por muchos de los profesores que usan BOOLE-DEUSTO: les descarga de trabajo, ayuda a los alumnos y no les causa problemas.

Queda una pregunta por responder ¿qué valoración tiene el alumno del BOOLE-DEUSTO? En el curso 2003-2004 pasamos a los alumnos una encuesta muy sencilla para conocer su opinión sobre el BOOLE-DEUSTO y para introducir las nuevas modificaciones. El resultado se muestra en la tabla 4, y tiene al 1 y al 5 como valores mínimo y máximo, respectivamente.

Pregunta	Resultado
Valor de BOOLE-DEUSTO como herramienta de aprendizaje	
¿Es BOOLE-DEUSTO didactico/pedagógico?	3,9
¿Es BOOLE-DEUSTO útil para documentar ejercicios?	3,5
¿Cubre BOOLE-DEUSTO la totalidad de tus necesidades como estudiante?	3,5
¿Se debería usar sw del tipo BOOLE-DEUSTO en el aula?	3,9
Características	
¿Es atractivo BOOLE-DEUSTO?	3,2
¿Es BOOLE-DEUSTO fácil de usar?	4,1
¿Es BOOLE-DEUSTO efectivo a la hora de mostrar los resultados?	3,5
¿Son correctos los resultados que genera BOOLE-DEUSTO?	4,5
¿Es BOOLE-DEUSTO realmente fácil de instalar?	4,8
Relevancia de BOOLE-DEUSTO en el curso	
¿Te ha sido útil BOOLE-DEUSTO en el desarrollo del curso?	2,6
¿Te ha sido útil BOOLE-DEUSTO para aprender a simplificar?	3,4
¿Te ha sido útil BOOLE-DEUSTO en el diseño de autómatas?	3
¿Has usado BOOLE-DEUSTO con regularidad este curso?	2,2

Tabla 4. Opinión de los alumnos sobre BOOLE-DEUSTO

En la tabla 4 destacan dos aspectos:

- Los alumnos estiman la utilidad, calidad y facilidad de la herramienta BOOLE-DEUSTO. Es decir, la herramienta es útil.
- Los alumnos no usan BOOLE-DEUSTO todo lo que deberían. Es decir, el profesor debe hacer más por situar al BOOLE-DEUSTO en el desarrollo del curso.

Por último, en cuanto a experiencia en el aula es significativo constatar la divulgación nacional e internacional de BOOLE-DEUSTO:

- BOOLE-DEUSTO es usado en muchas universidades españolas, incluso en facultades que no son de ingeniería.
- La entrada BOOLE-DEUSTO en Google ofrece cerca de 80 páginas.
- La última versión fue descargada varios centenares de veces en la dos primeras semanas.
- Algunas universidades extranjeras están usando BOOLE-DEUSTO, sobre todo iberoamericanas.
- La divulgación y uso de BOOLE-DEUSTO aumenta constantemente.
- La lista de distribución de nuevas versiones (a la que cualquiera se puede apuntar enviando un correo al autor) tiene más de 100 entradas.
- Y quizá la más significativa: en cuanto alguien conoce BOOLE-DEUSTO empieza a usarlo y lo recomienda.

5. Conclusiones y futuras líneas de trabajo

La conclusión evidente es que BOOLE-DEUSTO es potente, adecuado, didáctico, cómodo, gratuito y fácil de usar. Lo anterior queda refrendado por la amplia difusión y aceptación que tiene. Además, sus características técnicas son comparables, si no claramente superiores, a las del entorno LogicAid del profesor Roth de EE.UU.

En cuanto a las futuras líneas, estas van más por la utilidad que por el desarrollo. Los autores de BOOLE-DEUSTO hemos decidido no ampliar más el BOOLE-DEUSTO. Solo habrá una nueva versión para intentar eliminar algunos pequeños fallos. Es mucho más interesante trabajar en cómo utilizar BOOLE-DEUSTO en el aula, en cómo usarlo para acercarse a Bolonia, en cómo implicar al alumno con el BOOLE-DEUSTO, en cómo combinar BOOLE-DEUSTO con otros entornos para dar cobertura a toda la asignatura, etc. Muchos profesores ya tienen claro que mucho del esfuerzo que se avecina no se va a centrar en diseñar nuevos contenidos o herramientas, sino en reordenar ese material atendiendo al nuevo modelo de aprendizaje.

Agradecimientos

BOOLE-DEUSTO ha sido completado dentro del proyecto LOGBOT financiado por el Dpto. de Industria, Comercio y Turismo del Gobierno Vasco, SAIOTEK 2002, OD02UD05. Además la publicación definitiva del BOOLE-DEUSTO por la Universidad de Deusto ha contado con una subvención del Gobierno Vasco para la publicación de materiales didácticos bilingües.

Referencias

[1] Sicard, E. http//intrage.insa-toulouse.fr/~etienne /microwind/

[2] García Zubía, J., Sanz Martínez, J. y Sotomayor Basilio, B. *BOOLE-DEUSTO v2.1* Ed. Universidad de Deusto, 2005

[3] Roth, Ch. *User's guide and reference manual for LogicAid™ Second Edition and getting started with SimUaid™*. Brooks/Cole-Thomson Learning, Canada, 2002.

[4] García Zubía, J. "Metodología y necesidades computacionales en sistemas digitales". *Actas de las VII JENUI*, Palma de Mallorca, 2001.

[5] García Zubía, J., Sanz Martínez, J. y Sotomayor, B. "BOOLE-DEUSTO, la aplicación para sistemas digitales". *Actas de VII JENUI*, Palma de Mallorca, 2001.

[6] http://paginaspersonales.deusto.es/zubia

[7] García Zubía, J. *Problemas resueltos de electrónica digital*. Ed. Thomson-Paraninfo, 2004.

Uso de simuladores y herramientas Web para la enseñanza de Sistemas Operativos

Félix Buendía, Juan-Carlos Cano, Julio Sahuquillo
Departamento de Informática de Sistemas y Computadores
Escuela Técnica Superior de Informática Aplicada
Universidad Politécnica de Valencia
e-mail: {fbuendia, jucano, jsahuqui}@disca.upv.es

Resumen

La enseñanza de las materias relacionadas con los Sistemas Operativos se organiza en varias asignaturas que cubren un conjunto de contenidos esenciales en cualquier currículo universitario de informática. Las diferentes asignaturas impartidas incluyen temas con una fuerte componente teórica y conceptual. Aunque dichos conceptos son ampliamente tratados en numerosos libros de texto, no resulta sencillo encontrar entornos adecuados al nivel de conocimientos del alumno, que permitan realizar actividades donde poner en práctica los conceptos introducidos de forma teórica. En este trabajo se describen dos herramientas que permiten la puesta en práctica de dichos conceptos a partir del uso de aplicaciones Web. La primera representa un simulador utilizado para mostrar el funcionamiento de los principales aspectos relacionados con la gestión de memoria en un sistema operativo, de forma que el alumno pueda interactuar con el simulador, introduciendo ejemplos de carga y configurando parámetros del sistema. La segunda herramienta permite el acceso remoto a funciones de un sistema operativo real a través de un interfaz Web. La experiencia docente con ambas herramientas corrobora su utilidad tanto en aspectos relacionados con la comprensión del alumno como en la evaluación por parte del profesor.

1. Introducción

Las asignaturas relacionadas con los Sistemas Operativos son un elemento fundamental en cualquier currículo universitario de informática. Desde las propuestas de ACM/IEEE de 1991 [1] hasta las más recientes del CC 2001 [2], existe un consenso sobre la importancia de esta temática. Las directrices del MEC establecen una serie de descriptores que hacen referencia a aspectos tales como la gestión de procesos y ficheros así como los recursos de memoria y entrada/salida. Se trata de aspectos con una fuerte componente teórica y que deben encontrarse apoyados con actividades prácticas para facilitar su comprensión. Sin embargo, no resulta sencillo encontrar entornos de trabajo donde realizar este tipo de actividades que permitan al alumno fijar con la práctica los conceptos tratados de forma teórica. Por otro lado, las actividades prácticas se han realizado tradicionalmente sobre los sistemas operativos reales donde el acceso a los mecanismos internos puede resultar excesivamente complejo.

En algunos casos, se dispone de sistemas operativos reducidos como Minix [3] y Nachos [4], que aunque presentan una amplia funcionalidad, para asignaturas troncales básicas tienen como principal inconveniente su orientación hacia aspectos de diseño e implementación.

Otras herramientas como RCOS [5] o SOsim [6] simulan el funcionamiento de aspectos de un sistema operativo. Sin embargo, la propuesta de RCOS se centra demasiado en los aspectos visuales y no profundiza en la aplicación de aspectos teóricos. Magee y Kramer [7] presentan una herramienta que, de manera similar a la que se presenta en esta ponencia, se centra en la demostración visual de aspectos relacionados con los conceptos de concurrencia.

El uso de simuladores puede resultar útil para fijar los conceptos teóricos mediante actividades prácticas en asignaturas básicas. Sin embargo, su utilización no significa que haya que renunciar al trabajo con sistemas operativos reales. Una de las capacidades prácticas que recomienda el CC 2001

es la de "operar con equipamiento y software de sistema de manera eficiente". Los sistemas operativos suelen incluir asistentes que muestran al usuario su funcionamiento o guían a éste en su gestión. Sin embargo, existen pocas herramientas que permitan el trabajo práctico con sistemas operativos reales utilizando un enfoque didáctico. Las herramientas que se describen en este trabajo representan dos visiones de actividades prácticas relacionadas con el manejo de sistemas operativos y que pueden complementarse para facilitar una docencia más eficaz en esta disciplina.

El resto del trabajo se organiza como sigue. En la Sección 2, se presenta el simulador de la gestión de memoria de un sistema operativo accesible vía Web. La Sección 3 describe herramientas basadas en aplicaciones Web que permiten el acceso a funciones del sistema operativo de una forma didáctica. La Sección 4 refleja la valoración sobre el uso docente de estas herramientas y su papel en las nuevas metodologías que se avecinan con la integración en el espacio europeo de educación superior (EEES). Finalmente, la Sección 5 presenta las conclusiones del trabajo.

2. Simulador de Gestión de Memoria

La herramienta de simulación ha sido desarrollada en lenguaje Java. Una primera versión de la misma se desarrolló en el contexto de un proyecto final de carrera dentro de la Escuela de Informática de la Universidad Politécnica de Valencia [8]. Nuevas versiones con interfaz mejorada se han ido incorporando como parte del trabajo realizado en el ámbito de un proyecto europeo de educación a distancia [9].

2.1. Introducción al simulador

El Simulador de gestión de memoria tiene por objeto mostrar el funcionamiento de algunos aspectos de la gestión de memoria que intervienen en la ejecución de uno o más procesos en un entorno de multiprogramación. Concretamente, la herramienta describe la ejecución de un conjunto de procesos mediante el algoritmo de planificación *round-robin* y cómo estos procesos generan durante su ejecución accesos a direcciones lógicas que serán traducidas a direcciones en memoria física. El esquema de traducción de direcciones se basa en una técnica de paginación combinada con algoritmos de reemplazo, lo que se conoce globalmente como memoria virtual mediante paginación bajo demanda.

2.2. Inicio de una sesión

El primer paso en la ejecución de una simulación de gestión de memoria consiste en cargar el fichero de simulación (menú Archivo). Para describir un determinado programa, se ofrece un fichero denominado "Simulacion.inf" que describe las características de los diferentes procesos. Este fichero contiene una secuencia de direcciones lógicas agrupadas por cada proceso que se incluye en el sistema. Para ello se utiliza una cabecera con tres campos: el primero hace referencia al identificador del proceso, el segundo el tamaño asignado a dicho proceso y el tercero a la prioridad, que identificará la cola a la que pertenece el proceso. La Figura 1 muestra un ejemplo.

```
Proceso1 50000 0
5000
0
100
0
456
4000
1050
end
0…
```

Figura 1. Ejemplo de fichero Simulacion.inf.

En dicho ejemplo la primera línea del fichero contiene la siguiente declaración:

```
Proceso1 50000 0
```

que define el Proceso1, con un tamaño de 50000 bytes y una prioridad 0. A partir de esta línea aparecen valores numéricos separados en diferentes líneas, cada uno de los cuales representa una dirección lógica. Las direcciones lógicas contenidas en un mismo proceso finalizan cuando aparece la etiqueta "end". A continuación, y antes de empezar la especificación de un nuevo proceso, aparece un valor numérico que especifica el desplazamiento temporal en la activación del siguiente proceso respecto al anterior. En este caso, se utiliza un valor 0 que indica que el Proceso1 y el siguiente (Proceso2) se activan al mismo tiempo. Un valor de 2 significaría que

deben transcurrir dos instantes de tiempo antes de que se active el Proceso2.

2.3. Configuración del sistema

Una vez que se ha cargado el fichero de simulación, se debe proceder a la configuración del sistema mediante la opción de dicho nombre del menú Archivo. La Figura 2 muestra un ejemplo. Se puede observar cómo esta opción permite asignar diferentes atributos del sistema tales como el tamaño de la memoria, el tamaño de la página, los algoritmos de asignación y reemplazo del sistema.

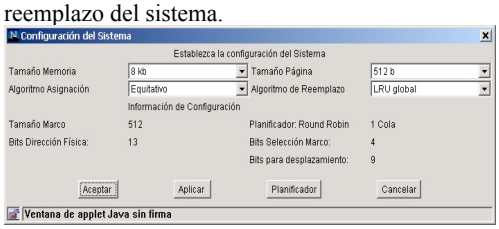


Figura 2. Ventana de configuración del sistema.

2.4. Comienzo de la simulación

Una vez cargado el fichero de simulación y realizada la configuración del sistema, para simular el comportamiento se elige la opción "Comenzar Simulación" dentro del menú Archivo.

Una vez iniciada la simulación, se puede observar la evolución de la misma a través de diferentes zonas de la pantalla mostradas en la Figura 3. Existen dos formas de controlar la simulación, las cuales se pueden seleccionar desde el menú de Simulación:

- Modo paso a paso: permite visualizar los resultados de cada paso (unidad temporal) en que se realiza cada acceso a memoria o la activación de un nuevo proceso.
- Modo automático: la ejecución de los procesos y, por tanto, el acceso a direcciones de memoria se realiza de forma automática sin intervención del usuario.

La parte superior izquierda de la pantalla muestra una representación gráfica de los procesos activos en cada cola. En el ejemplo se utiliza una única cola (Cola 0) que contiene los tres procesos definidos. Además, cada vez que se accede a la dirección lógica de un proceso, éste se pondrá al inicio de la cola y cambiará el color del símbolo que lo representa, para indicar que es el proceso en ejecución.

La parte superior derecha de la pantalla informa del estado del proceso en ejecución. Para ello, se incluye el contador de direcciones lógicas, el número de accesos realizados hasta el momento actual y el valor de la tasa de fallos de página.

La parte inferior izquierda representa la gestión de páginas e incluye información relativa al algoritmo de reemplazo seleccionado, el tipo de asignación de marcos si la hubiera y la lista de páginas utilizadas por cada proceso y sometidas al correspondiente algoritmo de reemplazo.

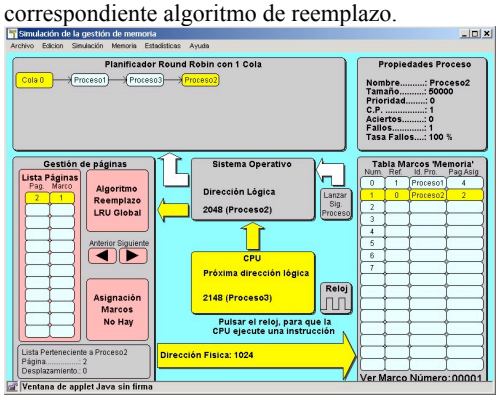


Figura 3. Ejemplo de ejecución del simulador.

La parte inferior derecha de la pantalla representa la Tabla de Marcos de la memoria física. En ella se muestra la siguiente información:

- El número de marco: su valor va desde 0 hasta N, siendo N+1 el número de marcos disponible.
- El campo referencia define la antigüedad de las páginas que se almacenan en memoria.
- El identificador de proceso: indica el proceso que accede a la página.
- Página asignada: se trata de la página almacenada en el marco referido.

Por último, la parte central de la pantalla define la relación entre las estructuras descritas previamente. Para ello, se indica en cada momento qué proceso ha sido elegido por parte del planificador para su ejecución y la dirección lógica implicada en este paso. También se encuentran en esta parte de la pantalla tres etiquetas que contienen la siguiente información:

- Sistema operativo: indica la dirección lógica que actualmente está siendo traducida.
- CPU: representa la próxima dirección a traducir.
- Dirección física: contiene la dirección física resultado de la traducción actual.

2.5. Resultados de la simulación

Mediante la opción "Ver Ventana de Direcciones Físicas" del menú Edición, se obtiene una traza de la simulación. Esta traza muestra un listado de direcciones físicas como consecuencia del proceso de traducción.

Tras finalizar la simulación y descargar los procesos de la memoria, se pueden obtener estadísticas de la ejecución de la simulación. La Figura 4 muestra un ejemplo de dichos resultados, los cuales se pueden obtener utilizando las opciones "Tasa de Fallos" y "Rendimiento" dentro del menú "Estadísticas".

```
**********************************************
Tasa de Fallos
**********************************************
El Proceso Proceso3 ha efectuado 7 accesos a memoria
obteniendo 1 aciertos y 6 fallos de página
Con una tasa de fallos del  85  %

El Proceso Proceso2 ha efectuado 7 accesos a memoria
obteniendo 2 aciertos y 5 fallos de página
Con una tasa de fallos del  71  %

El Proceso Proceso1 ha efectuado 7 accesos a memoria
obteniendo 3 aciertos y 4 fallos de página
Con una tasa de fallos del  57  %
**********************************************
Rendimiento del sistema
**********************************************
Se han producido un total de 21 accesos a memoria
contabilizando un total de 6 aciertos de página y
15 fallos de página. Con una tasa de fallos del  71  %
```

Figura 4. Estadísticas de la simulación.

3. Herramienta de acceso al sistema operativo

Esta sección describe una herramienta basada en aplicaciones Web que permiten el acceso a los servicios de un sistema operativo de una manera supervisada y didáctica. Los sistemas operativos raramente utilizan una interfaz Web como medio de acceso remoto y se emplean otros mecanismos como conexiones *telnet* o *rsh* en entornos UNIX o servicios de Terminal Server en Windows. Existen productos como HTSH [10] o PHPShell [11] que permiten la ejecución de programas shell a través de la Web, o WSH [12] dirigido a ejecutar programas en una máquina Windows remota pero sin una utilidad didáctica específica.

3.1. Antecedentes: RPEWA

La herramienta descrita se basa en una aplicación denominada RPEWA (Remote Program Execution Web Application) cuyo principal objetivo consistía en permitir a estudiantes de sistemas operativos la carga y comprobación de programas escritos en lenguaje *C* o *shellscript*. Esta aplicación se desarrolló como un proyecto final de carrera y su esquema se muestra en la Figura 5.

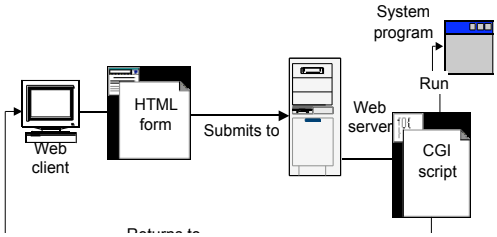


Figura 5. Esquema de RPEWA.

RPEWA funciona como sigue. En primer lugar, el cliente Web (p.e. un navegador) accede a una dirección (URL) que devuelve un formulario HTML que permite al alumno introducir el código del programa y sus parámetros. Una vez introducida esta información en el formulario se envía al servidor Web que a su vez la procesa mediante un *script* CGI programado en lenguaje Perl. Finalmente dicho *script* ejecuta el programa transferido y devuelve su resultado al cliente Web.

Esta aplicación funciona de forma autónoma, lo que le proporciona una gran flexibilidad a la hora de poder introducir ejemplos de código y realizar la comprobación de su funcionamiento. El hecho de poder utilizarse desde un navegador Web facilita su acceso universal y la posibilidad de utilizar servicios del sistema operativo no disponibles en la máquina cliente. Además proporciona al alumno una representación de las salidas estándar y de error de los programas ejecutados, así como la opción de guardar dichos

programas y recuperarlos en futuras sesiones de trabajo. Esta herramienta es un claro ejemplo del uso de aplicaciones Web para facilitar el acceso a servicios de un sistema operativo y, por tanto, de contribuir a su aprendizaje. Sin embargo, se detectaron ciertas carencias es su concepción que dieron lugar a una nueva herramienta que se describe en la siguiente sección.

3.2. RPEWA 2

Una de las carencias detectadas consistía en la falta de relación entre las actividades realizadas mediante RPEWA y los contenidos teóricos donde se podían enmarcar dichas actividades. Otra de las cuestiones pendientes en RPEWA era la ausencia de un entorno desde el que se pudiese controlar o supervisar el uso docente de esta herramienta.

Para solucionar estas carencias se integraron los mecanismo de RPEWA en un entorno de campus virtual denominado *Xedu* [14] dando como resultado una nueva versión denominada RPEWA2. Dicho entorno permite registrar y controlar el acceso de los diferentes usuarios a los contenidos y actividades de una determinada asignatura. Además se incorporó el uso de una entidad denominada *Unidad de Aprendizaje* (Learning Unit) [15] que permitía integrar los contenidos teóricos y las actividades prácticas para su mejor aprovechamiento.

Una *Unidad de Aprendizaje* consta de una serie de elementos para representar recursos didácticos expresados mediante un lenguaje de marcado denominado LMML (Learning Material Mark-up Language) [16]. Dichos elementos estructuran la información de tipo teórico a través de etiquetas como *paragraph* y los ejercicios o actividades prácticas (etiqueta *exercise*)

La Figura 6 muestra parte de un documento LMML asociado a una *Unidad de Aprendizaje* que se utiliza para la enseñanza de Sistemas Operativos en el apartado de gestión de procesos. En este caso se trata de explicar la finalidad de un intérprete de órdenes en entornos UNIX, y cómo éste se encarga de ejecutar órdenes simples como *ls* o *who*. Con el fin de reforzar la adquisición de estos conceptos teóricos se propone la realización de un ejercicio (elemento *exercise*) en el que el alumno tiene que elaborar un programa en lenguaje C que se encargue de interpretar las ordenes mencionadas.

```
<lmml>
    <collection title="Intérprete de órdenes sencillo (0)"
difficulty="low">
        <motivation>
            <LMMLtext>Se trata de diseñar y
codificar, en lenguaje C y sobre un sistema operativo
Linux, un programa que actúe como intérprete de
órdenes sencillo</LMMLtext>
        </motivation>
        <section title="Ejecución de órdenes
simples">
            <paragraph>
                <LMMLtext>En este caso, se
pretende ejecutar órdenes simples como
<emphasized>ls</emphasized> o <emphasized>
                who</emphasized></LMMLtext>
                </paragraph>
                <exercise>
                    <LMMLtext>Generar una versión de
ush interactiva con el nombre ush0.c y cuya sintaxis de
llamada sea: </LMMLtext>
                    <code><![CDATA[ush
<orden>]]></code>
                    <LMMLcode type="input" style="file"
uri="ush0.c"/>
                </exercise>
                <remark>
                    <LMMLtext>Se trata de utilizar la
llamada a sistema <emphasized>exec</emphasized>
tal como se explica en</LMMLtext>
                    <LMMLcode type="output"
uri="http://sop.upv.es/so2/pdf/UT1.pdf" title="Tema 1
(SO2)"/>
                </remark>…
```

Figura 6. Ejemplo de Unidad de Aprendizaje.

La Figura 7 muestra un esquema de la aplicación RPEWA2 que se encarga de procesar los documentos asociados a las *Unidades de Aprendizaje* dentro del entorno *Xedu*. El funcionamiento también se produce desde un cliente Web que accede a una URL implementada mediante un documento XML situado en un servidor Web bajo *Xedu*.

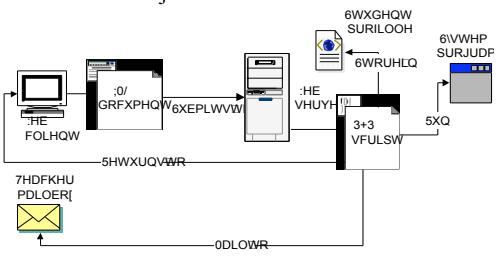


Figura 7. Esquema de RPEWA 2.

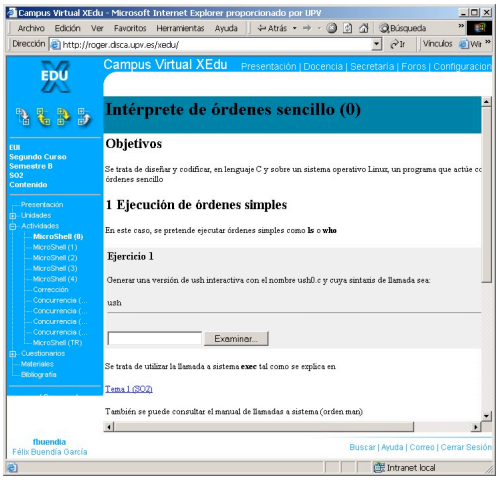


Figura 8. Ejemplo de presentación de la Unidad de Aprendizaje.

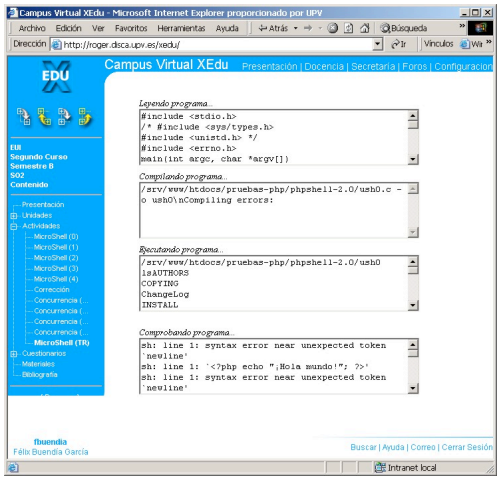


Figura 9. Ejemplo de ejecución de programa mediante RPEWA2.

La Figura 8 representa una *Unidad de Aprendizaje* que es interpretada dentro del servidor Web mediante código PHP. Cada una de las unidades de aprendizaje lleva asociadas las siguientes acciones.

1. Presentación de una página HTML que contiene explicaciones teóricas procedentes de los elementos *paragraph* o formularios para la realización de actividades asociadas a los elementos *exercise*.
2. Actualización de los resultados obtenidos por el alumno en su perfil de estudiante. Dichos resultados pueden consistir en respuestas a cuestiones planteadas en la actividad o en la elaboración de programas como respuesta a los ejercicios propuestos. En este caso se proporciona la posibilidad de comprobar el funcionamiento de dichos programas, tal como se muestra en la Figura 9.
3. Notificación al profesor por correo electrónico de la realización de las actividades propuestas en la Unidad de Aprendizaje. De esta forma se pueden aplicar acciones de supervisión y evaluación tales como acceder a las respuestas sobre las actividades planteadas en la Unidad de Aprendizaje, así como obtener los archivos que representan los programas desarrollados por el alumno. Así, el profesor puede supervisar mediante una pantalla de la aplicación el trabajo del alumno.

4. Valoración sobre el uso docente de las herramientas

Las herramientas descritas previamente reflejan la contribución de tecnologías como simuladores y aplicaciones Web para facilitar la docencia en una disciplina fundamental en los *curricula* de Informática como son el estudio de los Sistemas Operativos.

La primera herramienta, consistente en un simulador de memoria virtual, se utilizó hace unos años en una asignatura de segundo curso. En dicha asignatura se impartían conceptos básicos de Sistemas Operativos tales como la gestión de procesos o de memoria, y el objetivo del simulador consistía en facilitar la comprensión de estos conceptos. También servía de apoyo para la realización de actividades de diseño consistentes en que el alumno elaborara un simulador propio aunque sin componentes gráficos. La experiencia de uso del simulador fue positiva y uno de los aspectos más valorados consistió en la posibilidad de visualizar el funcionamiento de algoritmos, tanto relacionados con la planificación de procesos como con la traducción de direcciones y asignación de espacio en memoria. No obstante, con la llegada de los nuevos planes de estudio en las titulaciones de informática de la Escuela de Informática se produjo una compresión de los contenidos básicos que pasaron de una duración

anual a una semestral. Ello significó una reducción del tiempo dedicado para el estudio de aspectos teóricos y asimismo de las actividades, como el uso del simulador, orientadas a facilitar su comprensión. Por tanto, dicha actividad quedó fuera de la planificación de prácticas y simplemente se dejó como trabajo opcional.

Como consecuencia de este cambio de orientación en los planes de estudio se dio cabida a una nueva asignatura de tipo semestral basada en el estudio de casos y de corte mucho más práctico. En este contexto se introdujeron actividades relacionadas con las herramientas que se describen en la sección 3. También se produjo una novedad en la forma de impartir algunas asignaturas y que consistía en la introducción de grupos denominados de "tele-enseñanza" (TEL) cuya finalidad era la de flexibilizar su docencia. Ambas cuestiones han contribuido a potenciar una serie de herramientas que aprovechan las posibilidades de la Web para acceder a las funciones y mecanismos de un sistema operativo.

Por ejemplo, en la asignatura de Sistemas Operativos se pasó de un grupo TEL con menos de cinco alumnos en el curso 2002-2003 a 29 alumnos en el curso 2003-2004 y 50 alumnos en el curso actual. Esta aumento ha requerido el desarrollo de las herramientas utilizadas en dicho grupo, desde aplicaciones sencillas como RPEWA donde se produce una mera comprobación de programas C a través de una interfaz Web hasta la integración de dichas aplicaciones en un entorno de Campus Virtual como se manifiesta en la versión RPEWA2. El incremento del número de alumnos también ha exigido la incorporación de nuevas herramientas que faciliten la supervisión y control por parte del profesor.

El resultado de las experiencias llevadas a cabo durante el curso pasado junto con las ya iniciadas en el actual, ponen de manifiesto las siguientes cuestiones:

- Preparación: los alumnos no están acostumbrados a tomar iniciativas en el uso de herramientas que den soporte a la resolución de problemas que se plantean en las actividades prácticas. Habitualmente, ellos esperan un guión que les dirija la actividad a realizar.
- Conocimientos previos: se han detectado graves carencias en aquellos conocimientos (principalmente de programación) que son requeridos para un aprovechamiento de las herramientas proporcionadas.
- Conexión con la teoría: en algunos casos se ha observado una dificultad en conectar el trabajo realizado a partir de las herramientas con los conceptos teóricos que subyacen a las actividades prácticas puestas en marcha.
- Evaluación: resulta compleja la interpretación de las actividades realizadas a partir del uso de las herramientas y su traducción a una nota que evalúe el trabajo del alumno. También se detectan casos de "plagiarismo" y resulta difícil reconocer el trabajo individual.

Todas estas cuestiones se han trasladado a un grupo de innovación docente que se ha creado recientemente en la Universidad Politécnica de Valencia (UPV) donde se debaten aspectos de aplicación de las nuevas tecnologías a la docencia. Dicho grupo de innovación se denomina NTA^2 [18] y está formado por personal docente de varios departamentos, interesados en las Tecnologías de la Información y la Comunicación (TIC) y sus aplicaciones docentes. Esta iniciativa forma parte de los llamados Proyectos de Adaptación al Espacio Europeo de Educación Superior (PAEEES) dentro de la UPV y está integrada por diversas acciones que van desde los sistemas de evaluación, la utilización de herramientas de campus virtual y trabajo colaborativo, el uso de laboratorios virtuales/remotos hasta el aprendizaje basado en proyectos. En este contexto se pretende analizar el uso de las herramientas descritas en el presente trabajo y evaluar sus futuras aplicaciones para fomentar el aprendizaje activo del alumno según las directrices del nuevo Espacio Europeo de Educación Superior.

5. Conclusiones

En este trabajo se han descrito dos ejemplos de herramientas que representan dos formas de enfocar las actividades prácticas en el ámbito docente de los sistemas operativos. También se ha indicado el contexto donde fueron utilizadas y las valoraciones de dicho uso, haciendo hincapié en la oportunidad que supone su incorporación en el ámbito del Espacio Europeo de Educación Superior.

Sin embargo, se han detectado las dificultades propias del desarrollo y mantenimiento de

herramientas que exigen una gran cantidad de recursos y dedicación. Ello exige que las instituciones académicas apuesten por este tipo de herramientas y contribuyan a su soporte y gestión. Por otro lado, la utilización de estas herramientas requiere una adaptación de las metodologías empleadas en la docencia de las materias relacionadas. En el contexto de los Sistemas Operativos, esto significa conceder un mayor protagonismo al trabajo individual de alumno y que el profesor dedique más esfuerzo a tutelar y supervisar las actividades realizadas con dichas herramientas. También son necesarios nuevos procedimientos de evaluación que tengan en cuenta la realización de estas actividades.

Referencias

[1] A.B. Tucker et al., ACM/IEEE-CS Joint Curriculum Task Force. Computing Curricula 1991, ACM Press; IEEE Comp. Soc. Press. , 1991.

[2] ACM/IEEE Task Force on the Year 2001 Model Curricula for Computing , Computing Curricula 2001(CC-2001), (Online: http://www.computer.org/education/cc2001/).

[3] A. S. Tanenbaum and A. S. Woodhull, Operating Systems: Design and Implementation, 2ed., Prentice-Hall, 1997.

[4] W A Christopher et a. (1993), The Nachos Instructional Operating System. Proceedings of the Winter 1993 Usenix Technical Conference, pp 481-489.

[5] David Jones, Andrew Newman, A constructivist-based tools for operating systems education, Proceedings of EdMedia'2002, Denver, Colorado, Jun. 2002.

[6] L.P. Maia, A.C. Pacheco, A Simulator Supporting Lectures on Operating Systems. 33rd ASEE/IEEE Frontiers In Education Conference 2003 Boulder, Colorado.

[7] Jeff Magee and Jeff Kramer. Concurrency: State Models & Java Programs.

[8] Llopis Mengual, J.Espinosa Rodilla. Simulador de gestión de memoria. Proyecto final de carrera. Universidad Politécnica de Valencia, 1999.

[9] Innovations for Education in Information Technology through Multimedia and Communication Networks. Proyecto Socrates de Red Temática. (Online: http://www.eui.upv.es/ineit-mucon/).

[10] HTML Shell (Online: http://www.nmmm.nu/linux/sh/htsh.htm).

[11] M. Geisler. PHPShell (Online: http://www.gimpster.com/wiki/PhpShell).

[12] Windows Script Host (WSH) (Online: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/script56/html/wsconwshbasics.asp).

[13] J.M. Ruiz, J.F. Puchades. Programación de scripts Web. Proyecto final de carrera. Universidad Politécnica de Valencia, 2003.

[14] F. Buendía, M Agustí, J.V Benlloch., E. Bisbal, M. Lluesma, XEDU, an Open Learning Management System Proposal. International Conference on Network Universities and E-learning 2003, Valencia (Spain).

[15] F. Buendía, J.C. Burguillo, J.V Benlloch., D Rodríguez, J.M. Gómez, J.J. Vidal. Tools for Creation and Management of Didactic Resources in Electrical and Information Engineering. Computers and Education: Towards a Lifelong Learning Society, Kluwer Academic Publishers 2003.

[16] Süß,C., Kammerl, R., Freitag, B.A Teachware Management Framework for Multiple Teaching Strategies. In: Proceedings of ED-MEDIA 2000, World Conference on Educational Multimedia, Hypermedia & Telecommunications, Montreal, Quebec, 2000

[17] F.Buendía, J.V Benlloch, J.M. Gomez, Development of Didactic Resources for Distance Learning based on Simulation, Computers and Education Towards an Interconnected Society, Ed. Kluwer Academic Publishers 2001, pp. 93-101.

[18] F. Buendía et al. El Grupo de Innovación Docente NTA^2 "Nuevas Tecnologías para el Aprendizaje Activo". Enviado al Simposio Nacional de Tecnologías de la Información y las Comunicaciones en la Educación (Sintice 05).

Aprendizaje guiado con JBACI y tutorización mediante mensajería instantánea

Piñol,P., Martínez,M.O., López,O., Galiano,V., Migallón,H.
Dpto. Física y Arquitectura de Computadores
Universidad Miguel Hernández
03202 Elche
e-mail: {pablop,mmrach,otoniel,vgaliano,hmigallon}@umh.es

Resumen

En este artículo se pretende recoger la experiencia realizada en la asignatura 'Sistemas Operativos' de segundo curso de Ingeniería Técnica en Informática de Gestión, impartida en la Escuela Politécnica Superior de Orihuela de la Universidad Miguel Hernández. En concreto, la experiencia se ha realizado sobre la parte del temario de la asignatura que cubre los conceptos de concurrencia, exclusión mutua y sincronización de procesos.

La utilización de las nuevas tecnologías no garantiza unos mejores resultados en el aprendizaje de los alumnos. Lo que sí desempeña un papel crucial en la obtención de buenos resultados es, especialmente, la forma en la que la tecnología es puesta en práctica [13].

Con este ejemplo proponemos una metodología para la inclusión de las nuevas tecnologías en el aula y en las tutorías. Los objetivos principales son intentar garantizar un buen rendimiento en las prácticas de la asignatura así como obtener una mayor motivación del alumnado, una mejor comprensión de los conceptos teóricos y un uso planificado y guiado en el proceso de enseñanza-aprendizaje.

Para conseguir dichos objetivos se ha puesto en marcha, siguiendo la planificación de dicha metodología, el uso en el aula de una herramienta que simula el efecto de los *semáforos*, los *monitores* y la sincronización de procesos llamada JBACI (Java Ben-Ari Concurrent Interpreter)[15].

1. Introducción

Informes recientes han destacado la importancia de las tecnologías digitales para dar respuesta a los antiguos y recientes problemas de la educación [12], pero no siempre tienen suficientemente en cuenta la compleja realidad que la circunda. De este modo, es posible identificar dos grandes formas de concebir la utilización de estas tecnologías como factor de innovación educativa. La primera se caracteriza por centrar gran parte del interés y la atención en los sistemas informáticos. La segunda por situar la problemática de la mejora de la educación en el conjunto complejo de factores que configuran las situaciones de enseñanza y el aprendizaje que, o se transforman de forma paralela e interactiva, o inhiben la propia innovación.

Al inicio del curso se realizó una encuesta a los alumnos sobre los conocimientos previos necesarios para abordar sin dificultad ciertos aspectos prácticos de la asignatura. Esta encuesta mostró un perfil en el alumnado con conocimientos demasiado superficiales de programación en C, tanto en entornos tipo Unix (como puede ser Linux utilizando las normas POSIX) como en entornos Windows (usando el API Win32). También se descubrió que el porcentaje de alumnos que había instalado Linux en casa alguna vez era bajísimo y mucho menor el que lo utilizaba habitualmente.

Por otro lado, en las aulas disponibles para prácticas se encontraba instalado únicamente Windows. Además, cierta parte del alumnado estaba más interesado en aprender Linux co-

mo simples usuarios, aunque avanzados, que en 'perder tiempo' con la teoría de los Sistemas Operativos, ya que a su modo de ver 'nunca' se iban a encontrar con la necesidad de poner en práctica dichos conocimientos, bien como diseñadores o como programadores de sistemas. La desmotivación del alumnado estaba bien presente.

Nuestra intención inicial era desarrollar las prácticas de la asignatura relativas a concurrencia y sincronización de procesos en lenguaje C con las normas POSIX y pensamos que dicha aproximación es recomendable para esta asignatura si el perfil del alumnado lo permite. A parte de las carencias técnicas, se observó cierta pasividad en los alumnos respecto a la participación en el aula y la motivación en general; este comportamiento provoca la falta de 'feedback' hacia el profesor. El 'feedback' del alumno es necesario para el docente, pues gracias a él, el profesor detecta el nivel de aprendizaje que se produce en el aula y puede ajustar sus exposiciones.

Intuimos que siguiendo la aproximación docente basada en POSIX o Win32 seguramente no cumpliríamos con las expectativas cuantitativas y/o cualitativas que entendemos necesarias para estos temas. Supusimos que perderían más tiempo con las problemáticas típicas de la inexperiencia en el lenguaje de programación que en centrarse en la comprensión de dichos conceptos. Pretendíamos centrar su atención en la resolución del problema y no en la sintaxis. Igualmente pensamos que para que los conceptos quedaran suficientemente claros, los ejercicios que deberían hacer en las prácticas debían tener un nivel de complejidad medio-alto y esto era necesario mantenerlo.

Se trata de una asignatura cuatrimestral, con un contenido denso en conceptos que deben quedar bien afianzados en el alumno. Normalmente las prácticas relacionadas con los conceptos teóricos intentan profundizar en éstos a la vez que generan experiencia en el alumno. Lo ideal es realizar las prácticas de esta parte de la asignatura utilizando los mecanismos que un sistema operativo real pone a disposición del programador de sistemas (véase [10][2][3][4][14]). Pero como decimos, había que buscar una alternativa que asegurara, en cualquier caso, que el tiempo dedicado a llevarla a cabo fuese el mismo que el que se emplearía sin ella, pues el tiempo de cada bloque ya estaba fijado en la programación de la asignatura.

La alternativa que hemos encontrado se basa en el uso de JBACI, un simulador de concurrencia, ejecutable en distintos entornos y con una sintaxis muy sencilla y reducida. Es necesario adaptar los problemas típicos de concurrencia y sincronización a dicha herramienta. Hemos desarrollado una metodología para la docencia de los ejercicios y prácticas basándonos en dicha herramienta.

Debido a que el profesorado de la asignatura dista físicamente de los alumnos por el carácter multicampus de la Universidad Miguel Hernández, se hace necesario buscar alternativas para evitar desplazamientos a profesores y alumnos. Para evitar esto, el seguimiento, tutorización y soporte al grupo de alumnos se está realizando mediante la mensajería instantánea MSN Messenger de la que se utilizan sus características de compartición de aplicaciones, voz y ocasionalmente la pizarra.

2. Presentación de la Asignatura

La experiencia en concreto se ha realizado en la asignatura 'Sistemas Operativos'. Se trata de una asignatura cuatrimestral de 7,5 créditos distribuidos en 4,5 créditos teóricos y 3 créditos prácticos. Dicha asignatura se imparte en el primer cuatrimestre a razón de tres horas de teoría semanales y dos de prácticas. La experiencia debía garantizar no consumir más tiempo que el inicialmente programado (7,5 horas), distribuidas en cinco sesiones de hora y media, de las cuales 2 sesiones son de teoría y el resto de resolución de ejercicios. La experiencia se ha realizado sobre la parte del temario de la asignatura que cubre los conceptos de concurrencia, exclusión mutua y sincronización de procesos

3. Motivación metodológica

Nuestra propuesta de desarrollo de esta parte de la asignatura la basamos en las estrate-

gias para desarrollar expectativas positivas de aprendizaje descritas por J.L Castillejo en [5]:

- *Es necesario que el alumno tenga continuadas experiencias de éxito.* Para ello el profesor plantea problemas muy sencillos utilizando JBACI, pero con errores fácilmente detectables por el alumno, lo que provoca pequeños debates en el aula para resolverlos. Como efecto añadido, el alumno va asimilando la sintaxis de JBACI.
- *En el diseño de instrucción deben aparecer con especificidad y claridad los requisitos para lograr los objetivos propuestos.* Puesto que el objetivo era que realizaran prácticas complejas para los conceptos de exclusión mutua y sincronización, se presentaron las prácticas obligatorias que debían realizar.
- *Se eleva el nivel de expectativas en la medida que se permite un autocontrol de los resultados y se usan refuerzos de atribución que permitan conectar el éxito con su habilidad y esfuerzo personal.* Durante las sesiones teóricas en las que se presentan ejercicios a resolver se crean grupos de alumnos que deben solucionar dichos ejercicios; de esta forma se fomenta además de la competitividad, que en ocasiones puede ser beneficioso, la interrelación grupal en el aula y el aprendizaje cooperativo.

Para potenciar las estrategias expuestas anteriormente, nos hemos basado en algunos elementos didácticos que sugieren C. Coll y otros en [6] y, J. Gimeno en [9]. Estos autores argumentan que el diseño y la planificación de la enseñanza deberían prestar atención simultáneamente a cuatro dimensiones:

- *Los contenidos a enseñar.* Son los contenidos propios del temario de esta parte de la asignatura como son la exclusión mutua y sincronización de procesos usando *semáforos* y *monitores* entre otros mecanismos.
- *Los métodos de enseñanza.* Basándonos en las tres estrategias metodológicas descritas anteriormente, hemos puesto en práctica la planificación metodológica que se detalla en el próximo apartado, haciendo uso de herramientas tecnológicas en el aula para potenciar el proceso de enseñanza-aprendizaje.
- *La secuenciación de los contenidos.* La secuenciación de los contenidos se desarrolla en la metodología propuesta introduciendo primeramente ejemplos sencillos e incrementando posteriormente el nivel de dificultad y contenidos progresivamente.
- *La organización social de las actividades de aprendizaje.* Se ha fomentado la participación en grupos dentro del aula, resolviendo los ejercicios planteados con el fin de aumentar la motivación y la participación.

4. Planificación metodológica

Para plasmar las estrategias metodológicas comentadas, se ha realizado la planificación que se muestra en la figura 1.

Figura 1: *Esquema metodológico*

Se realiza la presentación teórica de los conceptos de concurrencia, exclusión mutua, sincronización, *semáforos*, *semáforos binarios* y *monitores*. Esta exposición teórica se realiza de la forma más concisa y clara posible, introduciendo algún ejemplo, pero sin dedicar excesivo esfuerzo en que el alumno sea capaz, con dichas exposiciones, de resolver cualquier problema. Seguidamente, se pone en conocimiento de los alumnos los enunciados de las prácticas obligatorias que deben realizar en el laboratorio, cuya complejidad es media-alta. Entendidos los enunciados, se pasa una encuesta a los alumnos cuyo objetivo es establecer un nivel de referencia para poder medir de alguna forma las mejoras obtenidas por la aplicación de la metodología. Tras evaluar los resultados de la encuesta, vimos que la mayoría de los alumnos pensaban que les resultaría muy difícil realizar la práctica, tal y como habíamos previsto. Posteriormente, se hace una presentación del entorno gráfico de programación concurrente JBACI aprovechando que se muestra el primer ejemplo de programación concurrente, de manera que se les explica los menús y las diferentes opciones del programa. Tras esta presentación del entorno, se presentan problemas resueltos con JBACI en el aula de teoría mediante el uso de un cañón conectado al ordenador donde el profesor utiliza el simulador y transparencias para ir explicando los conceptos teóricos. Se empiezan las simulaciones con problemas muy sencillos implementados con *semáforos*, *semáforos binarios* y *monitores* cuyo único objetivo es afianzar los conocimientos teóricos y llamar la atención del alumno para conseguir que piense que no es tan difícil como parecía a priori y que se sienta capaz de ir participando en los siguientes ejercicios planteados. Para cada problema presentado se discute en clase el enunciado y una vez entendido éste se presenta el código en JBACI. En unos casos este código corresponde a la solución y en otros se presenta una solución incompleta o con errores que se resuelven 'en directo', preferiblemente como consecuencia de un debate provocado acerca de cómo solucionarlo.

Estos ejercicios planteados por el profesor han sido desarrollados por un grupo de alumnos del curso anterior en prácticas internas (prácticas tutorizadas que convalidan créditos docentes). Para la tutorización de estas prácticas internas utilizamos las tutorías virtuales con la herramienta MSN Messenger. Dado el buen funcionamiento de la herramienta para este fin (véase [7]) y el carácter multicampus de nuestra universidad, hemos extendido este tipo de tutorías a todos los alumnos de este curso.

Cuando el tiempo destinado a esta parte del temario se ha agotado y, tras la realización de las prácticas, es el momento de volver a realizar la encuesta al alumnado con el objetivo de poder medir las diferencias respecto a la encuesta inicial y analizar sus resultados. Una vez evaluado el cuestionario, la mayoría de los alumnos coinciden en que les ha resultado menos difícil de lo que inicialmente habían supuesto. También han respondido de forma satisfactoria en cuanto al uso del simulador JBACI.

5. Herramientas

5.1. JBACI

Los conceptos tales como sección crítica, concurrencia y las técnicas de sincronización entre procesos son importantes en informática. Además, debido al auge de la computación paralela y distribuida, entender la concurrencia y sincronización de procesos se hace más necesario que nunca para los alumnos. En general, para obtener un conocimiento robusto de los conceptos teóricos, es aconsejable cierta dosis de experimentación. En concreto, para asimilar completamente los conceptos mencionados, es necesario experimentar con la programación concurrente. JBACI es una opción para obtener esta experiencia con programación concurrente, que facilita el entendimiento de los conceptos de sección crítica, concurrencia y sincronización. JBACI viene de Java Ben-Ari Concurrent Interpreter. Se trata de un compilador de lenguaje $C--$, un dialecto restringido de lenguaje C++, que genera código objeto interpretable (PCODE). El compilador y el intérprete fueron originariamente procedimientos en un programa escrito

por M. Ben-Ari [1], basados en el compilador original de Pascal de Niklaus Wirth. Ben-Ari tomó el lenguaje Pascal-S y le añadió construcciones de programación concurrente tales como la construcción *cobegin... coend* y la variable tipo *semáforo* con operaciones *wait* y *signal*. JBACI simula la ejecución de procesos concurrentes y soporta las siguientes técnicas de sincronización: *semáforos enteros*, *semáforos binarios* y *monitores*.

Como hemos comentado, el nivel de programación en lenguaje C de los alumnos era suficientemente bajo como para cuestionarnos el uso de las normas POSIX en las prácticas de la asignatura. Pensamos que perderían más tiempo resolviendo problemas relacionados con la inexperiencia en el lenguaje, que centrándose en los contenidos. Por este motivo y los siguientes decidimos utilizar el simulador JBACI:

- *Simplicidad en la sintaxis de JBACI.* Un subconjunto muy limitado de C++.
- *Entorno multiplataforma.* Permite que cualquier alumno pueda instalarlo fácilmente en casa, independientemente del sistema operativo de que disponga, pues existen las versiones en JAVA (Windows), Linux/Unix y MS-DOS.
- *Se trata de un software de distribución gratuita y se distribuye con el código fuente.*
- *El entorno JBACI utilizado es un entorno gráfico muy sencillo de utilizar.* Permite la ejecución paso a paso, pudiendo realizar operaciones de depuración y obtener el valor de las variables tanto globales como las internas de los *semáforos* y *monitores*. Esto no es sencillo de realizar en una aproximación basada en POSIX para un alumnado sin experiencia.
- *JBACI actualmente es una herramienta muy extendida y usada en otras universidades para la docencia de esta parte de la asignatura [15].*

5.2. Tutorías con MSN Messenger

MSN Messenger es una aplicación de libre distribución que permite la conexión directa entre dos o más ordenadores de forma que dos o más personas pueden mantener una conversación en tiempo real, bien de forma escrita u oral (si se dispone de tarjeta de sonido y micrófono) e incluso visual (si se dispone de webcam). El único medio técnico que se necesita es una conexión a Internet con lo cual está disponible en cualquier lugar en el que se tenga acceso a dicha conexión.

En cuanto a la aplicación MSN Messenger decir que su uso está resultando de una utilidad fundamental en el desarrollo de las tutorías virtuales. Varias son las características que hacen de este software una herramienta tan útil para nuestro objetivo. Entre ellas cabe destacar que se trata de un software de libre distribución, muy intuitivo y sobre todo ampliamente utilizado por la comunidad estudiantil, lo que implica que no es necesario ningún tipo de explicación sobre la instalación o el manejo de dicha aplicación. Se ha utilizado la opción 'Compartir aplicación' que permite al profesor ver lo que está realizando el alumno en el entorno JBACI, así como tomar el control de dicha aplicación de forma remota y solucionar las dudas que plantea. Además de poder ver y manejar la aplicación que se está compartiendo se dispone a la vez de otros tres medios que permiten que la comunicación sea completa: la ventana de texto, que permite el intercambio de preguntas y respuestas de forma escrita; el micrófono, que permite comunicarse de forma verbal (que suele ser más rápida y menos ambigua que la forma escrita); la pizarra, que esporádicamente permite realizar esquemas o gráficos sencillos que añadan claridad a la explicación. Tanto las conversaciones en texto como las explicaciones en la pizarra pueden ser guardadas para consultas posteriores por los estudiantes.

Este sistema permite una reunión 'virtual' en la que cada persona puede estar en campus distintos (o incluso en su propia casa) y participar de forma conjunta. Además de evitar los desplazamientos de los alumnos, éstos no están sujetos a las horas de tutorías pues se les

ha ofrecido la posibilidad de contactar con el profesor en cualquier momento en el que éste esté conectado (siempre que éste disponga de tiempo, lógicamente).

Hay otro medio disponible en MSN Messenger que es el uso de una webcam. Nosotros no hemos utilizado dicho dispositivo pero puede ser útil para que los alumnos muestren más interés y las sesiones de tutorías resulten más 'cercanas'.

Figura 2: *Pantalla de conversación del Messenger*

6. Ejemplo de tutoría

Para poner un ejemplo de cómo se ha realizado la experiencia, se ha capturado una imagen de JBACI. JBACI es un entorno gráfico de desarrollo realizado en JAVA que utiliza BACI en su núcleo pero que ofrece un interfaz más completo y cómodo para el usuario. En la página web de BACI [15] se puede acceder a este entorno.

En la figura 3 se muestra una captura de la ventana de edición del entorno de desarrollo JBACI, que es el que se ha usado en clase para mostrar los problemas. El código de la figura corresponde al primer problema propuesto, donde se utilizan dos procesos cooperantes para conseguir un objetivo, sumar 20, donde cada proceso aporta 10 a la suma total. Se

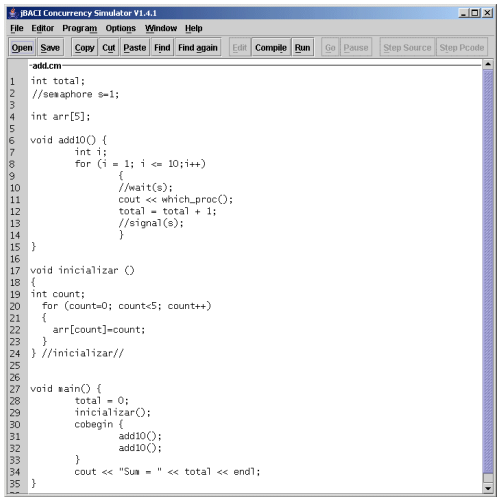


Figura 3: *Editor código JBACI*

aprovecha el ejemplo para presentar la forma de inicializar en JBACI los *arrays*, la sintaxis del bucle *for*, la sentencia *cobegin*, cómo imprimir en la consola con *cout* y cómo obtener el identificador de un proceso con la función *which-proc*. Pero sobre todo se llama la atención de los alumnos en cómo se puede dividir una tarea en la ejecución de procesos cooperantes para realizarla. En el ejercicio se pretende demostrar cómo, si no se protege mediante exclusión mutua la variable *total*, que es utilizada por los procesos cooperantes, se pueden producir resultados erróneos por no garantizar la exclusión mutua en la lectura y actualización de la variable *total*, es decir, la instrucción $total = total + 1$ debería ejecutarse sin interrupción. En la imagen aparecen comentadas las sentencias de declaración del *semáforo* que se usa para establecer la sección crítica y las sentencias *wait* y *signal* que la implementan. En el ejemplo que se muestra a los alumnos, estas instrucciones comentadas, no aparecen. Se ejecuta el programa varias veces y se observa que cada vez da un resultado distinto y que normalmente no se obtiene el resultado esperado, 20.

En la figura 4 se observan varias pantallas

del entorno JBACI, donde podemos ver el proceso principal *main* y los dos procesos cooperantes *Process 1 add10* y *Process 2 add10*, todos ellos con su código correspondiente. Este

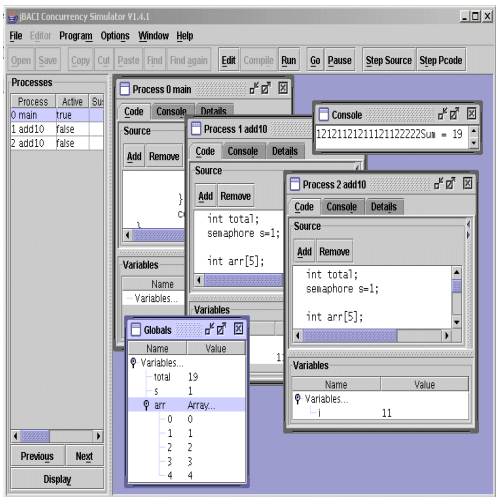


Figura 4: *Pantalla ejecución JBACI*

código se puede ejecutar sin pausa o paso a paso y observar sus efectos. En la ejecución paso a paso se puede observar cómo el código que se ejecuta es de uno u otro proceso, además se puede observar en la ventana de la consola cómo se van mostrando las impresiones y en la ventana de variables globales los valores de las variables y el valor interno de los *semáforos* según se ejecuta el código. Cuando se produce un cambio de proceso se puede observar este hecho en la lista de procesos (a la izquierda de la imagen) indicando en cada momento qué proceso está activo.

7. Futuras actuaciones

Para el curso que viene queremos introducir la 'educación constructivista' en esta parte de la materia, pretendiendo no sólo que el alumno adquiera conocimientos sino también destrezas y habilidades como la capacidad de análisis crítico de la información localizada en Internet. Para ello nos hemos propuesto realizar una guía de aprendizaje siguiendo el modelo CAIT [11] en el que el alumno aprende a construir su propio conocimiento. El alumno, siguiendo esta guía de aprendizaje, localizará el material teórico y práctico sobre *concurrencia* así como el propio simulador. Lo contrastará con la documentación proporcionada por el profesor que tendrá un rol de *potenciador de la actividad, facilitador* y *observador*.

Los alumnos podrán utilizar dos foros, uno técnico, referido a problemas de instalación de la herramienta JBACI y otro teórico-práctico sobre aspectos propios de la materia y la resolución de problemas. La participación en los foros será tenida en cuenta en la evaluación final, de forma que se motive a los grupos a resolver los problemas planteados por los compañeros.

También pretendemos generar unas FAQ's (*Frequently Asked Questions*) con las dudas más interesantes resueltas durante las tutorías. Estas FAQ's podrían incluir grabaciones de las sesiones de tutorías utilizando software de captura de pantalla. Dicho material multimedia podría servir como material complementario autodidacta para los alumnos (véase [8]).

8. Conclusiones

Nuestra experiencia nos ha proporcionado una gran satisfacción al ver que el grado de asimilación de los conceptos presentados ha sido mayor de lo esperado. Su aplicación tuvo un efecto positivo, incrementando la participación y atención del alumno.

Como principales logros obtenidos podemos enumerar que:

- En general el alumnado ha perdido el 'miedo' a la participación en clase y a utilizar las tutorías.
- Se ha mejorado la comprensión de la teoría presentada.
- Los alumnos llegan a las prácticas más preparados y aprovechan mejor el tiempo de la práctica.
- Se han obtenido mejores resultados que en otros temas de la asignatura.

- Se ha eliminado la problemática de la distancia para las tutorías gracias a la tutoría virtual con MSN Messenger.
- Dado el alto grado de participación del alumnado en el uso de las tutorías virtuales, podemos decir que se ha logrado una mayor motivación del alumnado que en otras partes de la asignatura

Haber conseguido esta actitud positiva en el alumnado ha hecho que el esfuerzo empleado en hacer viable nuestra idea haya merecido la pena.

Como conclusión final queremos remarcar que esta metodología podría ser aplicada a la docencia de cualquier otro área siempre que se disponga de la herramienta software necesaria, ya que dicha metodología tiene su base teórica principalmente en la secuenciación de los contenidos, la sensación de logro por parte del alumnado (auto motivación), la claridad en la exposición de los contenidos y, la participación grupal en el aula y fuera de ella.

Referencias

[1] M. Ben Ari. *Principles of concurrent and distributed programming*. Ed: Prentice-Hall, 1990.

[2] J. Carretero, F. Garcia, P.M. Anasagasti y F. Pérez. *Sistemas Operativos: una visión ampliada*. Mc Graw Hill, 2001.

[3] J. Carretero, F. Garcia y F. Pérez. *Prácticas de Sistemas Operativos: de la base al diseño*. Mc Graw Hill, 2002.

[4] J. Carretero, F. Garcia y F. Pérez. *Problemas de Sistemas Operativos: de la base al diseño*. Mc Graw Hill, 2003.

[5] J.L Castillejo. *Pedagogía Tecnológica*. Ceac, 1987.

[6] C. Coll, J. Palacios y A. Marchesi. *Desarrollo psicológico y educación*. Alianza, 1991.

[7] V. Galiano, O. López, M. Martínez, H. Migallón, P. Piñol y D. Ubeda. *Docencia de semáforos y Monitores con BACI: Un ejemplo de aprendizaje guiado*. X Jornadas de enseñanza universitaria. Jenui 2004.

[8] M.M Galotto, A. Pomares, O. López, P. Piñol, M. Martínez. *Mejora de la Docencia con la creación de material docente complementario autodidáctico*. www/Internet (CIAWI 2004).

[9] J. Gimeno Sacristán. *Autoconcepto, Sociabilidad y Rendimiento escolar*. MEC, 1976.

[10] F.M. Márquez. *Unix : Programación avanzada*. Ra-ma, 1996.

[11] Martín Patiño J.M., Beltrán Llera J.A., Pérez Sánchez L. *El modelo pedagógico CAIT*. Fundación Encuentro, 2003. http://www.educared.net/-InnovacionPedagogica/htm/modelocait_-definicion.htm

[12] OCDE. *Learning to Change Embracing: ICT at Schools*. 2001.

[13] P. Rivière. *Los negocios del multimedia en la escuela*. Le monde Diplomatique, páginas 27, 28, 04 1998.

[14] Kay A. Robbins y S. Robbins. *Unix: Programación práctica. Guía para la concurrencia, la comunicación y los multihilos*. Prentice Hall Hispanoamericana, 1997.

[15] *Página web principal de BACI: http : //www.mines.edu/fshome/tcamp/baci/*

Prácticas de domótica y sistemas industriales. Entornos de control y comunicaciones

Andrés Fuster Guilló
Dpto. de Tecnología Informática y Computación
Universidad de Alicante
Apdo. Correos 99. E-03080. Alicante
fuster@dtic.ua.es

F. Javier Ferrández Pastor
Dpto. de Tecnología Informática y Computación
Universidad de Alicante
Apdo. Correos 99. E-03080. Alicante
fjferran@dtic.ua.es

Jorge Azorín López
Dpto. de Tecnología Informática y Computación
Universidad de Alicante
Apdo. Correos 99. E-03080. Alicante
jazorin@dtic.ua.es

Resumen

En este artículo se presenta una serie de entornos para prácticas docentes en asignaturas de domótica y sistemas industriales donde se hace hincapié en el control y las comunicaciones. Estos entornos representan situaciones comunes en la sociedad de la información, donde el acceso a los servicios será posible desde cualquier lugar y en cualquier momento. Con estos requerimientos se diseña la arquitectura del sistema siendo conscientes de las necesidades didácticas y económicas de las prácticas de laboratorio. A partir de este diseño general se instancian dos paneles: industrial y domótico. Las conclusiones que se han extraído de la experiencia en las aulas han motivado el desarrollo de nuevos entornos con tecnología domótica X10 y EIB.

1. Introducción

La necesidad de material didáctico enfocado hacia las prácticas de laboratorio en las asignaturas relacionadas con el control y las comunicaciones en entornos industriales y domésticos y, en general, para las asignaturas relacionadas con las IST (Information Society Thecnologies — Tecnologías de la Sociedad de la Información) [1] se hace cada vez más aparente cuando se precisa de un acercamiento por parte del alumno hacia situaciones más realistas. Esta necesidad viene motivada más si cabe cuando es una realidad que el "*miedo al hardware"* está presente entre los alumnos de ingeniería en informática. El tratamiento de los procesos de información deriva en muchos casos hacia un aislamiento del exterior. Obtener un dato del entorno para poder elaborar información o actuar en un determinado proceso es ofrecido por los interfaces entre el mundo exterior y el proceso de información. Es importante dar a conocer estos interfaces y de alguna forma hacer notar en el alumno que es capaz de encender una bombilla, mover un motor o saber si hace frío o calor en una habitación. La intención es que el alumno adquiera la actitud necesaria para enfrentarse a cualquier tipo de problema, ya sea el control de un proceso industrial, ya sea el programa de contabilidad de una empresa. El alumno es más reacio a enfrentarse al primer tipo de problema.

Las nuevas metas de la sociedad de la información exigen una transformación desde enfoques de disciplinas aisladas con integraciones no estructuradas hacia una integración de servicios de procesamiento de información y comunicaciones en el marco de las IST gracias a la evolución de proyectos globales donde existirán campos de trabajo multidisciplinares como el de automática, electrónica, informática y comunicaciones. En este nuevo marco el ingeniero en informática debe ser la pieza fundamental en el desarrollo de servicios para las IST.

En este contexto, las prácticas relacionadas con el control y monitorización del entorno han sido realizadas habitualmente mediante simuladores debido a su gran versatilidad y economía. Sin embargo, genera una pérdida de realidad con el correspondiente miedo a lo desconocido en el alumno.

En este artículo planteamos la utilización de material docente para prácticas de laboratorio a caballo entre la implementación real y la simulación con la intención de aprovechar las ventajas de ambas visiones. Para ello se presentan una serie de paneles didácticos que permiten una mejor comprensión de los interfaces entre el mundo exterior y el procesamiento por parte del alumno, y que pueda ser explotado a distintos niveles de abstracción para las distintas prácticas que se pueden elaborar sobre éstos. La

arquitectura que se plantea presenta las características de funcionalidad y economía necesarias para poder ser implantada en las asignaturas relacionadas con las nuevas metas de la sociedad de la información.

Concretamente, en el contexto de los planes de estudios de Informática 2001/2002 [2][3][4] de la Universidad de Alicante y dentro del área de conocimiento Arquitectura y Tecnología de Computadores, se contemplan asignaturas optativas concebidas con carácter complementario al cuerpo de conocimiento impartido en materias obligatorias y troncales (Sistemas Operativos, Arquitectura de Computadores, Redes, etc.) y otras con un marcado carácter aplicado. Estas asignaturas de enfoque aplicado de las materias ofertadas a los estudiantes de informática están completamente relacionadas con los objetivos de las IST: Informática Industrial, Domótica y Edificios Inteligentes, Informática de Comunicaciones, etc. Uno de los objetivos de estas asignaturas es familiarizar al alumno sobre las posibilidades de integración de Internet en el desarrollo de redes de control especializadas en tareas de supervisión y control. Adoptando una filosofía de microdispositivos integrados capaces de obtener datos y comunicarse mediante protocolos muy establecidos como el TCP/IP, estas tecnologías ofrecen un valor añadido a las actuales soluciones propietarias que ofrece el mercado.

A través de los ejemplos en las asignaturas relacionadas con los sistemas industriales y la domótica vamos a exponer las características que presenta el panel de prácticas. Para ello primero definiremos en el siguiente apartado los servicios generales de la sociedad de la información en el marco socioeconómico europeo mediante las metas de las IST. Más tarde, se expone la arquitectura general del sistema. En los siguientes puntos se tratan las instancias para prácticas industriales y domótica. Finalizaremos el artículo con las conclusiones extraídas del trabajo.

2. Servicios de la sociedad de la información

En la cumbre de marzo de 2000 en Lisboa, los líderes de la Unión Europea fijaron un nuevo y ambicioso reto para Europa: convertirse en la economía basada en el conocimiento más competitiva y dinámica del mundo. Así el programa de las Tecnologías de la Sociedad de la Información que parte del V Programa Marco propone una visión de futuro donde sitúa al usuario en el centro del desarrollo de servicios de las tecnologías de información y la comunicación. El objetivo principal es la ubicuidad: tecnología casi invisible que se mezcla con nuestro entorno diario. Las personas son capaces de acceder a los servicios y aplicaciones de las IST: desde donde quieran, cuando quieran, y como quieran.

Se abren cuatro líneas clave de actuación:

- *Sistemas y servicios para los ciudadanos*: sistemas y aplicaciones innovadoras de interés general (salud, discapacitados, administración, medio ambiente, transporte, turismo,…).
- *Nuevos métodos de trabajo y comercio electrónico*: tecnologías que permiten a los individuos, empresas y otras organizaciones adaptarse y competir en la economía digital.
- *Contenido y herramientas multimedia*: funcionalidad, utilidad e impacto de futuros servicios y productos de información, particularmente en el contexto de la cultura y diversidad del lenguaje europeos.
- *Tecnologías esenciales e infraestructuras*: tecnologías que soportan la actual convergencia en industria e infraestructura y su integración en sistemas, aplicaciones y redes.

En relación a estas cuatro líneas clave y sobre todo con las de *Sistemas y servicios para los ciudadanos* y las *Tecnologías esenciales e infraestructuras*, los proyectos evolucionan hacia espacios personales (coche, casa, trabajo…) inteligentes e interconectados constituyendo los interfaces a un mundo de servicios: *e-learning*, *e-health*, *e-work*. Consideran también una evolución hacia una nueva generación de objetos inteligentes donde los objetos cotidianos tendrán capacidades de comunicación y procesamiento (papel, bolígrafos, mesas en los colegios o en los cafés, bancos de un parque, carteleras, ropas, electrodomésticos inteligentes,…).

En este contexto, el objetivo principal es extender las plataformas de procesamiento y de comunicación para ofrecer servicios, con lo que vendrá aparejado la necesidad de la utilización de plataformas y estándares abiertos que permitan el enlace de equipos en redes heterogéneas del

hogar, de la industria, etc. Este hecho permitirá una evolución desde los enfoques clásicos de automatización, control, monitorización, etc., hacia un enfoque de comunicaciones entre microdispositivos.

3. Arquitectura del sistema

El principal objetivo que nos marcamos para el diseño de los paneles de prácticas es que el alumno sea capaz de adquirir las habilidades y actitudes suficientes para enfrentarse a la integración del control y las comunicaciones en entornos industriales y domésticos. Con esto, se han seguido distintos criterios funcionales y de coste que han dado como resultado paneles que están a caballo entre el funcionamiento real y la simulación. Ubicar el panel entre estas dos situaciones nos aporta la versatilidad y economía de la simulación sin perder la visión de la realidad y evitando la complejidad de trasladar un escenario industrial y/o domótico a un laboratorio.

Atendiendo a los criterios funcionales y siguiendo el contexto de aplicación expuesto en el apartado anterior y a la evolución que se vislumbra —necesidad de plataformas y estándares abiertos que permitan el enlace de equipos en redes heterogéneas—, la tecnología de Internet y los dispositivos embebidos capaces de operar con ella ofrecen multitud de posibilidades. Es con esta filosofía que el panel dispone de distintos dispositivos con capacidades de control y comunicaciones a través de servidores web embebidos.

Los dispositivos ofrecen gran versatilidad y generalidad asociadas a las posibilidades que determinan el intercambio de información basado en páginas web. Además, disponen de las características necesarias para su integración en tareas de control ya sean de procesos industriales como de servicios domóticos. A continuación se exponen las principales características de éstos:

- *Facilidad de uso*: Esta tecnología nos aporta aplicaciones fáciles de utilizar, mantener o modificar. La interfaz con el usuario está basada en páginas web cuya utilización está ampliamente difundida y aceptada. El diseño, mantenimiento y modificación de páginas web no necesita de un alto grado de especialización.
- *Sistema abierto*: ofrecen soluciones independientes de la plataforma. Los servicios web no dependen ni del tipo de bus de control instalado ni de los sistemas operativos con los que se trabaje. Cada bus de control deberá contar con una pasarela a Internet y cada sistema operativo deberá incluir la pila de protocolos de comunicaciones necesaria.
- *Acceso transparente*: permiten el acceso a la información y los servicios desde cualquier nodo conectado a la red.
- *Estándar libre*: el diseño y desarrollo de las aplicaciones que implementan los servicios están basados en estándares y tecnologías ampliamente extendidas y de acceso libre. Los ficheros HTML pueden crearse con cualquier editor de texto disponible y las aplicaciones pueden implementarse en distintos lenguajes de programación establecidos (C, Java, etc).

Estos dispositivos además tienen un coste económico bajo. En [5] se revisan distintas características de microcontroladores que ofrecen servidores de páginas web embebidos.

Para la realización de los paneles disponemos de distintos dispositivos de red embebidos montados en forma de microautómata: el DK40 de la empresa alemana Beck [6]. Este microautómata dispone de 8 entradas y salidas digitales entre 15 y 30 Vcc con led de estado, 2 interfaces puerto serie (RS232, RS485) e interfaz Ethernet (10BaseT). Utiliza el IPC@CHIP® Single Chip Embedded-Webserver de la misma empresa [7]. Se trata de un controlador embebido diseñado para aplicaciones en red. El controlador incorpora el hardware y software necesario para tal fin. El hardware consiste en una CPU 186 de 16 bits 20 MHz, memoria RAM y Flash, Ethernet, Watchdog y detección de falta de alimentación. El software preinstalado es un sistema operativo en tiempo real (RTOS) con sistema de ficheros capaz de ejecutar aplicaciones DOS de forma concurrente, la pila TCP/IP, un servidor Web con soporte para CGI, Ftp y Telnet.

El funcionamiento básico del microautómata DK40 se puede dividir en dos grandes bloques: los procesos de control y los procesos para servicio web. Estos bloques pueden funcionar de manera independiente, sin embargo la fusión de los dos procesos aporta toda la funcionalidad descrita en los párrafos anteriores.

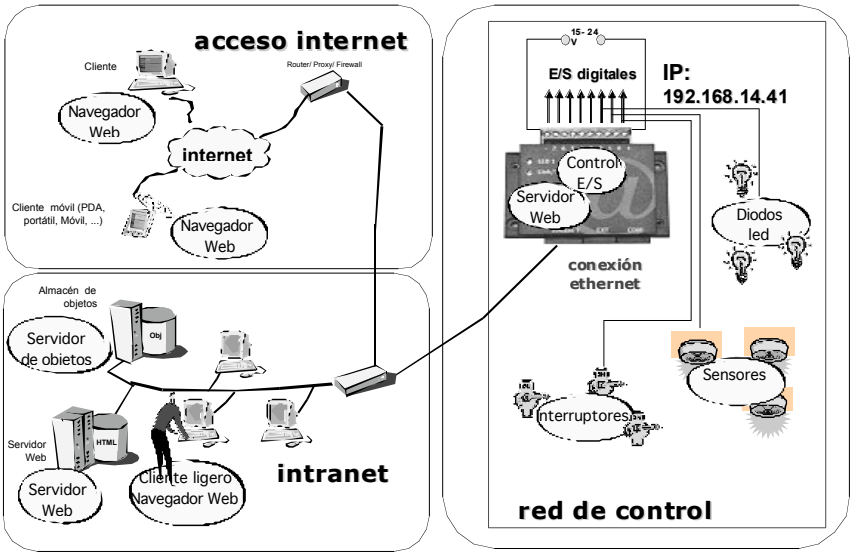


Figura 1. Escenario de aplicación de los entornos de prácticas de control y comunicaciones

En este sentido, de las características que vamos a utilizar del dispositivo, el soporte para CGI es la más importante. Seremos capaces de dar "órdenes" al microautómata para que active o desactive sus salidas; o que podamos "leer" sus entradas a través de páginas web. La plataforma soporta programas CGI implementados en distintos lenguajes de programación.

Expuesta la plataforma que nos permite el control y la comunicación a través de servicio web, nos falta determinar cuál es la planta o proceso a controlar. Estos procesos dependerán del entorno de aplicación. Es por ello que desde este punto y siguiendo criterios de coste de realización así como de versatilidad, nos hemos planteado la simulación de los distintos procesos utilizando una serie de diodos led y de interruptores conectados a la matriz de entradas y salidas del DK40. Estos diodos determinarán, dependiendo del caso, la conexión de un determinado aparato, el transvase de información, etc. Los interruptores, por su parte, generarán las entradas al sistema: apertura de una puerta, un incendio, una inundación, etc.

En la figura 1 se puede observar cual es el posible escenario de aplicación de los paneles. En los laboratorios de prácticas (intranet) conectamos los distintos microautómatas de los paneles a la red existente en el mismo y les asignamos distintas direcciones IP. A partir de aquí, el microautómata se convierte en un servidor de páginas web y un procesador de control accesible desde cualquier punto según la configuración existente en los laboratorios. En este escenario los alumnos podrán enviar peticiones al servidor utilizando las máquinas del laboratorio así como descargar los programas de control necesarios en el mismo dependiendo de los requerimientos de las prácticas.

A continuación analizamos las prácticas desarrolladas para asignaturas relacionadas con sistemas industriales y domóticos.

4. Prácticas industriales

Las asignaturas relacionadas con los sistemas industriales, *Sistemas Industriales* en la Universidad de Alicante [8], ofertadas a los alumnos de ingeniería informática pretenden cubrir los siguientes objetivos, entre otros:

- Conocer tecnologías basadas en computador que resuelven tareas de regulación y control de procesos industriales.
- Comprender modelos que describen el comportamiento de los procesos y saber

manejar las herramientas de simulación que los analizan.
- Poder identificar el tipo de proceso y proponer los dispositivos hardware y los programas necesarios para su automatización o control.
- Adquirir el conocimiento en la materia que sirva de base para poder integrarse con éxito en las actividades profesionales relacionadas con la informática industrial.

En las prácticas de laboratorio se profundizan diferentes aspectos relacionados con la adquisición de conocimientos, habilidades y actitudes, desarrollando distintos tipos de escenarios. Se plantea analizar los problemas de automatización secuencial y proponer algoritmos que los resuelvan; proponer interfaces hombre máquina en tareas de monitorización y supervisión de procesos; analizar los requerimientos de las redes de comunicación en entornos industriales, etc.

Siguiendo los objetivos de la asignatura se plantea un escenario básico que nos permita tratarlo a diferentes niveles de realización. Para ello, suponiendo una empresa ficticia que se encarga de la fabricación de puertas a partir de troncos de madera, el proceso se desarrolla de la siguiente forma (ver Figura 2).

La fabricación toma el punto de partida con la entrada de la materia prima que se almacena en el *Almacén de materia prima*: troncos de madera. A continuación, ésta se transporta desde el punto de entrada a dos máquinas que realizan un acondicionamiento de los mismos (automático y semiautomático) para más tarde ser cortados en dos máquinas (cada uno con un tipo de corte). El proceso termina con el producto elaborado, almacenando el deshecho para que pueda ser reciclado. Mediante un sistema de visión artificial se realiza la inspección del producto y los tablones son ordenados y almacenados por calidades.

En este hipotético escenario el objetivo es la automatización del proceso industrial y la convergencia con los objetivos de la empresa. Para ello contamos con una serie de servicios que engloban desde la monitorización del estado de las máquinas hasta los servicios de administración. En los servicios de administración se contempla la gestión de pedidos, albaranes, facturas, nóminas, etc. En la monitorización se conocerá el estado o ritmo de fabricación de las máquinas de la industria, información que podrá ser utilizada para una planificación posterior del proceso de fabricación. También se considerarán servicios para planes de contingencia donde se podrá consultar las actuaciones a tomar caso de ocurrir cualquier incidencia en el proceso de fabricación.

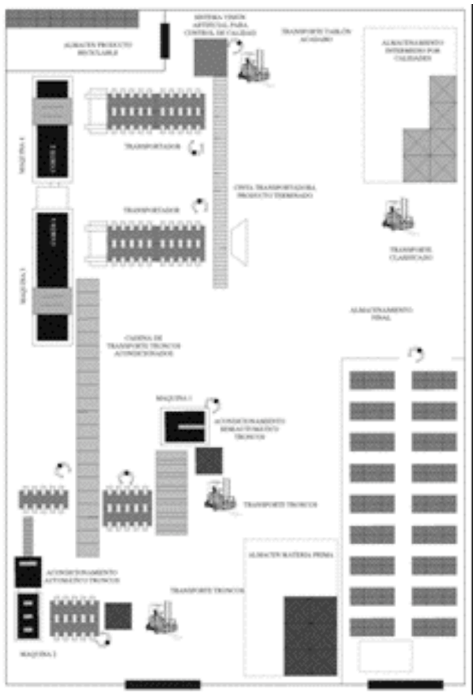


Figura 2. Escenario de aplicación para la industria.

Las ventajas proporcionadas por las IST en cuanto a la gestión de los procesos de conocimiento toman relevancia en este marco, cuando se realiza la gestión de la información, permitiendo realizar una planificación de la fabricación a partir de los planes estratégicos de la empresa. A partir de los objetivos de fabricación se pueden lanzar planes específicos por máquina: descripción de la fabricación, ritmo, etc.

Con este escenario y con estos objetivos generales y para las prácticas de laboratorio se plantea el panel industrial de la Figura 3. En este panel se introduce un marco funcional que simula cualquier transvase de información entre los distintos elementos que intervienen en el proceso de fabricación planteado. El nivel de realización práctico que se puede plantear varía en el detalle de control y las comunicaciones.

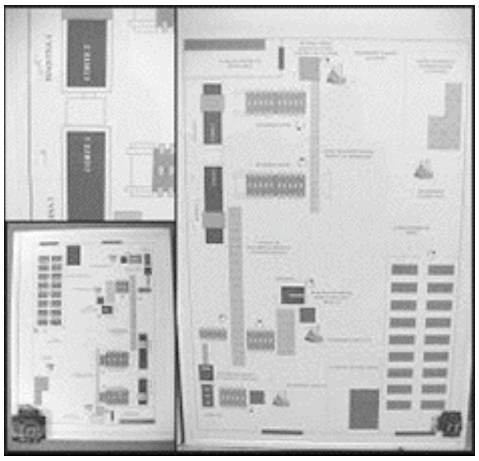


Figura 3. Foto del panel industrial y detalle de la implementación.

En el nivel de mayor abstracción y que sirve de base para posteriores implementaciones, se han diseñado una serie de páginas web que sirven como introducción a los alumnos. En la Figura 4 se pueden observar las páginas relacionadas con la monitorización de la maquinaria y los servicios generales en la industria.

Figura 4. Páginas web para el panel industrial.

En prácticas posteriores se enfatiza el apartado de control de los dispositivos. En este sentido, se propone el control de la máquina de corte para la madera. Ésta se realizará posicionando un cabezal de corte gestionado por uno o dos motores. En la Figura 5 se observa la versión que simula el corte de la madera utilizando el motor de una vieja impresora y control directo a través de las salidas del microautómata.

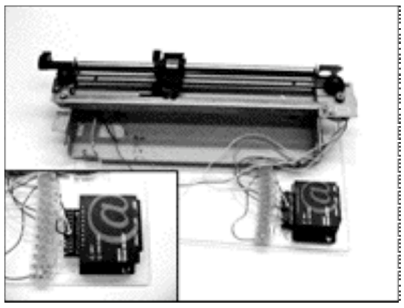


Figura 5. Prototipo que simula el corte de madera en un solo eje.

En la Figura 6 se observa una versión refinada donde se utiliza un cabezal que puede moverse en los dos ejes X e Y.

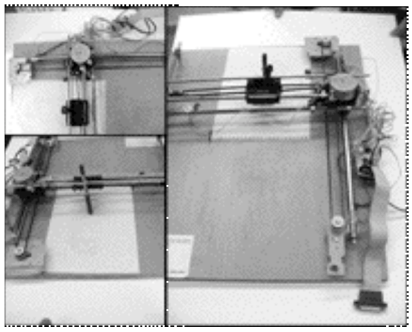


Figura 6. Prototipo refinado de corte de madera con posibilidad de movimiento en el plano.

5. Prácticas domóticas

En la asignatura relacionada con el control y las comunicaciones en el entorno doméstico, *Domótica y Edificios Inteligentes* en la Universidad de Alicante [9][10], el alumno debe conocer cuáles son las tecnologías que existen actualmente y ser capaz de enfrentarse a la especificación y diseño de estos sistemas domóticos. Como grandes subobjetivos se plantean, entre otros, el conocer los subsistemas domóticos y los dispositivos de percepción y actuación, así como estudiar los estándares actuales y las tecnologías de comunicación en edificación. Ambos relacionan la capacidad de obtención y actuación con el mundo exterior así como con las capacidades de comunicación.

El escenario de aplicación que se plantea para llevar a cabo estas prácticas es una vivienda unifamiliar (ver Figura 7) con dependencias como dormitorios, salón, cocina, etc. Sobre esta vivienda se pretende desarrollar una serie de servicios que establezcan un sistema domótico incidiendo en la integración del control y las comunicaciones. Los servicios a considerar serán los de seguridad, ahorro energético, confort y comunicaciones.

Los objetivos para las prácticas que se desarrollan están relacionados con los aspectos a incidir en el conocimiento del alumno: desde la concepción de los subsistemas domóticos y los dispositivos de percepción y actuación, en su enfoque clásico, hasta la integración de servicios de procesamiento de información y comunicaciones.

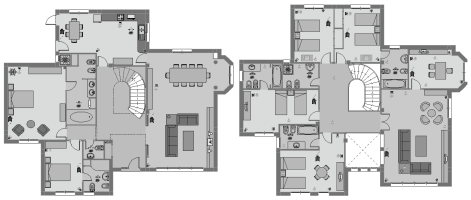


Figura 7. Escenario de aplicación para entornos domésticos.[1]

En este escenario y con estos objetivos, el panel domótico que se plantea está relacionado con los servicios que se pueden llevar a cabo en el hogar. Con esto, las necesidades de comprensión de la integración domótica y las características de las prácticas de laboratorio marcan la instancia desarrollada de la arquitectura del sistema. Para ello complementando la visualización del transvase de información, este panel está dotado de distintos interruptores que pretenden simular distintos eventos de entrada al sistema de tipo "todo o nada". En este caso se podrá simular un incendio, la entrada de un intruso, etc. Con este esquema el alumno podrá plantearse el diseño del control necesario para que ante diferentes eventos el sistema actúe en consecuencia (al producirse un incendio debemos avisar a los servicios de emergencia y accionar los mecanismos de extinción, si los hubiera) así como el planteamiento de distintos servicios de comunicaciones (*e-health*, *e-learning*, etc.)

En la Figura 8 se observa una foto del panel realizado junto con algunos detalles de implementación y las conexiones con el microautómata.

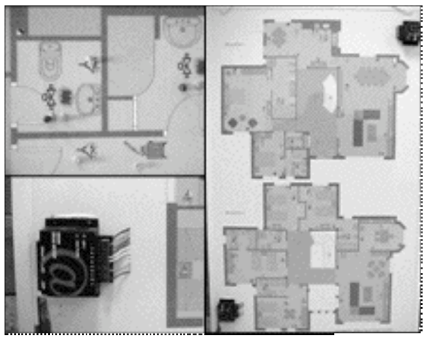


Figura 8. Foto del panel domótico y detalle de la implementación

Para la comprensión del sistema y como ocurre con el panel industrial se han desarrollado una serie de páginas web que el alumno tomará como base para la demostración de los servicios y los programas de control (ver Figura 9).

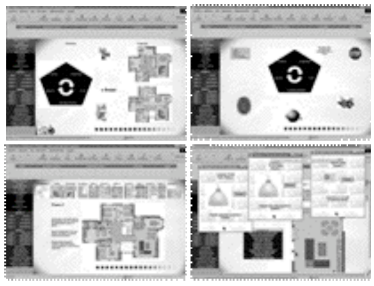


Figura 9. Páginas web para el panel domótico

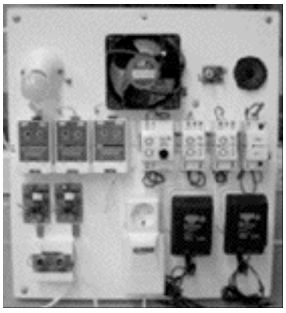
Siguiendo la misma filosofía de prácticas para docencia de la domótica, pero en este caso utilizando dispositivos físicos reales, durante el curso 2003/2004 [11] se elaboró un panel basado en el estándar X10 (ver Figura 10a). Este panel ha sido desarrollado para tratar los problemas de la pasarela residencial basada en el estándar OSGI. Así, se han utilizado distintos dispositivos X10 y la pasarela CM11 que permite realizar el interfaz entre la red de control eléctrica y la de datos. Con este panel el alumno encuentra una instalación

[1] Plano cedido por la arquitecto Mª José Fuster Guilló.

más cercana a la realidad doméstica y también se le permite realizar simulaciones.

En esta misma línea, durante este curso se está desarrollando un panel basado en la tecnología EIB (Figura 10b). Sin embargo, el coste asociado es mucho mayor que los anteriores debido al importe de cada uno de los dispositivos.

a)



b)

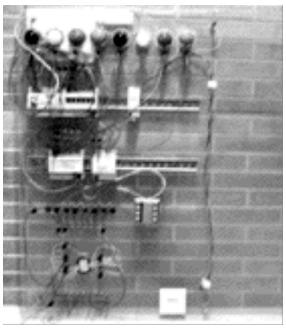


Figura 10. Paneles domóticos. a) X10 b) EIB

6. Conclusiones

En este artículo se han analizado los paneles diseñados para prácticas de laboratorio que inciden en el control y las comunicaciones de las asignaturas relacionadas con las nuevas metas de las tecnologías de la sociedad de la información.

Para el diseño de los paneles se han seguido diferentes criterios de funcionalidad y de coste dando como resultado una arquitectura que aporta versatilidad y economía, a su vez que repercute en una visión más realista por parte del alumno.

Las prácticas que se pueden desarrollar utilizando estos paneles varían en su nivel de abstracción: desde el nivel más alto considerando la funcionalidad de los sistemas hasta el más bajo a nivel de transducción de energías.

La comprensión por parte de los alumnos de los procesos de control y comunicaciones aumenta, de la misma forma que el interés mostrado en las prácticas donde se introducen estos paneles.

Referencias

[1] Programa de trabajo 2003-2004 del VI Programa Marco de la Unión Europea. *http://www.cordis.lu/fp6/home.html*.

[2] *Plan de Estudios conducente al título de Ingeniero en Informática de la Escuela Politécnica Superior de la Universidad de Alicante.* BOE número: 230-2001.

[3] *Plan de Estudios conducente al título de Ingeniero Técnico en Informática de Sistemas de la Escuela Politécnica Superior de la Universidad de Alicante*. BOE número: 230-2001.

[4] *Plan de Estudios conducente al título de Ingeniero Técnico en Informática de Gestión de la Escuela Politécnica Superior de la Universidad de Alicante*. BOE número: 230-2001.

[5] Eisenreich, D.; DeMuth, B. *Designing Embedded Internet Devices*. Elsevier Science, 2003

[6] Beck IPC *DK40 Datasheet & Hardware Manual* Mayo 2002. *www.bcl.de*

[7] Schlösser, E.; Gatrost, J. *SC12 Getting Started*. Oct. 2001 *www.bcl.de*

[8] Pujol López, F.; Ferrández Pastor, F.J.; Sánchez Romero, J.L.; García Chamizo, J.M.: *Propuesta para la asignatura Sistemas Industriales en las titulaciones de Informática.* JENUI'01. Palma de Mallorca. 2001

[9] Azorín López, J.; Fuster Guilló, A.; Maciá Pérez, J; Ferrández Pastor, F.J.: *Domótica y Edificios Inteligentes en la Universidad de Alicante*. JENUI'03. Cádiz, 2003

[10] Fuster Guilló, A.; Ferrández Pastor, F.J; Azorín López, J. *Entornos para prácticas de control y comunicaciones en asignaturas de informática industrial y domótica*. JENUI'04. Alicante, 2004.

[11] Fuster Guilló, A.; Capella D'Alton, A.; Azorín López, J. *Nuevas Tendencias en Domótica*. Actas JDAI'04. Alicante, 2004

Práctica de optimización de código teniendo en cuenta la arquitectura para primer ciclo

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

Resumen

Con el objetivo de motivar a los estudiantes en el estudio de materias de Arquitectura y Tecnología de Computadores, estamos proponiendo prácticas en las que los alumnos comprueban que pueden mejorar prestaciones en sus programas aplicando conocimientos adquiridos en estas materias. Aquí presentamos una práctica propuesta para asignaturas de estructura del primer ciclo, en las que se estudia la arquitectura *abstracta* del procesador o arquitectura del repertorio de instrucciones (ISA, *Instruction Set Architecture*). En la práctica propuesta los estudiantes, usando ensamblador dentro de código de alto nivel, mejoran las prestaciones de varios ejemplos: limitar los componentes de una lista a un rango, copia, inicialización y búsqueda. Comprueban que, utilizando conocimientos sobre arquitectura abstracta, pueden reducir el tiempo de ejecución de estos ejemplos en un 25%, 50%, o incluso en más de un 75%. La reducción depende del ejemplo, del procesador concreto, y del tamaño y ubicación en memoria de las entradas.

1. Introducción

La *motivación* es uno de los aspectos que más influye en el proceso de aprendizaje, de hecho el rendimiento y aprendizaje de un alumno motivado se aproxima al máximo de sus posibilidades. Para favorecer la motivación es esencial proporcionar un *sentido de utilidad* a los contenidos que se imparten. Los estudiantes que prevén dedicar su vida laboral a la programación de aplicaciones, a menudo, no se enfrentan al estudio de las asignaturas de arquitectura y de estructura de computadores suficientemente motivados, lo que les puede llevar a fracasar en estas materias. Con objeto de incrementar la motivación, se les puede demostrar que pueden mejorar las prestaciones de sus programas aplicando los conocimientos que adquieren en estas asignaturas. Para ser más convincentes, deberían obtener las mejoras ellos mismos, o se les debería mostrar resultados que estimen que pueden obtener ellos mismos, como por ejemplo, resultados logrados por otros estudiantes de la propia titulación. Otro objetivo que perseguimos es que los estudiantes tomen conciencia de que pueden usar estos conocimientos como argumentos a su favor a la hora de *competir por un trabajo* de programador con estudiantes de otras titulaciones superiores o estudios medios, en las que no se abarca el hardware de los computadores, o no se profundiza en su estudio, en la misma medida en la que se hace en las titulaciones de informática. Hay que tener en cuenta que las empresas de software que aprovechen las características de las arquitecturas disponibles, se verán beneficiadas dentro de un mercado competitivo.

Para alcanzar estos objetivos, por una parte, hemos propuesto trabajos fin de carrera, en los que los estudiantes comprueban que pueden obtener una buena relación entre prestaciones y coste de desarrollo, aplicando los conocimientos sobre arquitectura y estructura que han adquirido en la titulación ([6],[7]). Los resultados obtenidos en estos proyectos se muestran a los estudiantes de asignaturas de arquitectura y de estructura.

Por otra parte, se han introducido prácticas en asignaturas de arquitectura y de estructura en las que los estudiantes comprueban por sí mismos las mejoras en prestaciones que pueden conseguir aplicando contenidos que están estudiando (un trabajo de motivación similar ya se presentó en

[2]). En asignaturas de la materia troncal Tecnología y Estructura de Computadores de primer ciclo (Real Decreto 1459/1990 de 26 de octubre sobre *directrices generales propias* comunes de los planes de estudio de Ingeniero en Informática, BOE 20/11/1990), introdujimos una práctica sobre mejora de prestaciones en el curso 2002-2003, concretamente en la asignatura Estructura de los Computadores de 2º de Ingeniero Técnico en Informática de Sistemas. Aquí se presenta la práctica desarrollada en el curso 2004-2005.

Para mejorar prestaciones en una aplicación, se pueden utilizar tanto los conocimientos adquiridos sobre la *arquitectura abstracta* del procesador o ISA (repertorio de instrucciones, conjunto de registros, tipos de datos, modos de direccionamiento, lenguaje máquina y ensamblador), como los adquiridos sobre la *arquitectura concreta* o *microarquitectura* del procesador (implementación segmentada, superescalar, VLIW, SIMD, multihebra simultánea, etc.) y la arquitectura concreta de los sistemas de cómputo (multiprocesadores, multicomputadores, cluster). La arquitectura abstracta del procesador es objeto de estudio de asignaturas de la materia troncal de primer ciclo Tecnología y Estructura de Computadores, donde también se introduce la arquitectura concreta. Por otra parte, en las asignaturas de la materia troncal de segundo ciclo Arquitectura e Ingeniería de Computadores, se estudia en detalle la arquitectura concreta de los procesadores y de los computadores actuales.

Dado este reparto de contenidos entre estructura y arquitectura, las prácticas de mejora de prestaciones en asignaturas de arquitectura muestran al estudiante cómo incrementar prestaciones teniendo en cuenta la arquitectura concreta actual de los procesadores y computadores. Mientras que con la práctica para estructura que aquí se presenta se pretende que los estudiantes comprueben la utilidad de usar ensamblador en sus programas, aplicando así los conocimientos sobre arquitectura abstracta que adquieren en estructura: repertorios, tipos de registros, tipos de datos o modos de direccionamiento. Los estudiantes comprueban, con ejemplos muy sencillos, que resulta útil utilizar ensamblador para mejorar el *tiempo de ejecución* o también para obtener una *funcionalidad* que de otra forma no obtendrían. Resultados para aplicaciones completas se muestran al comienzo del curso a través de trabajos fin de carrera, como comentamos más arriba. Por otra parte, esta práctica familiariza al estudiante con otros conceptos abordados en la asignatura como el sistema de memoria y la segmentación del procesador.

En la Sección 2 se describe la práctica propuesta. La Sección 3 describe el trabajo realizado por los estudiantes e ilustra al profesor interesado en utilizar esta práctica sobre cómo usar los resultados como complemento a otros conceptos abordados en clases de teoría (sistema de memoria, segmentación). La Sección 4 muestra la repercusión de la práctica en la motivación de los estudiantes a través de dos encuestas. Por último, la Sección 5 presenta algunas conclusiones.

2. Descripción de la práctica.

En la Tabla 1 se puede ver el índice de la práctica. Se trata de la última práctica de un total de tres que realizan los estudiantes en la primera de las dos asignaturas de estructura impartidas en primer ciclo. Esta práctica tiene como objetivo adicional que los estudiantes aprendan el ensamblador usado tradicionalmente en Linux y la interfaz entre ensamblador y gcc/g++. En prácticas anteriores los estudiantes han utilizado la notación del ensamblador de Intel (usada tradicionalmente bajo DOS y Windows). Por este motivo, el apartado 3 del guión muestra las peculiaridades del ensamblador de Linux.

El apartado 2 del guión presenta al estudiante las herramientas que va a utilizar para desarrollar la práctica. Al buscar y elegir para la práctica un compilador para la sintaxis gcc/g++ y un entorno de desarrollo, se han tenido en cuenta varios criterios: (1) Gratuidad de las herramientas. (2) Compilador reciente. (3) Entorno de desarrollo que permita ejecutar paso a paso las instrucciones en ensamblador incluidas dentro del código en alto nivel. (4) Entorno de trabajo conocido por el estudiante, con el fin de evitar la pérdida de tiempo que supondría el aprendizaje de un nuevo entorno. Este año se ha optado por realizar las prácticas en Linux, ya que actualmente los estudiantes utilizan gcc/g++ en Linux en asignaturas de programación (en lugar de

Windows como en años anteriores). Para depurar se usa DDD (http://www.gnu.org/software/ddd/), que permite ejecutar paso a paso, además de instrucciones de alto nivel, las instrucciones en ensamblador.

Tabla 1. Índice de la práctica.

1. Resumen de objetivos.
2. Herramientas
3. Ensamblador de GNU.
3.1 Diferencias entre el ensamblador GNU y el de Intel.
3.2 Ensamblador en línea en gcc/g++.
4. Ejemplos de programas con ensamblador en línea.
5. Utilidad del ensamblador en línea.
6. Trabajo a desarrollar
6.1 Probar los programas de ejemplo del guión
6.2 Instrucciones de movimiento condicional
6.3 Instrucciones de manejo de cadenas
7. Referencias.

El apartado 4 del guión presenta ejemplos de programas gcc con ensamblador en línea que muestran errores habituales cuando se usa por primera vez la interfaz con gcc.

En asignaturas de estructura, como apoyo a sus contenidos, se suele estudiar en más profundidad el repertorio de instrucciones de algún procesador particular realizándose prácticas en las que se programa en ensamblador dicho procesador. En nuestras titulaciones se estudian como ejemplo los procesadores de la línea x86 tanto en asignaturas de estructura como de arquitectura. Hay varias razones que nos motivan a utilizar la línea x86: (1) Los alumnos, en el primer curso de la titulación, ya se han familiarizado con los conceptos de arquitectura abstracta estudiando un procesador sencillo [5]. (2) Son las arquitecturas disponibles en las aulas de prácticas, por lo que no se familiarizan con la arquitectura a través de simuladores. (3) La arquitectura abstracta de la línea x86 domina el mercado del PC (Athlon, Pentium 4, Pentium M), y se está extendiendo a otros mercados, como el de productos embebidos, de estaciones, de servidores (Xeon, Opteron) e incluso en supercomputadores (hay clusters en la lista de supercomputadores de www.top500.org basados en x86). (4) Destacamos también que Intel ofrece actualmente la arquitectura abstracta del 8086 para aplicaciones embebidas (para escáner, impresora, etc.) a través del 80186 (http://www.intel.com/design/intarch/intel186).

En el apartado 5 del guión se dan al estudiante argumentos a favor de usar los conocimientos que adquieren en estructura en sus programas de alto nivel. Deben tener en cuenta que hay instrucciones de los procesadores que los compiladores no generan o no lo hacen en todos los casos donde resultan rentables, o las generan en casos o de forma que no son rentables. Generalmente, los compiladores advierten al programador que debe comprobar si el código generado usando alguna opción de optimización realmente mejora prestaciones. Usar estas instrucciones puede ser rentable para disminuir tiempo de ejecución o para aprovechar una determinada funcionalidad. Dentro de estas instrucciones que mencionamos están las que se han incorporado recientemente a las arquitecturas, ya que los compiladores que aprovechan una arquitectura aparecen generalmente después de ésta, pero también hay instrucciones más antiguas.

Para poder beneficiarse de estas instrucciones el programador puede usar ensamblador o, si el ensamblador disponible no incluye la instrucción, código máquina. Otra alternativa para aprovecharlas es buscar bibliotecas de funciones que las utilicen, lo que puede encarecer el coste por tiempo (búsqueda, entrenamiento) y dinero (para adquirirlas inicialmente y mantenerlas) de desarrollo del software. Hay que tener en cuenta que las bibliotecas que aprovechan una arquitectura aparecen generalmente en el mercado después de la arquitectura y probablemente con un alto precio. Además, las funciones de biblioteca son generales, no proporcionan las mismas prestaciones que una implementación que se adapte específicamente a las necesidades de la aplicación. Por otro lado, algún informático debe programarlas.

Instrucciones de los procesadores de la línea x86 de Intel cuyo uso explícito en ensamblador (por el programador o a través de bibliotecas) nos pueden permitir reducir tiempo, teniendo en cuenta los compiladores actuales, son por ejemplo: las de manejo de cadenas (*movs, scas, cmps, stos*), de precaptación de memoria, instrucciones que saltan caches, instrucciones con procesamiento SIMD (MMX, SSE, SSE2, SSE3, 3DNow!), instrucciones de movimiento condicional (*cmov*) o instrucciones *setcc* ([3], [1]).

Otras instrucciones nos ofrecen alguna funcionalidad que podemos necesitar, como por ejemplo *xchg* o *rdtsc* en los procesadores de la línea x86. La instrucción *xchg* es útil para implementar primitivas de sincronización al permitir un acceso a memoria de lectura-modificación-escritura atómico. La instrucción *rdtsc* devuelve en 64 bits el número de ciclos de reloj desde que se inició el sistema, por lo que es útil para obtener el tiempo de ejecución con precisión de ciclos de reloj permitiéndonos así comparar la arquitectura concreta de los procesadores.

En el apartado 6 del guión ("Trabajo a desarrollar") se describe al estudiante el trabajo que ha de realizar. El trabajo se ha dividido en tres subapartados. En el primero el estudiante se familiariza con el ensamblador en línea dentro de gcc/g++ probando los ejemplos que hay en el guión de prácticas.

En el segundo, comprueba la utilidad de las instrucciones de movimiento condicional para mejorar prestaciones con dos ejemplos sencillos: (1) cálculo del mínimo de una lista, (2) limitar los elementos de una lista entre dos valores (usado, por ejemplo, en vídeo MPEG). El estudiante compara los ciclos de reloj que consumen estos ejemplos usando instrucciones de movimiento condicional (*cmov*), incluidas en la línea x86 a partir del PentiumPro (1995), y usando instrucciones de salto condicional. En Visual C++ de Microsoft se ha incorporado en la versión .NET 2003 la opción arch:SSE, que permite al compilador buscar puntos en los que resulta rentable usar la instrucción *cmov* además de instrucciones incluidas en el repertorio multimedia SSE [4]. En gcc esto se consigue usando opciones del compilador como -msse. No obstante, los compiladores no son capaces de usar estas instrucciones (incluida *cmov*) en todos los puntos donde resultan rentables. Por ejemplo, los compiladores mencionados son incapaces de generar cmov en el ejemplo que limita los componentes de una lista entre dos valores.

Con el tercer subapartado dentro del apartado 6, se ilustra la utilidad de las instrucciones de manejo de cadenas para mejorar prestaciones con tres ejemplos: (1) copiar de datos de 32 bits de una zona a otra de memoria, (2) inicializar una lista de componentes de 32 bits a un valor, y (3) buscar un dato de 32 bits en una lista. Las instrucciones de tratamiento de cadenas estaban ya incluidas en el repertorio del 8086. Hay funciones de biblioteca de los compiladores que se suelen implementar en ensamblador para utilizar estas instrucciones con el fin de mejorar prestaciones (por ejemplo, las funciones memchr, memcmp, memset o memcpy de la biblioteca string.h).

En todos estos ejemplos se mide el tiempo de ejecución en ciclos de reloj usando una macro basada en *rdtsc* que se da en el apartado 5 del guión.

3. Trabajo del estudiante

La práctica tiene asignada tres sesiones de dos horas, una sesión para cada subapartado de "Trabajo a desarrollar". La primera sesión es una sesión guiada en el que el profesor "obliga" a probar los ejemplos del guión dando puntos a los primeros alumnos en contestar a preguntas relacionadas con la ejecución de estos ejemplos. El objetivo principal de esta sesión es que los estudiantes se familiaricen con la sintaxis del ensamblador de Linux y con la interfaz entre gcc/g++ y ensamblador.

En las dos siguientes sesiones los estudiantes deben implementar, para cada una de las cuatro funciones de la Tabla 2, un código con ensamblador en línea dentro de código C que mejore las prestaciones de una versión que usa exclusivamente instrucciones en alto nivel. Además, realizan comparativas de tiempos de ejecución entre el ejecutable que genera el compilador a partir del fuente basado exclusivamente en instrucciones de alto nivel y el que genera a partir de la versión mejorada implementada por ellos mismos. En ningún momento los estudiantes han manifestado falta de tiempo para desarrollar la práctica, aún cuando la entregan trascurrida una semana desde la última sesión de la práctica.

Aquí presentaremos tiempos de ejecución que se pueden obtener en la práctica para dos de las funciones. Estos resultados permiten ilustrar: (1) en qué medida los alumnos observan que se pueden mejorar prestaciones usando, en ciertos puntos, ensamblador dentro de código de alto nivel; y (2) que el profesor puede usar la práctica para familiarizar al alumno sobre otros contenidos abordados en estructura. Estos resultados se han

obtenido compilado los códigos fuente con gcc 3.3.2 usando las opciones para optimización -O3 y -msse. Se dan resultados para dos computadores con diferentes procesadores: el Pentium 4 (P4) y el Pentium M (PM). Se recomienda a los estudiantes tomar datos en diferentes procesadores para poder así realizar comparativas entre arquitecturas concretas. En la Tabla 3 se pueden ver algunas características de los procesadores utilizados, entre ellos la familia (obtenida con la instrucción *cpuid*); un cambio de familia, frente a un cambio de modelo o actualización, supone un cambio sustancial en la arquitectura concreta del procesador.

Tabla 2. Funciones que realizan los códigos que mejoran los estudiantes.

6. Trabajo a desarrollar
6.2 Instrucciones de movimiento condicional
1) Limitar los componentes de una lista entre -256 y 255
6.3 Instrucciones de manejo de cadenas
2) Copia de datos entre listas
3) Inicializar a un valor una lista
4) Buscar un valor en una lista

Tabla 3. Procesadores utilizados.

Procesador	Familia	cache L1 datos	cache L1 inst.	cache L2
Pentium 4	15	8 KB	12Kμoper.	512KB
Pentium M	6	32KB	32KB	1MB

En todos los ejercicios, se trabaja con listas con componentes de 32 bits. El estudiante realiza ejecuciones para diferentes tamaños de lista: 16, 32, 64, 128, 256, 512, 1024, 2048 y 4096 componentes. Se llega a un tamaño en bytes de 16 KB. Así observan el comportamiento para tamaños pequeños y para tamaños grandes que incluso incluyen varias páginas de memoria (tamaño de páginas de 4 KB). Obtienen, para cada código a evaluar, el número de ciclos que consume su ejecución, y los ciclos que como media supone el procesamiento de una componente. Para cada tamaño obtienen la media de cuatro ejecuciones. Con estos valores rellenan una tabla, en la que también deben indicar la ganancia en tiempo de ejecución conseguida, para cada tamaño, con el código que usa ensamblador en línea frente al que no lo usa. Además, deben representar estos datos en gráficas, para poder así examinar/interpretar cómodamente los resultados.

En el subapartado 6.2 del guión, el estudiante primero comprueba la diferencia en prestaciones que supone para calcular el mínimo utilizar una instrucción de salto condicional frente a la utilización de una instrucción de movimiento condicional. En este ejemplo no debe implementar ningún código, se incluye para ilustrarle sobre cómo ha de implementar los códigos y las comparativas de prestaciones.

Por otra parte, en este subapartado del guión se les da código en alto nivel que limita los componentes de una lista entre -256 y 255, y se les pide que implementen código que mejore las prestaciones para esa función utilizando explícitamente instrucciones de movimiento condicional dentro del código gcc. (Tabla 2). En las Figuras 1 y 2 se pueden ver los resultados de ejecución en los dos procesadores de la Tabla 3 para: (1) el código para limitar que se da a los estudiantes (C), y (2) un programa gcc con ensamblador en línea (ASM) que limita cada componente con dos instrucciones de movimiento condicional en lugar de los saltos condicionales que genera el compilador. En estas gráficas se muestran, para cada tamaño, los ciclos que como media requiere el procesamiento de una componente (eje Y izquierda) y la ganancia en velocidad que se consigue (eje Y derecha) para los diferentes tamaños. En los dos programas se genera aleatoriamente la lista antes de pasar a limitar sus valores. Esto hace que la lista esté completa en la cache de nivel 2 antes de comenzar los cálculos; por tanto, no se observa la influencia ni de los fallos de cache en L2 ni del hardware de precaptación.

En el P4, el programa con ensamblador supone, para tamaños grandes, aproximadamente un 22,5% del tiempo del código generado por gcc (ganancia 4,4). Mientras que en el PM es aproximadamente un 28,5% (ganancia 3,5). Dado que cada etapa del cauce (*pipeline*) consume 1 ciclo, los estudiantes pueden percibir en las gráficas cómo se pone de manifiesto en los resultados el mayor número de etapas del cauce del P4 (tiene el doble de etapas que PM), especialmente para tamaños pequeños (para 16 componentes P4 consume 59,9 ciclos/componente frente a los 28 de PM).

En el tercer subapartado del apartado 6 del guión de prácticas (Tabla 2), los estudiantes escriben código ensamblador dentro de gcc en el que utilizan instrucciones de manejo de cadenas combinadas con prefijos de repetición (*rep*, *repnz*). Se pretende ilustrar a los estudiantes sobre el uso de estas instrucciones y sobre su utilidad para mejorar prestaciones. Utilizan en concreto: *rep movs* para copiar una lista de componentes de 32 bits en otra*, rep stos* para inicializar una lista de componentes de 32 bits a un valor, y *repnz scas* para encontrar un valor en una lista de 32 bits.

Como ejemplo, mostraremos aquí resultados para *rep stos.* El código de alto nivel que ejecutan los estudiantes se basa en:

```
for (i=0;i<num;i++) vector[i]=valor;
```

Para mejorar el código máquina generado por gcc utilizan *rep stos* a través de la siguiente sentencia *asm* (incluye la interfaz con gcc):

```
asm ( "movl %0,%%edi          \n\t"
      "movl %1,%%ecx          \n\t"
      "movl %2,%%eax          \n\t"
      "cld                    \n\t"
      "rep stosl              \n\t"
      ::"m"(vector),"m"(num),"m"(valor)
      :"%eax","%ecx","%edi","memory");
```

Es conveniente en este caso, que los estudiantes realicen tanto ejecuciones en las que los datos estén en cache L2 antes de pasar a ejecutar el código a evaluar, como ejecuciones en las que los datos no estén en cache; de esta forma pueden ver la influencia de los fallos de cache en las prestaciones, ilustrándose así lo comentado en clase de teoría sobre la necesidad de incluir caches en la jerarquía de memoria. En las Figuras 3 y 4 se pueden ver resultados obtenidos para este ejemplo en los procesadores de la Tabla 3. En cada gráfica se presentan los ciclos por componente para cuatro ejecutables. Hay dos ejecutables con código máquina generado por gcc (C y C-Ca), y dos ejecutables con parte del código fuente en ensamblador (ASM y ASM-Ca). En las versiones C-Ca y ASM-Ca los componentes de la lista se generan aleatoriamente antes de pasar al cálculo, la lista estará pues en la cache L2. En estas gráficas se representan también la ganancia para el código con *rep stos* sin la lista en L2 (C/ASM) y la ganancia con la lista en L2 (C/ASM-Ca).

Se debería indicar a los estudiantes que el hecho de que los datos estén o no en cache al realizar el cálculo, dependerá de la aplicación concreta en la que se utilice el código, del hardware de precaptación que incluya el procesador, y también de la utilización o no de instrucciones ensamblador de precaptación.

Con estos ejecutables se observa la efectividad del hardware de precaptación a L2 en los procesadores de la Tabla 3. Este hardware precapta datos a L2 en caso de que se detecte un acceso a memoria secuencial con salto constante. El mecanismo se dispara concretamente cuando se detecta en un acceso regular 2-4 fallos consecutivos. No obstante, la precaptación no se realiza a través de páginas; esto hace que en el momento en que se accede a datos de la lista que están en una nueva página de memoria virtual, el número de ciclos que supone la ejecución de los ejecutables sin la lista en L2 sufra un aumento debido a los fallos de cache. Para observar claramente esta subida, la lista se ha almacenado en los ejemplos cerca del comienzo de una página.

La ganancia en prestaciones de las versiones con la lista en L2 (Ca) respecto a las versiones en la que no está en L2 es poco pronunciada de 16 a 512 componentes, debido a la actuación del hardware de precaptación. Para 1024, 2048 y 4096, el incremento en ciclos de las versiones sin componentes en cache se debe a que el hardware no precapta al cruzar de página: en los tres casos se ha cambiado de página poco antes de terminar la ejecución. Los fallos de L2 ocultan la mejora de las instrucciones de manejo de cadenas.

Los resultados permiten que el estudiante perciba la necesidad de la jerarquía de memoria y la utilidad del hardware de precaptación incorporado en los procesadores para evitar la penalización en el acceso a memoria principal.

Las gráficas además reflejan la arquitectura segmentada de los procesadores. Conforme se aumenta el tamaño de las listas, se decrementa el número de ciclos que requiere el procesamiento de una componente (siempre que no se acceda a un bloque de memoria que no está en cache o que se encuentre en una nueva página). Efectivamente, conforme se aumenta el tamaño se va tendiendo a la productividad que proporciona el cauce segmentado del procesador para el código que se ejecuta cíclicamente (obsérvese que conforme aumenta el tamaño quedan más diluidas en el tiempo de ejecución la penalización por predicción errónea en la última ejecución de la instrucción condicional del bucle, el coste de las instrucciones de fuera y de inicio del bucle y el de

los primeros acceso a memoria hasta que se activa la precaptación automática).

Así pues, como reflejan las gráficas, los resultados que se obtienen en la práctica permiten que el estudiante perciba las ventajas en tiempo de ejecución que puede conseguir utilizando ensamblador en ciertos puntos de su programa en alto nivel. Además, estos resultados familiarizan al estudiante con otros conceptos que han abordado en la asignatura y que se amplían en asignaturas posteriores, como la jerarquía de memoria (paginación, cache) o la segmentación del procesador. Así, por ejemplo, observan los beneficios de la segmentación del procesador y la degradación de prestaciones si hay *paradas* en el cauce debidas a instrucciones de salto (en mínimo, limitar) o a esperas de datos (fallos de cache).

4. Motivación del estudiante

Se han pasado dos encuestas (Tabla 4) para comprobar si la práctica motiva al estudiante a estudiar contenidos de asignaturas de estructura por su aplicabilidad en la mejora de prestaciones del software. La primera encuesta se pasó (a un total de 65 estudiantes) una vez realizada y evaluada en clase la primera práctica de ensamblador. En el guión y en la presentación en clase de esta primera práctica se comentaba la utilidad de programar en ensamblador dentro de código de alto nivel para: (1) aprovechar aspectos de la arquitectura no explotados por los compiladores y (2) conseguir las prestaciones necesarias en algunas aplicaciones.

Para apoyar estos argumentos se presentó a los estudiantes un trabajo fin de carrera que mejora, teniendo en cuenta la arquitectura, un reproductor MP3 [7]. Los estudiantes observan que la primera versión del reproductor, implementada siguiendo exactamente el estándar, no permite una reproducción en tiempo real, y que aplicando sucesivas mejoras, incluido el uso puntual de instrucciones ensamblador, el reproductor se equipara en prestaciones a los disponibles comercialmente sin utilizar la tarjeta de sonido. Esto se muestra en clase ejecutando las diversas versiones del reproductor implementadas y viendo en directo su consumo de CPU.

La segunda encuesta se pasó (a un total de 60 estudiantes) una vez realizadas y evaluadas todas las prácticas (la última de las cuales es la aquí presentada). El tanto por ciento de estudiantes que contestaron "sí" a la primera cuestión (Tabla 4) y dieron una respuesta correcta a la segunda fue de un 32,3% en la primera encuesta, mientras que en la segunda subió a un 71,7%. Tal vez, habría que tener en cuenta además, que un 31,7% de estos 60 alumnos no aprobaron las prácticas. Así pues, el número de respuestas favorables subió una vez que los alumnos comprobaron *por sí mismos* a través de la práctica aquí presentada que pueden aplicar contenidos estudiados en estructura para mejorar su software.

Tabla 4. Cuestiones contestadas por el estudiante.

1) ¿Cree que podrá aprovechar parte de los contenidos de Estructura de los Computadores (por ejemplo, la programación en lenguaje ensamblador) para mejorar, de alguna forma, los programas que desarrolle en su vida laboral? SI/NO.
2) ¿Para qué cree que un informático utilizaría código ensamblador en algún punto de su programa en alto nivel (C/C++)?

5. Conclusiones

Para incrementar la motivación en el estudio de las materias de arquitectura y tecnología de computadores, proponemos a los estudiantes de informática prácticas en las que verifican *por sí mismos* que pueden mejorar las prestaciones de sus programas aplicando contenidos de estas materias. En la práctica para asignaturas de estructura que aquí presentamos, los estudiantes reducen el tiempo de ejecución en varios ejemplos sencillos (limitar los componentes de una lista, copiar, inicializar y buscar) usando conocimientos sobre arquitectura *abstracta* adquiridos en estructura.

Creemos conveniente que los estudiantes perciban ya en el primer ciclo de las titulaciones de informática (cursos de grado en las próximas titulaciones) la importancia de que un ingeniero que va a desarrollar software conozca el hardware actual para poder y saber aprovecharlo. Las encuestas reflejan que esta práctica ha incrementado de un 32,3% a un 71,7% el número de estudiantes que opinan que pueden aplicar para mejorar prestaciones los conocimientos que sobre arquitectura abstracta han adquirido en estructura.

Referencias

[1] "AMD Athlon™ Processor x86 Code Optimization Guide", http://www.amd.com.

[2] J. Flich, J. Real, J.C. Cano y J. Sahuquillo. "Prácticas de ensamblador". JENUI 2000, Alcalá de Henares. Madrid.

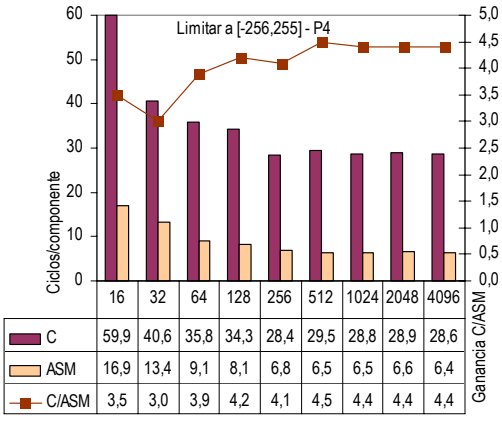
[3] "Intel® Pentium® 4 Processor Optimization Reference Manual", http://www.intel.com

[4] M. Lacey, "Optimizing Your Code with Visual C++" , Microsoft Corporation, 2003

[5] A. Prieto, F.J. Pelayo, F. Gómez-Mula, J. Ortega, A. Cañas, A. Martínez, F.J. Fernández. "Un computador didáctico elemental (CODE-2)", JENUI 2002, Cáceres.

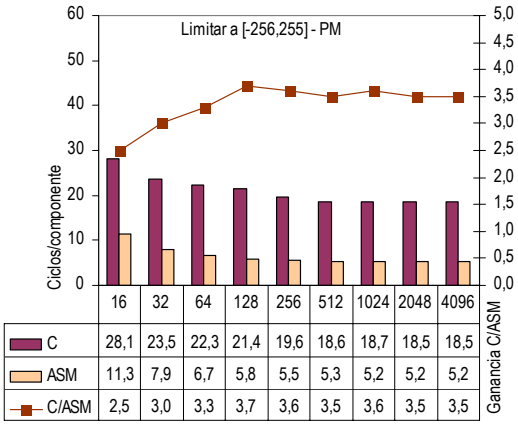
[6] Padilla, J.A; Anguita, M., Fernández, F.J.; Díaz, A.F.; Cañas, A.; Prieto, A. "Optimización de una implementación JPEG teniendo en cuenta la arquitectura actual de los procesadores", JENUI 2003, Cádiz.

[7] J. M. Martínez Lechado, M. Anguita. "Comparativa de implementaciones MP3 con diferente aprovechamiento de la arquitectura y complejidad algorítmica". *XIV Jornadas de Paralelismo*. Leganés, 2003.



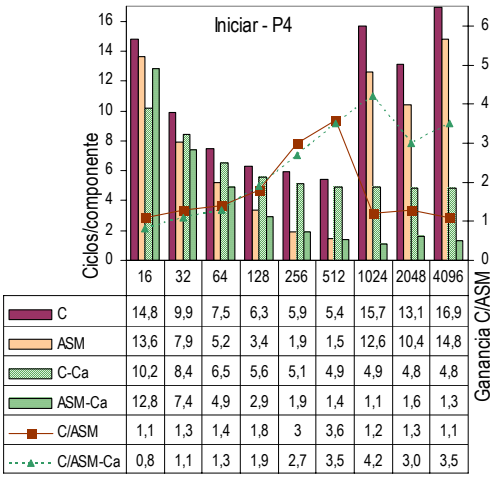
	16	32	64	128	256	512	1024	2048	4096
C	59,9	40,6	35,8	34,3	28,4	29,5	28,8	28,9	28,6
ASM	16,9	13,4	9,1	8,1	6,8	6,5	6,5	6,6	6,4
C/ASM	3,5	3,0	3,9	4,2	4,1	4,5	4,4	4,4	4,4

Figura 1. Limitar en Pentium 4



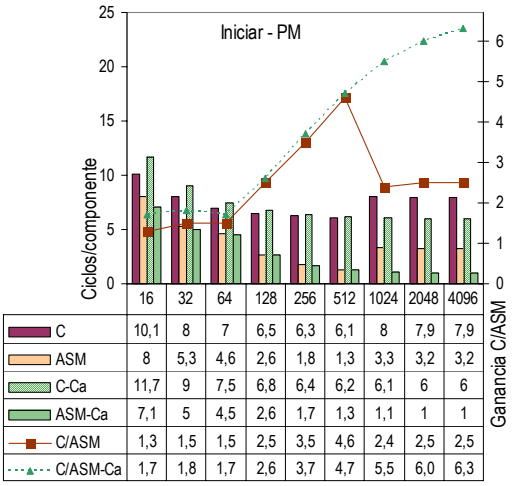
	16	32	64	128	256	512	1024	2048	4096
C	28,1	23,5	22,3	21,4	19,6	18,6	18,7	18,5	18,5
ASM	11,3	7,9	6,7	5,8	5,5	5,3	5,2	5,2	5,2
C/ASM	2,5	3,0	3,3	3,7	3,6	3,5	3,6	3,5	3,5

Figura 2. Limitar en Pentium M



	16	32	64	128	256	512	1024	2048	4096
C	14,8	9,9	7,5	6,3	5,9	5,4	15,7	13,1	16,9
ASM	13,6	7,9	5,2	3,4	1,9	1,5	12,6	10,4	14,8
C-Ca	10,2	8,4	6,5	5,6	5,1	4,9	4,9	4,8	4,8
ASM-Ca	12,8	7,4	4,9	2,9	1,9	1,4	1,1	1,6	1,3
C/ASM	1,1	1,3	1,4	1,8	3	3,6	1,2	1,3	1,1
C/ASM-Ca	0,8	1,1	1,3	1,9	2,7	3,5	4,2	3,0	3,5

Figura 3. Inicializar en Pentium 4



	16	32	64	128	256	512	1024	2048	4096
C	10,1	8	7	6,5	6,3	6,1	8	7,9	7,9
ASM	8	5,3	4,6	2,6	1,8	1,3	3,3	3,2	3,2
C-Ca	11,7	9	7,5	6,8	6,4	6,2	6,1	6	6
ASM-Ca	7,1	5	4,5	2,6	1,7	1,3	1,1	1	1
C/ASM	1,3	1,5	1,5	2,5	3,5	4,6	2,4	2,5	2,5
C/ASM-Ca	1,7	1,8	1,7	2,6	3,7	4,7	5,5	6,0	6,3

Figura 4. Inicializar en Pentium M

CEDI 2005

Software, Programación, Algoritmos y Fundamentos Teóricos

Experiencias en la realización de Estudios Empíricos en cursos de Ingeniería del Software

Félix García, Manuel Serrano, José A. Cruz-Lemus, Marcela Genero, Coral Calero, Mario Piattini
Grupo Alarcos
Escuela Superior de Informática
Universidad de Castilla-La Mancha
13071 Ciudad Real
{Felix.Garcia, Manuel.Serrano, JoseAntonio.Cruz, Marcela.Genero, Coral.Calero, Mario.Piattini}@uclm.es

Resumen

Los estudios empíricos en Ingeniería del Software son fundamentales para la validación de diversos métodos, técnicas, herramientas, etc., y los alumnos juegan un papel fundamental a la hora de llevarlos a cabo. Estos estudios no permiten obtener beneficios centrados exclusivamente en los aspectos de investigación, sino que es muy importante considerar también sus beneficios en la docencia. En este artículo se estudia la aplicación de experimentos controlados en cursos de Ingeniería del Software, destacando los beneficios que estos estudios aportan a los alumnos y a los docentes e investigadores que los llevan a cabo. Además, se presentan los resultados obtenidos en la realización de varios experimentos en cursos de ingeniería del software destacando los importantes beneficios pedagógicos obtenidos.

1. Introducción

Uno de los problemas de la Ingeniería del Software consiste en que a menudo se proponen una gran diversidad de métodos, lenguajes, herramientas, entornos, etc., de los que no se demuestra su utilidad práctica. El mercado competitivo actual, en el que se ha convertido el mundo del software, fuerza a las empresas a buscar la mejora de su calidad. Esta búsqueda supone en muchas ocasiones la adopción de nuevas tecnologías sin constancia de su utilidad práctica dejando de lado otras a pesar de que existen evidencias de su utilidad. Por lo tanto, resulta fundamental que los gestores de las empresas adopten un enfoque de "ingeniería del software basada en la evidencia" a la hora tomar decisiones que pueden resultar beneficiosas para el funcionamiento de la empresa [9].

Debido a esta necesidad, los métodos empíricos han centrado la atención de la comunidad científica en la Ingeniería del Software durante los últimos años. Mediante los métodos empíricos es posible evaluar nuevas aportaciones antes de que sean introducidas en los procesos software de las empresas [15]. Los estudios empíricos más comúnmente utilizados en la Ingeniería del Software son: experimentos controlados, casos de estudio y encuestas, los cuales difieren fundamentalmente en sus objetivos y restricciones. En el entorno académico los estudios empíricos más significativos tanto desde el punto de vista investigador como docente son los experimentos [1].

A la hora de llevar a cabo experimentos, los alumnos juegan un papel muy importante, ya que antes de realizar estos estudios en entornos industriales (lo que requiere un gasto significativo de tiempo, esfuerzo y recursos) en muchas ocasiones los investigadores llevan a cabo estudios piloto con alumnos en entornos académicos [5]. De hecho, hay que considerar que los alumnos constituyen la próxima generación de profesionales [17]. Por ello, los resultados de estos estudios en entornos académicos tienen una gran importancia y permiten obtener conclusiones significativas de cara a su realización posterior en entornos industriales. Bajo ciertas circunstancias, las diferencias entre los alumnos y los profesionales son pequeñas y las tareas requeridas en ciertos experimentos no requieren experiencia industrial, por ello se puede considerar la experimentación con alumnos como viable [14], [2].

Además, los estudios empíricos no sólo constituyen una aportación científica, sino que también proporcionan importantes beneficios pedagógicos en cursos de Ingeniería del Software, por lo que se establece una importante conexión entre la investigación y la docencia que es fundamental analizar [5].

Hoy en día los alumnos de cursos de Ingeniería del Software empiezan a participar en experimentos, sobre todo en universidades norteamericanas y británicas. Como resultado de ello una gran cantidad de métodos y técnicas han podido ser validadas empíricamente tal y como se refleja en numerosas publicaciones científicas. Sin embargo, muchas de estas publicaciones se centran en presentar los beneficios que los estudios han aportado a la investigación, dejando de lado a los alumnos, lo que incluso hace pensar en ocasiones, que los alumnos han sido "utilizados" de forma egoísta para obtener resultados en la investigación. Es fundamental abordar los beneficios que la experimentación aporta desde el punto de vista pedagógico y aportar estos beneficios a los alumnos cuando se planifican y se llevan a cabo experimentos en entornos académicos.

El principal objetivo de este trabajo es el de estudiar la aplicación de experimentos en cursos de Ingeniería del Software, destacando los beneficios que estos estudios aportan, tanto desde el punto de vista pedagógico como investigador.

En el siguiente apartado se identifican los distintos beneficiarios de los experimentos y las ventajas que estos estudios les aportan, sobre todo al investigador, al docente y a los alumnos. En el apartado 3 se describen las fases que hay que considerar a la hora de llevar a cabo experimentos. Los resultados de diversos experimentos llevados a cabo con alumnos en distintos cursos de Ingeniería del Software, se presentan en el apartado 4. Finalmente se presentan las conclusiones obtenidas y las consideraciones para el futuro.

2. Beneficiarios de la Experimentación en Ingeniería del Software

Cuando se desarrollan experimentos se pueden identificar cuatro beneficiarios principales de los resultados, que tendrán diferentes puntos de vista [5]:

- **Investigador**. Es el encargado de planificar y llevar a cabo el experimento. Su objetivo es demostrar la utilidad práctica de su propuesta u obtener conclusiones preliminares para realizar experimentos en entornos industriales.
- **Docente**. Es el profesor responsable de la asignatura o grupo de alumnos, que constituyen el contexto en el que se realiza el experimento. Su principal objetivo es enseñar los conocimientos y habilidades relacionadas con los estudios empíricos realizados y que van a ser de utilidad a los alumnos en su trabajo como futuros profesionales.
- **Alumnos**. Son los sujetos utilizados en el experimento. Su objetivo es aprender técnicas y habilidades que les puedan servir como futuros profesionales.
- **Empresa**. Las empresas de software son las beneficiarias últimas de los estudios empíricos. Como resultado de dichos estudios, las empresas pueden adoptar nuevos métodos o tecnologías que influirán en la mejora de sus procesos software y en definitiva les permitirán obtener beneficios económicos a medio o largo plazo.

En el contexto académico es de especial relevancia establecer las principales ventajas de los estudios empíricos desde el punto de vista del docente e investigador, que en muchas ocasiones son la misma persona, y sobre todo el beneficio que se aporta a los alumnos. En la Tabla 1 se resumen las ventajas más significativas de llevar a cabo estudios empíricos con alumnos de acuerdo al análisis realizado en [5]:

Tabla 1. Beneficios de los estudios empíricos en entornos académicos

Beneficiario	Beneficios
Docente	Nueva forma de formar a los alumnos respecto de la enseñanza tradicional
	Fomento de la participación en grupo de los alumnos en determinados estudios empíricos
	Mejora de la comunicación con los alumnos
	Nuevas formas de evaluación de los alumnos en situaciones en las que no tienen el estrés típico de un examen formal
	Introducir la ingeniería del software empírica como parte de la enseñanza en la ingeniería del software

Beneficiario	Beneficios
Investigador	Evidencia preliminar para aceptar o rechazar hipótesis
	Demostrar a las empresas software la relevancia de la investigación y la utilidad de llevar a cabo estudios empíricos en sus propias empresas
	Prever los recursos necesarios para realizar experimentos en entornos industriales y preparar el material necesario del experimento para realizarlo en la industria
	Formación de investigadores noveles en el desarrollo de estudios empíricos
Alumno	Formación en materias complementarias a la formación de grado
	Conciencia de nuevos problemas a resolver en el software en general y en las industrias en particular
	Mejor autoevaluación de cuál es su nivel en determinados temas de ingeniería del software que en las clases tradicionales más centradas en los aspectos teóricos
	Percepción de las ventajas de usar métodos empíricos en la ingeniería del software
	Preparación para su futuro profesional en el que en muchas ocasiones serán sometidos a cuestionarios, informes, encuestas, etc.

En la práctica, es muy importante conocer los efectos negativos de llevar a cabo experimentos en entornos académicos, con el fin de que la planificación de los experimentos sea suficientemente cuidadosa para que estos efectos no se produzcan [5]:

- Desde el punto de vista del **investigador** los inconvenientes de realizar estudios empíricos con alumnos son el esfuerzo necesario para prepararlos, ya que nunca deben olvidarse de los aspectos pedagógicos al preparar el estudio. Otro problema son las amenazas a la validez del estudio, tema que se aborda brevemente en el apartado 3.
- Desde el punto de vista del **docente** los problemas que pueden surgir son la necesidad de motivar al docente para llevar a cabo el experimento, ya que le supone un esfuerzo de formación mucho mayor para esa clase que si se tratara de una clase normal y además debe motivar a los propios alumnos creando el ambiente necesario en la clase. El docente debe tener capacidad de atender cualquier duda de los alumnos. Estas necesidades se satisfacen en gran medida si el docente y el investigador son la misma persona.
- Desde el punto de vista de los **alumnos** pueden surgir problemas derivados de pérdida de tiempo, si se trata de experimentos que requieren una formación extensa que les hace perder varias clases en lugar de aprovecharlas formándoles en temas más interesantes o útiles para su futuro profesional. Otro problema potencial es que el resultado del experimento demuestre que la técnica o método que acaban de aprender no es efectivo, aunque en este caso también se puede encontrar la parte pedagógica, consistente en demostrarles que una tecnología o técnica no se puede aceptar sin evaluarla en la práctica por muy nueva o prometedora que sea. Además, en este último caso se pueden intentar hallar las causas del resultado.

En definitiva, es fundamental considerar las ventajas e inconvenientes comentados en la planificación de estudios empíricos, con el fin de que las ventajas sean percibidas por los alumnos y se puedan evitar en la medida de lo posible los problemas que de ello se puedan derivar, como la desmotivación o el descontento. Los autores de este artículo han llevado a cabo como docentes/investigadores experimentos con alumnos en diferentes cursos de la Escuela Superior de Informática de Ciudad Real. Los resultados de dichos experimentos centrados en los aspectos más pedagógicos se describen en el apartado 4. A continuación se describen de forma resumida las principales características y aspectos a considerar a la hora de llevar a cabo experimentos controlados con alumnos.

3. Proceso Experimental

A la hora de realizar experimentos controlados, hay que considerar una serie de factores esenciales para conseguir una buena planificación y un desarrollo satisfactorio, con el fin de obtener resultados que sean creíbles y útiles [16], [23],[18], [3], [17].

Las ventajas de los experimentos es que pueden determinar las situaciones en las que

ciertas afirmaciones son verdaderas y pueden proporcionar el contexto en el que ciertos estándares, métodos y herramientas son recomendables. Sólo si el experimento se realiza adecuadamente, es posible obtener conclusiones sobre las hipótesis planteadas.

Los experimentos requieren ser planificados cuidadosamente si queremos que nos proporcionen resultados útiles y significativos. Por ello es necesario seguir un proceso experimental como el que se propone en [23] que consta de las siguientes etapas: **Definición**, en el que se define el experimento en términos del problema y los objetivos; **Planificación**, donde se determina el diseño del experimento y la instrumentación del mismo; **Operación**, en la que se lleva a cabo el experimento y se recogen los datos empíricos; **Análisis e interpretación**, donde se analizan e interpretan los datos recogidos utilizando técnicas estadísticas; **Evaluación de la validez**, en la que se evalúan los aspectos que pueden amenazar a la validez del experimento (validez de constructo, validez interna, validez externa, validez de la conclusión); y **Presentación y difusión**, en la que se elabora un informe sobre los resultados para facilitar que otros investigadores puedan replicar el experimento.

4. Experiencias de Experimentos con Alumnos en Cursos de Ingeniería del Software

4.1. Trabajos Previos

En este apartado se presentan de forma resumida los resultados de tres experimentos controlados llevados a cabo en asignaturas de Ingeniería del Software y que constituyen los trabajos previos a los presentados en el presente artículo [12]. Los experimentos tuvieron lugar en horario normal de clase, su realización fue voluntaria y se motivó especialmente a los estudiantes para su realización, destacando los beneficios que los experimentos les proporcionan como futuros profesionales. Para conseguir evitar las posibles amenazas a la validez de los experimentos,

- Los sujetos tenían una experiencia y unos conocimientos parecidos.
- Los dominios de los diagramas y modelos del material experimental eran suficientemente sencillos y comunes para facilitar su entendimiento.
- Los esquemas fueron entregados a cada sujeto en un orden diferente para evitar efectos de aprendizaje.
- Los sujetos realizaban por primera vez el experimento (efectos de la persistencia atenuados).
- Los sujetos estaban motivados, dado que los ejercicios formaban parte de los conocimientos que debían adquirir en su formación.
- No se permitió que los sujetos hablaran entre ellos ni que pudieran copiar los resultados unos de otros.
- Todas las dudas fueron resueltas por la persona que supervisaba el experimento.
- Los sujetos no tenían conocimiento, a priori, de los aspectos que se pretendían estudiar ni cuales eran las hipótesis que se habían planteado.

En la tabla 2 se resumen los experimentos llevados a cabo previamente y los resultados obtenidos.

La experiencia obtenida en la realización de experimentos previos ha sido utilizada para la planificación y ejecución de nuevos experimentos con alumnos de ingeniería del software en los que se ha hecho hincapié en los aspectos pedagógicos, sólo llevando a cabo aquellos experimentos acordes con los contenidos de la asignatura y formando a los alumnos en la importancia de los métodos empíricos. Estos experimentos se describen brevemente en el apartado 4.2.

4.2. Nuevas Experiencias

A la hora de llevar a cabo los experimentos y con el fin de evitar en lo posible las amenazas a su validez se han adoptado las mismas medidas descritas en el apartado 4.1.

Tabla 2. Experiencias Previas en la Realización de Experimentos en cursos de Ingeniería del Software

	Métodos de Diseño en Almacenes de Datos [20]	**Modelos de Procesos Software** [10, 11]	**D. Clases UML** [13]
Objetivo	Determinar si los **diseños de almacenes de datos** basados en diagramas de **estrella** son más comprensibles que aquellos realizados basándose en el **diseño tradicional** de **bases de datos**	Establecer la relación entre la **complejidad estructural** de los **modelos de procesos** y su **mantenibilidad** (entendibilidad y modificabilidad)	Averiguar si existe correlación entre la complejidad estructural y el tamaño de los diagramas de clases UML y la entendibilidad y mantenibilidad de los mismos
Participantes	**11 alumnos** de la Escuela Superior de Informática (ESI) de Ciudad Real (UCLM)	**45 alumnos**, 3º curso de Ingeniería Técnica en Informática de Gestión (ESI) y **41 alumnos** 3º curso de Ingeniería Técnica en Informática de Sistemas (ESI)	**24 alumnos** 3º curso de Ingeniería en Informática (ESI)
Material	**3 Diagramas** diseñados según el **diseño tradicional** de bases de datos **3 Diagramas en estrella** (equivalentes semánticamente a los anteriores)	**10 Modelos de Procesos** software representados con el lenguaje de modelado **SPEM** (*Software Process Engineering Metamodel*) [22].	9 Diagramas de clases UML
Modo de Operación	**Realizar consultas** sobre los diagramas del material utilizando **SQL**	En 5 modelos: **Responder 5 cuestiones** relacionadas con los modelos En los 5 modelos restantes: **Realizar 4 modificaciones** en base a nuevos requisitos	**Contestar** el cuestionario adjunto (cinco preguntas) a cada diagrama **Modificar** cada diagrama de clases (cuatro nuevos requisitos)
Datos Analizados	**Tiempos** empleados en realizar las **consultas SQL**	**Tiempos de respuesta** (entendibilidad) y de **modificación** de los modelos (modificabilidad)	**Tiempos de respuesta** (entendibilidad) y de **modificación** de los diagramas (mantenibilidad)
Conclusiones	No hay diferencia en la comprensión de los esquemas debido al método de diseño utilizado y que por tanto es equivalente diseñar un almacén de datos utilizando una metodología tradicional o los diagramas en estrella.	Algunas métricas definidas para evaluar la complejidad estructural de los modelos eran –en cierta manera– válidas, y pueden ser utilizadas como indicadores del tiempo de entendimiento y modificación de los modelos de procesos software	La mayoría de las métricas definidas (complejidad estructural de los diagramas UML) eran –en cierta manera– válidas, y pueden ser utilizadas como indicadores del tiempo de entendimiento y mantenimiento de los diagramas de clases UML
Beneficios Pedagógicos	Los alumnos adquirieron experiencia sobre la realización de experimentos para demostrar empíricamente la utilidad de una técnica	Se impartió un tema especial de Proceso Software en la asignatura de Ingeniería del Software (3º). Ello permitió a los alumnos tener una visión más amplia de la asignatura y aprendieron a modelar procesos	Los alumnos obtuvieron cierta experiencia en el entendimiento y modificación de diagramas de clases UML tareas que podrían tener que realizar tanto en los exámenes como en el desempeño de la profesión
	Los alumnos aprendieron a dudar de los prejuicios y a analizar las causas de que los resultados obtenidos no fueran los esperados ya que los diagramas en estrella son ampliamente utilizados y aceptados para diseño de almacenes de datos	Se dedicó una clase a mostrar los resultados dónde percibieron la importancia de realizar estudios empíricos en Ingeniería del Software	Se les explicó a los alumnos el objetivo de investigación, las métricas definidas, las hipótesis planteadas y los resultados obtenidos. Ello complementó su formación con el aprendizaje de estudios empíricos y medición del software

a) Experimento sobre "Pair-Designing"

Definición

Uno de los aspectos relevantes a estudio en el ámbito de la Ingeniería del software es la transferencia de conocimiento cuando se trabaja en pares. Dentro de las técnicas que favorecen esta transferencia de conocimiento se encuentra la denominada "Pair programming" que es una práctica utilizada en metodologías ágiles consistente en que dos programadores trabajan "hombro con hombro" (*side by side*) en el desarrollo de una misma pieza de código. Un programador, que asume el rol de conductor ("driver") escribe activamente el código mientras que el otro que asume un rol de meramente observador ("observer") identificando defectos y aspectos tácticos y estratégicos. Los roles se intercambian periódicamente. Un beneficio de esta práctica es que se refuerza el incremento de conocimiento de los participantes, en concreto el conocimiento tácito. Esta práctica podría ser aplicada al diseño con los mismos beneficios denominándose en este caso "Pair Designing" en el que el "driver" edita activamente el documento de diseño mientras que el "observer" realiza una revisión continua. Con todo ello se planificó un experimento cuyo objetivo fue demostrar la relación existente entre la aplicación de la práctica "pair designing" y la construcción de conocimiento [4].

Planificación

El experimento se realizó con 42 alumnos de ingeniería técnica en informática de gestión, 39 de ingeniería técnica en informática de sistemas de tercer curso y 12 alumnos de 5º curso de la Escuela Superior de Informática de Ciudad Real. Con el fin de evaluar la construcción de conocimiento durante la aplicación de la técnica "pair designing" los sujetos tenían que mejorar el diseño de un sistema existente en UML. Todos los sujetos tenían conocimientos de modelado del producto con UML pero no sobre metodologías ágiles y en concreto sobre la técnica "pair programming". Por ello se preparó una clase especial en la que se impartieron los conceptos relacionados con las metodologías ágiles y técnicas relacionadas haciendo hincapié en el aprendizaje del "pair programming". Se realizó una pequeña prueba en la que los alumnos resolvieron un ejercicio de especificación de casos de uso aplicando "pair designing". Tras el ejercicio se dedicaron unos veinte minutos a recabar y debatir las opiniones de los alumnos sobre las ventajas e inconvenientes de la técnica. Los alumnos participaron muy activamente en la iniciativa.

El diseño del sistema que los sujetos tenían que mejorar consistía en: una especificación textual con los requisitos, dos diagramas de casos de uso y dos diagramas de clases (un total de 15 clases). Las tareas de mantenimiento que los sujetos debían realizar fueron básicamente dos:

- **Reducir la complejidad**, eliminando entidades (casos de uso, actores, clases, atributos, métodos) o relaciones no significativas para el entendimiento y cumplimiento de requisitos del diseño existente;
- **Mejorar la legibilidad**, modificando las entidades existentes o añadiendo nuevas.

Los sujetos fueron distribuidos equitativamente o de forma individual (32 sujetos) o por parejas (64 sujetos). Para evaluar la variable dependiente (conocimiento del sistema) se prepararon dos cuestionarios QA y QB. Ambos fueron distribuidos a la entrada (antes de iniciarse el ejercicio) y a la salida (tras la mejora del diseño). Cada sujeto recibía un cuestionario a la entrada QA o QB y a la salida recibía el otro cuestionario. Para evaluar los cuestionarios se contaban las respuestas correctas. El proceso experimental a seguir fue:

- Cada sujeto examinaba la documentación de forma individual durante 20 minutos.
- Cada sujeto respondía un cuestionario de entrada durante 10 minutos. El objetivo de este cuestionario era establecer la línea base, es decir, el nivel de conocimiento del sistema antes de trabajar sobre él.
- Los sujetos distribuidos en pares (aplicando "pair designing") y los sujetos distribuidos de forma individual realizaban las tareas de mantenimiento requeridas durante 80 minutos.
- Cada sujeto respondía un cuestionario de salida (10 minutos) para evaluar la construcción de conocimiento tanto para los que trabajan de forma individual como para las parejas.

Operación

Previamente a la realización del experimento se impartió una breve charla en la que se explicó el proceso experimental.

Análisis e Interpretación

Para analizar si el crecimiento de conocimiento era significativo cuando los sujetos trabajaban aplicando "pair designing" respecto a los que trabajaban de forma individual se aplicó el test estadístico de "Mann Whitney". Como resultado se obtuvo que el crecimiento de conocimiento se refuerza de forma significativa al trabajar en pares aplicando "pair designing" respecto a los que trabajan de forma individual. Además este experimento permitió realizar una comparativa con los resultados de un experimento anterior obteniéndose una nueva conclusión de que el crecimiento de conocimiento en pares es más significativo respecto al individual cuando los sujetos emparejados tienen parecido nivel educativo ("background").

Conclusiones

Desde el punto de vista del investigador se encontraron evidencias de que formar pares con individuos del mismo "background" educativo enfatiza los beneficios de la técnica "pair designing" respecto al crecimiento de conocimiento. Comparando los resultados con experimentos previos también se llegó a la conclusión de que agrupar a una persona con "background" científico con otro de "background" no científico no favorece significativamente el crecimiento de conocimiento e incluso podía empeorar los conocimientos del primero.

Desde el punto de vista pedagógico, los resultados del experimento pueden considerarse interesantes para el sistema educativo en las universidades. Tal y como se trata de motivar en nuevas regulaciones como Bolonia, la educación está experimentando un giro del estilo de enseñanza tradicional a un estilo de aprendizaje en el que los alumnos aprenden por sí mismos con la guía de los profesores. En la planificación de este tipo de educación se podrían aplicar prácticas del estilo de "pair designing" para incrementar en la mayor medida de lo posible el crecimiento de conocimiento.

Desde el punto de vista de los alumnos el experimento fue una experiencia muy enriquecedora para ellos:

- Aprendieron un tema adicional a su formación reglada como son las metodologías ágiles y sus técnicas asociadas. Ello les beneficia como futuros profesionales ya que este tipo de enfoques está adquiriendo un auge creciente en las empresas software.
- Tras la realización del experimento se les explicó brevemente los objetivos perseguidos y los resultados, con lo que contrastaron sus opiniones previas sobre la validez de la práctica de "pair designing" con su propia experiencia en el experimento y con los resultados empíricos. Ello les aporta una mayor capacidad crítica a la hora de adoptar nuevos métodos o técnicas sin antes demostrar su validez.

b) Experimento sobre estados compuestos en diagramas de estados UML

Definición

Los diagramas de estados de UML se han convertido en una técnica muy importante a la hora de describir el comportamiento dinámico de un sistema software. En trabajos previos [7] ya habíamos definido un conjunto de métricas para evaluar las propiedades estructurales de los diagramas de estados de UML y las habíamos validado como indicadores tempranos a través de una famila de experimentos [8].

Estos experimentos también habían revelado que, aparentemente, los estados compuestos no influían en la entendibilidad de los diagramas. Este hecho nos parecía un tanto sospechoso por lo que decidimos ir un paso más allá y realizar un experimento controlado y una réplica del mismo centrándonos en el efecto que los estados compuestos tienen sobre la entendibilidad de los diagramas de estados de UML.

Planificación

El experimento fue realizado por 55 estudiantes de Ingeniería Informática de la Universidad de Murcia y la réplica por 178 estudiantes de las

distintas titulaciones (Ingeniería Informática e ingenierías técnicas) de la Universidad de Alicante.

Cabe destacar que la experiencia previa que atesoraban los primeros era notablemente superior a la de los segundos, ya que en las asignaturas relacionadas con la Ingeniería del Software que habían cursado, mientras los sujetos del experimento habían estudiado los diagramas de estados de UML a fondo (incluyendo los estados compuestos), los sujetos de la réplica sólo habían trabajado someramente con los diagramas de estados de UML y no habían estudiado aún los estados compuestos.

Cada sujeto recibía dos diagramas, uno modelado usando estados compuestos y otro sin usarlos, pertenecientes además a dos dominios distintos. Para cada diagrama tenía que responder un conjunto de seis cuestiones para comprobar si había entendido el modelo, anotando el instante en que comenzaban y terminaban de responder el cuestionario.

Como variable independiente tomamos la presencia o no de estados compuestos al modelar los diagramas, mientras que nuestra variable independiente era la eficiencia de los diagramas, medida como la relación entre el número de respuestas correctas y el tiempo invertido en responderlas.

Operación

Antes de que los sujetos realizaran las tareas que se les solicitaban, se les impartió una breve charla en la que se les explicaban las principales características de los diagramas de estados de UML, además de resolver algunos ejemplos del estilo a los que se iban a encontrar durante la realización del experimento.

Análisis e Interpretación

Se utilizó un diseño factorial con interacción confundida para realizar un análisis de la varianza. Este análisis, junto al de los estadísticos descriptivos de la variable dependiente produjeron los siguientes resultados:

- En el experimento, obtuvimos que la utilización de estados compuestos mejoraba notablemente la entendibilidad de los diagramas, ya que los sujetos obtenían mucho mejores resultados.
- En la réplica, las diferencias entre aquellos diagramas que se modelaban usando estados compuestos y los que no, eran muy pequeñas en cuanto a la eficiencia que los sujetos mostraban al entenderlos. De hecho, los resultados para los diagramas modelados sin estados compuestos eran ligeramente mejores. En cualquier caso, en análisis de la varianza de nuevo indicaba que el uso de estados compuestos en diagramas de estados UML está fuertemente relacionado con la entendibilidad de los mismos.

Conclusiones

Como principal conclusión, confirmamos nuestras sospechas de que los estados compuestos son un importante elemento a tener en cuenta al medir la entendibilidad de los diagramas de estados UML.

Un hallazgo que consideramos muy importante es que cuando alguien se enfrenta a la tarea de comprender un diagrama de este estilo, la presencia de estados compuestos ayuda a que la tarea se haga de un modo más eficiente, siempre y cuando se tenga un conocimiento previo del uso de los mismos.

De ahí que consideremos que es ampliamente recomendable que en aquellas asignaturas de Ingeniería del Software que se dediquen al estudio de UML se detengan en el trabajo con estados compuestos cuando se estén estudiando los diagramas de estados.

5. Conclusiones

En este artículo se ha abordado la realización de experimentos en entornos académicos, destacando las ventajas que estos estudios aportan tanto desde el punto de vista del docente/investigador como del punto de vista del alumno. Además, se han presentado los resultados obtenidos en la realización de 2 experimentos en cursos de ingeniería del software en los que se han aplicado las lecciones aprendidas en experimentos previos y que han permitido obtener diversos beneficios pedagógicos.

Uno de los aspectos en los que se ha centrado la planificación, ejecución y comunicación de resultados de los experimentos ha sido la

formación preliminar del alumno en métodos empíricos como una de los paradigmas en ingeniería del software que convierten esta disciplina en una "ciencia" en el que hay que adoptar un enfoque basado en la evidencia a la hora de proponer o adoptar nuevos métodos y tecnologías. Precisamente, lo que distingue la "ciencia" del "arte" es el modo en que los gestores y trabajadores de la empresa toman decisiones en base a argumentos racionales basados en la evidencia obtenida a partir de la experiencia y la investigación y esta visión no es particular a la ingeniería del software, sino que caracteriza a una buena ciencia en general [19]. Siempre que sea posible en asignaturas relacionadas con ingeniería del software se debe inculcar esta visión de la ingeniería a los alumnos, y la realización de experimentos en entornos académicos es una buena forma de conseguirlo.

El entusiasmo mostrado por la mayoría de los alumnos al realizar el experimento y su interés por conocer los resultados obtenidos nos lleva a pensar que este tipo de experimentos son muy beneficiosos y se deberían realizar siempre que sea posible en cursos de Ingeniería del Software, tal y como ya se está realizando en otros centros de estudio internacionales [6], [15].

No obstante, a la hora de llevar a cabo experimentos en cursos de ingeniería del software siempre hay que considerar los beneficios que se pueden aportar a los alumnos y además hay que tener en cuenta diversas consideraciones éticas [21], para que como resultado del experimento no se produzca desmotivación y descontento por parte de los alumnos.

Referencias

[1] Baresi, L., Morasca, S. and Paolini, P. Estimating the Design Effort of web Applications. Proceedings of the 9th International Software Metrics Symposium (METRICS'03), pp. 62-71, 2003.

[2] Basili V., Shull F. and Lanubile F. Building Knowledge Through Families of Experiments. IEEE Transactions on Software Engineering, Vol. 25 Nº 4, pp. 435-437, 1999.

[3] Briand, L., Arisholm, S., Counsell, F., Houdek, F. and Thévenod-Fosse, P. (2000). Empirical Studies of Object-Oriented Artefacts, Methods, and Processes: State of the Art and Future Directions. Empirical Software Engineering, Vol. 4 Nº 4, pp. 387-404, 2000.

[4] Canfora, G., Cimitile, A., Garcia, F., Piattini, M., Visaggio, C. Confirming the influence on educational background in pair-design knowledge through experiments. In Proceedings of The 20th Annual ACM Symposium on Applied Computing (SAC 2005), Santa Fe (New Mexico).

[5] Carver, J., Jaccheri, L., Morasca. S. y Shull. F. Issues in Using Students in Empirical Studies in Software Engineering Education. Proceedings of the 9th International Software Metrics Symposium (METRICS'03), pp. 239-251, 2003.

[6] Carver, J., Jaccheri, L., Morasca. S. y Shull. F. Using Empirical Studies during Software Courses. Experimental Software Engineering Research Network 2001-2003. LNCS 2765, pp. 81-103, 2003.

[7] Cruz-Lemus, J. A., Genero, M. and Piattini, M.: Metrics for UML Statechart Diagrams. In: Metrics for Software Conceptual Models. Genero, Piattini and Calero (eds.), Imperial College Press, UK, 2005.

[8] Cruz-Lemus, J. A., Maes, A., Genero, M., Poels, G. and Piattini, M.: Analyzing Data Extracted from a Family of Experiments for Evaluating UML Statechart Diagrams Understandability. Research Working Paper, University of Ghent (to appear) (2005)

[9] Dyba, T., Kitchenham, B. and Jorgensen, M. Evidence-Based Software Engineering for Practitioners. IEEE Software, pp. 58-65, 2005.

[10] García, F., Ruiz, F. y Piattini, M. "Medición del Proceso Software", VIII Jornadas de Ingeniería del Software y Bases de Datos, Actas de las Jornadas E. Pimentel, N. Brisaboa, J. Gómez (eds), Alicante (España), pp. 303-314, 2003.

[11] García, F., Ruiz, F. y Piattini, M. An Experimental Replica to Validate a set of Metrics for Software Process Models. European Software Process Improvement Conference, Lecture Notes in Computer Science (LNCS 3281), pp. 146-158, 2004.

[12] García, F., Serrano, M., Cruz-Lemus, J.A, Genero, M., Calero, C. y Piattini, M. La Experimentación en la Docencia de Ingeniería del Software. Proceedings de las X Jornadas

de Enseñanza Universitaria de la Informática (JENUI), pp. 237-245, 2004.

[13] Genero, M. Defining and Validating Metrics for Conceptual Models. Ph.D. Thesis Department of Computer Science, University of Castilla-La Mancha, 2002.

[14] Höst, M., Regnell, B. and Wholin, C. Using Students as Subjects – A comparative Study of Students & Professionals in Lead-Time Impact Assessment". 4th Conference on Empirical Assessment & Evaluation in Software Engineering (EASE), Keele University, UK, pp. 201-214, 2000.

[15] Höst. M. Introducing Empirical Software Engineering Methods in Education. Proceedings of the 15th Conference in Software Education and Training (CSEET'02), pp. 170-179, 2002.

[16] Juristo, N. and Moreno, A. Basics of Software Engineering Experimentation. Kluwer Academic Publishers, 2001.

[17] Kitchenham, B, Pfleeger, S., Pickard, L., Jones, P., Hoaglin, D., El Emam, K. and Rosenberg, J. Preliminary Guidelines for Empirical Research in Software Engineering. IEEE Transactions on Software Engineering, Vol. 28 Nº 8, pp. 721-734, 2002.

[18] Perry, D., Porter, A. and Votta, L. Empirical Studies of Software Engineering: A Roadmap, Future of Software Engineering, ACM, Ed. Anthony Finkelstein, 2000, pp. 345-355.

[19] Pfleeger, S.L Soup or Art? The Role of Evidential Force in Empirical Software Engineering. IEEE Software, pp. 66-73, 2005.

[20] Serrano, M., Calero, C. and Piattini, M. An Empirical Study with Datawarehouse Design Methods, 1st International Workshop Empirical Studies in Software Engineering, Bunse, C., Jedlitschka, A. (eds), pp. 49-54, Finland, 2002.

[21] Singer, J. and Vinson, N. Ethical Issues in Empirical Studies of Software Engineering. IEEE Transactions on Software Engineering, Vol. 28 Nº 12, pp. 1171-1180, 2002.

[22] Software Process Engineering Metamodel Specification; adopted specification, version 1.0. Object Management Group. November (2002). Available in http://cgi.omg.org/cgi-bin/doc?ptc/02-05-03.

[23] Wohlin C., Runeson P., Höst M., Ohlson M., Regnell B. and Wesslén A. Experimentation in Software Engineering: An Introduction. Kluwer Academic Publishers, 2000.

Programación Declarativa utilizando XML, representaciones gráficas y mundos virtuales infinitos

Jose Emilio Labra Gayo
Dpto. de Informática
Universidad de Oviedo
CP. 33007 Oviedo, Spain
labra@uniovi.es

Resumen

En la asignatura *Programación Declarativa* se ha incorporado la utilización de vocabularios XML para generar representaciones gráficas, *quadtrees* y *octrees* para facilitar la enseñanza de las principales características de los lenguajes declarativos: funciones de orden superior, evaluación perezosa, polimorfismo, variables lógicas e indeterminismo. La utilización de vocabularios XML estándar: SVG para gráficos y X3D para realidad virtual permite disponer de numerosas herramientas de visualización. En este artículo se presenta un nuevo esquema de prácticas que incorpora estas tecnologías mostrando aplicaciones reales de los lenguajes declarativos y aumentando la motivación de los estudiantes.

1. Introducción

La asignatura *Programación Declarativa* se imparte como optativa de 6 créditos en las titulaciones de Ingeniero Técnico de Informática de Gestión y de Sistemas en la Universidad de Oviedo. El resto de asignaturas de programación de dicha titulación se centra en lenguajes imperativos y orientados a objetos: Java, C y C++. Esta asignatura es, por tanto, el primer y en muchas ocasiones, único contacto que los estudiantes de estas titulaciones tienen con los lenguajes declarativos. Dado su carácter optativo, la supervivencia de la asignatura depende del número de alumnos matriculados. Aparte de la dificultad de una asignatura de programación, la elección de los estudiantes se ve condicionada por otros aspectos relacionados con este tipo de lenguajes: entornos de programación rudimentarios, escasez de sistemas gráficos de depuración y traza, carencia de librerías, dificultad para enlazar con librerías escritas en otros lenguajes, etc. Con el fin de ampliar la motivación de los estudiantes y el alcance de la asignatura, en el curso 2001/02 se incorporó al proyecto IDEFIX [17,18] que perseguía la enseñanza a través de Internet de lenguajes de programación.

La evaluación de la asignatura se realiza fundamentalmente mediante la realización de trabajos de programación. Los enunciados de estos trabajos siguen un esquema de presentación gradual basado en la taxonomía de objetivos cognitivos [15]: en primer lugar se presenta un programa correcto que los estudiantes deben compilar y ejecutar. Seguidamente, se les pide la realización de modificaciones para que demuestren que comprenden el programa y adquieran por imitación una habilidad básica de construcción de programas. Finalmente, se solicita a los estudiantes la creación de programas que deben construir por su cuenta.

En este artículo se describe la experiencia llevada a cabo al incorporar tecnologías XML y aplicaciones gráficas en la asignatura. El artículo es una continuación de [19] en el que se planteaba un esquema de prácticas para la asignatura *Programación Lógica y Funcional*. Con la entrada en vigor de un nuevo plan de estudios en el año 2002 pasó a denominarse *Programación Declarativa*. Aunque el enfoque ha sido similar, se ha decidido sustituir el lenguaje *Prolog* por el lenguaje *Curry*. El motivo de este cambio es que el lenguaje *Curry* ofrece unas características muy similares a las de *Haskell* por lo que el cambio conceptual es mínimo y puede verse la programación lógica como una evolución de la programación funcional que incluye indeterminismo y variables lógicas.

El presente artículo incluye es esquema de ejercicios que se ofrece a los estudiantes. Inicialmente, se presenta el lenguaje XML y algunos vocabularios XML habituales como SVG. A continuación se presenta el primer ejercicio básico, que consiste en representaciones gráficas de funciones y permite manipular el concepto de funciones de orden superior. En la sección 4 se estudia la estructura recursiva de *quadtree*. En la sección 5 se presentan los *octrees* y la sección 6 se

presenta la generación de mundos virtuales infinitos mediante evaluación perezosa. En la sección 7 se describe la creación de tipos de datos polimórficos. La sección 8 describe la utilización de características propias de la programación lógica (especialmente la utilización de variables lógicas e indeterminismo) y la siguiente sección se centra en la utilización de programación lógica con restricciones. Finalmente se resumen los principales trabajos relacionados y se detallan las principales conclusiones y líneas de investigación futuras.

Notación. A lo largo del artículo se utilizan fragmentos de código *Haskell* y *Curry*. Se supone que el lector tiene ligeros conocimientos de la sintaxis de ambos lenguajes. También se supone cierto conocimiento de vocabularios XML.

2. Vocabularios XML

El lenguaje XML [2] se ha convertido en el principal estándar para intercambio y representación de información en Internet. Uno de los primeros ejercicios planteados es la construcción de una librería de funciones que permita generar ficheros XML. Esta librería, que los estudiantes construyen, permitirá utilizar una base común en el resto de prácticas que facilita la generación de ficheros con vocabularios XML específicos.

La librería contiene las siguientes funciones básicas:

- *vacio e as* genera un elemento vacío e con atributos as
- *gen e es* genera un elemento *e* con subelementos *es*
- *genAs e as es* genera un elemento e con atributos *as* y subelementos *es*

Uno de los vocabularios XML específicos que se utilizará es el lenguaje SVG (Scalable Vector Graphics) [8] que se está convirtiendo en el principal estándar de representación de gráficos bidimensionales vectoriales en Internet.

Asimismo, X3D es vocabulario XML desarrollado por el consorcio Web3D [27] a partir del lenguaje VRML, que puede considerarse el principal lenguaje para representación de mundos virtuales en Internet.

3. Funciones de orden superior: Representaciones gráficas

El segundo trabajo práctico que se plantea consiste en la representación gráfica de funciones. A los estudiantes se les presenta la función plotF que permite almacenar en un fichero SVG la representación gráfica de una función.

```
plotF ::(Double→Double)→String→IO ()
plotF f fn = writeFile fn (plot f)

plot::(Double→Double)→String
plot f = gen “svg” (plotPs ps)
 where
  plotPs       = concat . map mkline
  mkline (x,x')= line (p x) (p x') c
  ps           = zip ls (tail ls)
  ls           = [0..sizeX]
  p x          = (x0+x, sizeY - f x)

line (x,y) (x',y') c =
  vacio "line"
    [(“x1”,show x) ,(“y1”,show y),
     (“x2”,show x'),(“y2”,show y')]

sizeX = 500; sizeY = 500; x0 = 10
```

Obsérvese que `plotF` realiza acciones de Entrada/Salida. Tradicionalmente, los cursos que enseñan el lenguaje *Haskell* tendían a retrasar la presentación del sistema de Entrada/Salida mediante mónadas. Creemos que una presentación gradual permite evitar malentendidos posteriores, ya que en caso contrario, muchos estudiantes consideraban extraña la posterior introducción de efectos laterales en un lenguaje que consideraban *puro*.

El código de la función `plot` sirve como ejemplo de utilización de los combinadores recursivos `zip, map, foldr`, etc. característicos del lenguaje *Haskell*.

En este trabajo práctico, los ejercicios que se proponen a los estudiantes utilizan el concepto de funciones de orden superior, clave del paradigma funcional. Así, por ejemplo, se solicita al estudiante que construya una función `plotMedia` que escriba en un fichero la representación de dos funciones y la función media de ambas. Por ejemplo, en la figura 1 se representa la media de las funciones `(\x -> 10 * sin x)` y `(\x -> 10 + sqrt x)`.

La solución del ejercicio en *Haskell* puede ser:

```
media f g = plotF (\x→(f x + g x)/2)
```

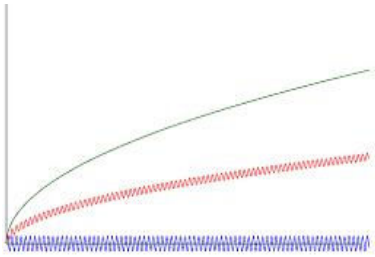


Figura 1. Representación de dos funciones y su media

4. Tipos de datos recursivos: *Quadtrees*

La siguiente unidad didáctica se centra en la presentación de técnicas de definición de tipos recursivos y su manipulación. Tradicionalmente, los trabajos prácticos en esta sección se realizaban con el tipo predefinido lista y con un tipo de datos definido por los alumnos para representar árboles binarios. Los estudiantes tienen serias dificultades para comprender los procesos recursivos y se sienten habitualmente poco motivados por este tipo de ejercicios [10,24]. Para intentar aumentar su motivación se trabajará con *quadtrees* [23] que son estructuras recursivas que permiten representar imágenes y tienen numerosas aplicaciones prácticas. En un *quadtree* las figuras se representan mediante un único color o la subdivisión de cuatro cuadrantes que son a su vez *quadtrees*. La generalización de los *quadtrees* al espacio tridimensional se denomina *octree* ya que cada subdivisión se realiza en ocho *octrees*. Estas estructuras son ampliamente utilizadas en el campo de la informática gráfica ya que permiten optimizar la representación interna de escenas y las bases de datos tridimensionales para sistemas de información geográfica.

En Haskell, un quadtree puede representarse mediante el siguiente tipo de datos:

```
data Color = RGB Int Int Int
data QT = B Color
        | D QT QT QT QT
```

Un ejemplo de quadtree sería

```
ejQT = D r g g (D r g g r)
  where r = B (RGB 255 0 0)
        g = B (RGB 0 255 0)
```

El ejemplo anterior define un *quadtree* formado por la subdivisión en cuatro cuadrantes, dos rojos y dos verdes dispuestos en diagonal. En la figura 2 puede observarse una representación del quadtree `ejQT`.

La visualización de *quadtrees* se realiza mediante una sencilla función que genera un fichero en formato SVG utilizando las funciones *vacio*, *gen,* y *genAs* implementadas por los estudiantes para la generación de documentos XML.

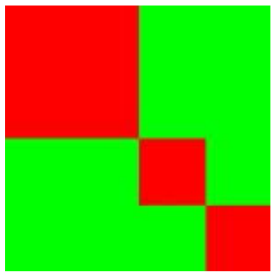


Figura 2. Ejemplo de *Quadtree*

```
type Punto = (Int,Int)
type Dim   = (Punto,Int)

verqt::QT→IO ()
verqt q = writeFile “qtree.svg”
      (gen “svg” (ver ((0,0),500) q))

ver::Dim→QT→String
ver ((x,y),d) (B c) =
 rect (x,y) (x+d+1,y+d+1) c

ver ((x,y),d) (D ul ur dl dr) =
  let d2 = d `div` 2
  in
   if d <= 0 then “”
   else ver ((x,y),d2)       ul ++
        ver ((x+d2,y),d2)    ur ++
        ver ((x,y+d2),d2)    dl ++
        ver ((x+d2,y+d2),d2) dr

rect::Punto→Punto→Color→String
rect (x,y) (x',y') (RGB r g b) =
 vacio "rect"
     [(“x”,show x),(“y”,show y),
      (“height”,show (abs (x - x'))),
      (“width”, show (abs (y - y'))),
      (“fill”, “rgb(”++show r++ “,”++
                       show g ++“,”++
                       show b ++“,”)]
```

5. *Octrees* y mundos virtuales

Un *octrree* es una generalización de un *quadtree* para representaciones tridimensionales: al subdividir cada cara de un cubo en cuatro partes se obtienen ocho

cubos. La representación de *octrees* en *Haskell* podría ser la siguiente:

```
data OT = Vacio
        | Cubo Color
        | Esfera Color
        | D OT OT OT OT OT OT OT OT
```

La representación anterior indica que un *octree* puede estar vacío, ser un cubo o una esfera con un determinado color, o una división en ocho *octrees*.

Un ejemplo de *octree* sería:

```
ejOT :: OT
ejOT = D v e e v r v v g
 where v = Vacio
       e = Esfera (RGB 0 0 255)
       r = Cubo   (RGB 255 0 0)
       g = Cubo   (RGB 0 255 0)
```

A los estudiantes se les presenta la función

```
wOT :: OT → FileName → IO()
```

que toma como argumentos un *octree,* un nombre de fichero y escribe en dicho fichero una representación en X3D del *octree*. En la figura 3 se presenta una pantalla capturada de la representación del oc*tree* ejOT en realidad virtual. Aunque en la figura se representa una versión impresa, el sistema genera un modelo virtual en el que los estudiantes pueden navegar.

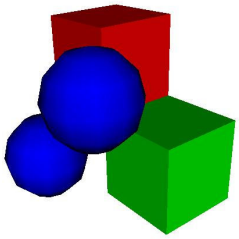


Figura 3. Ejemplo de *Octree*

6. Evaluación *Just-in time:* Mundos infinitos

La evaluación perezosa es una característica representativa del lenguaje Haskell que permite definir algoritmos que manipulan estructuras potencialmente infinitas. La evaluación perezosa puede también considerarse un tipo de evaluación *Just-in time* en la que el sistema no evalúa los argumentos de una función hasta que realmente necesita su valor. El programador puede definir y manipular *quadtrees* infinitos. Por ejemplo, es posible definir:

```
inf :: OT
inf = D inf v s v v v r inf
 where v = Vacio
       s = Esfera (RGB 0 0 255)
       r = Cubo   (RGB 255 0 0)
```

Obsérvese que el *octree* se define en función de sí mismo. Su visualización se presenta en la figura 4.

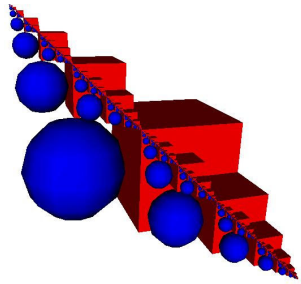


Figura 4. *Octree* infinito

Gracias a la evaluación perezosa, es posible definir funciones que manipulen mundos virtuales infinitos. Por ejemplo, la función `repite` toma como argumento un *octree* y genera un nuevo *octree* x repitiendo en cada cuadrante el *octree* x.

```
repite :: OT → OT
repite x = D x x x x x x x
```

Al aplicar la función repite al *octree* inf se obtendría el resultado de la figura 5.

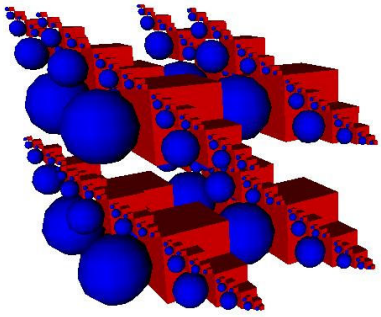


Figura 5. Repetición de un *Octree* infinito

7. Polimorfismo paramétrico: *Quadtrees* de alturas

El sistema de tipos del lenguaje *Haskell* admite la utilización de polimorfismo paramétrico. Las listas son el ejemplo tradicional de tipos de datos polimórfico. Aunque los *quadtrees* tradicionales contienen en cada cuadrante información del color, podría estudiarse una generalización que contuviese en cada cuadrante información de un tipo *a* que se pasa como parámetro. La nueva definición sería:

```
data QT a = B a
          | D (QT a) (QT a)
              (QT a) (QT a)
```

Los *quadtrees* con información de color serían valores de tipo `QT Color`.

La generalización anterior permite definir otros ejemplos de *quadtrees,* como los que contienen en cada cuadrante información de la altura (un valor de tipo `Float`). Estos *quadtrees* pueden utilizarse para la representación de alturas de terrenos. Por ejemplo el siguiente *quadtree*:

```
qta :: QT Float
qta = D x x x x
  where x = D b c a b
        a = B 0
        b = B 10
        c = B 20
```

se representa en la figura 6.

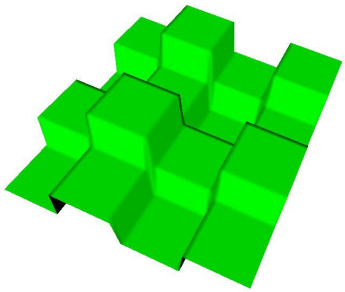


Figura 6. *Quadtree* de alturas

El lenguaje *Haskell* facilita y promueve la utilización de funciones genéricas. La función *foldr* es ejemplo de función genérica predefinida para el caso de las listas. Esta función puede también definirse para el tipo de datos *quadtree:*

```
foldQT::
  (b→b→b→b→b)→(a→b)→QT a→ b
foldQT f g (B x) = g x
foldQT f g (D a b c d) =
 f (foldQT f g a) (foldQT f g b)
   (foldQT f g c) (foldQT f g d)
```

Es posible definir múltiples funciones a partir de *foldQT*. Por ejemplo, para calcular la lista de valores de un *quadtree* puede definirse:

```
valores :: QT a → [a]
valores = foldQT
    (\a b c d → a ++ b ++ c ++ d)
    (\x → [x])
```

La profundidad de un *quadtree* puede definirse como:

```
profundidad :: QT a → Int
profundidad = foldQT
   (\a b c d → 1 + maximum [a,b,c,d])
   (\_ → 1)
```

La función *foldQT* pertenece al conjunto de funciones que recorren y transforman una estructura recursiva en un valor. Estas funciones se denominan también *catamorfismos* y son estudiadas en el campo de la programación genérica [1].

8. Indeterminismo: Generación de *quadtrees* en programación lógica

Una de las dificultades de la asignatura era la introducción de los paradigmas funcional y lógico en un breve espacio de tiempo. Hasta el curso pasado se empleaban dos lenguajes completamente diferentes: *Haskell* y *Prolog*. Sin embargo, desde el curso 2004-05 se ha optado por sustituir el lenguaje Prolog por el lenguaje Curry [11], un lenguaje híbrido lógico-funcional que tiene una sintaxis similar a Haskell pero añade variables lógicas e indeterminismo. En concreto, se ha utilizado el compilador Zinc [28] desarrollado en la Universidad de Oviedo, que añade clases de tipos al lenguaje Curry acercándose aún más al lenguaje Haskell.

Para la enseñanza del uso de variables lógicas e indeterminismo se realiza un primer ejercicio práctico utilizando listas. El ejercicio permite definir las permutaciones de una lista y ordenar una lista por el método de buscar una permutación que cumpla la condición de estar ordenada. Aunque el programa resultante es poco eficiente, el código resulta de una cierta elegancia:

```
inserta x [] = [x]
inserta x (y:ys) = x:y:ys
inserta x (y:ys) = y:inserta x ys

perm [] = []
```

```
perm (x:xs) = inserta x (perm xs)

ordenada [] = True
ordenada [x] = True
ordenada (x:y:ys) = x <= y
                 && ordenada (y:ys)

ordena xs | perm xs =:= ys
          & ordenada ys =:= True = ys
      where ys free
```

En el campo de los *quadtrees,* el siguiente predicado permite rellenar un *quadtree* con una lista de colores.

```
col (B _) (y:ys) = B y
col (B x) (y:ys) = col (B x) ys
col (D a b c d) xs =
   D (col a xs) (col b xs)
     (col c xs) (col d xs)
```

Obsérvese que al rellenar un quadtree con dos colores se obtienen varias respuestas.

```
? col (D (B 0) (B 0) (B 0) (B 0)) [1,2]
D (B 1) (B 1) (B 1) (B 1)) ;
D (B 1) (B 1) (B 1) (B 2)) ;
D (B 1) (B 1) (B 2) (B 1)) ;
. . .
```

Un problema clásico en el campo algorítmico es el de colorear un mapa de regiones con una serie de colores de forma que ninguna región adyacente tenga el mismo color. El problema puede plantearse para colorear *quadtrees*. En la figura 7 se presenta una posible solución al colorear un *quadtree* que representa un rombo.

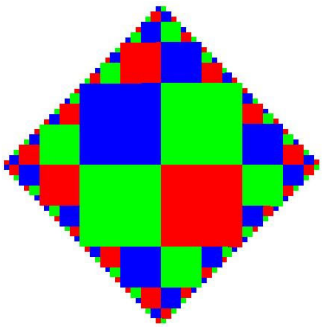


Figura 7. Solución del problema de coloreado

Una solución directa utilizando programación lógica consiste simplemente en generar todos los posibles *quadtrees* y comprobar la condición de no adyacencia. El siguiente predicado `noColor` realiza dicha comprobación.

```
noColor :: QT -> Success
noColor (B _) = success
noColor (D a b c d) =
  noColor a  & noColor b  &
  noColor c  & noColor d  &
  diff (right a) (left b) &
  diff (right c) (left d) &
  diff (down a)  (up c)   &
  diff (down b)  (up d)

data Tree a = L a | F (Tree a) (Tree a)

up (B x)       = L x
up (D a b _ _) = F x y
   where { x = up a; y = up b }

down (B x)        = L x
down (D _ _ c d ) = F x y
   where { x = down c; y = down d }

left (B x)       = L x
left (D a _ c _) = F x y
   where { x = left a; y = left c }

right (B x)       = L x
right (D _ b _ d) = F x y
  where { x = right b; y = right d }

diff (L x)  (L y)   = x =/= y
diff (L x) (F a b) = notInTree x a &
                     notInTree x b
diff (F a b) (L x) = notInTree x a &
                     notInTree x b
diff (F a b) (F c d) = diff a c
                     & diff b d

notInTree x (L y)   = x =/= y
notInTree x (F a b) = notInTree x a
                    & notInTree x b
```

El último ejercicio que se plantea es el desarrollo de un programa conversacional. Para ello se toma como base una implementación utilizando Eliza que contiene la funcionalidad mínima, y se solicita a los estudiantes que añadan más capacidad de conversación, bien aumentando el dominio de conocimiento sobre un determinado campo, o utilizando diferentes algoritmos conversacionales.

9. Trabajo relacionado

La exigua utilización de los lenguajes declarativos [25] ha llevado a varios miembros de esta comunidad a la búsqueda de aplicaciones atractivas que resalten las capacidades de este tipo de lenguajes. Cabe destacar el libro de texto escrito por P. Hudak [12] en el que se presenta una introducción a la programación

funcional incluyendo ejemplo relacionados con sistemas multimedia. En el libro se utilizan varias librerías específicas que permiten generar y visualizar los diversos ejercicios propuestos. La estrategia seguida en este artículo es similar en el objetivo, pero difiere en la forma de visualizar las construcciones gráficas. Se ha optado por utilizar vocabularios XML que se están convirtiendo en estándares en sus respectivos campos. Esta técnica supone varias ventajas: existencia de mayor número de herramientas de visualización, portabilidad entre plataformas e independencia de bibliotecas específicas. Además, los estudiantes emplean tecnologías XML estándar que pueden ser beneficiosas en otros campos de su trayectoria profesional.

Las representaciones declarativas de *quadtrees* fueron estudiadas en [3,9,6,26]. Recientemente, C. Okasaki [21] toma como punto de partida la representación en Haskell de un *quadtree* para definir una implementación eficiente de matrices cuadradas mediante tipos anidados. Su implementación mantiene la consistencia de la representación gracias al sistema de tipos de Haskell.

En el campo imperativo existen varios trabajos [7,14,16] que resaltan la utilización de *quadtrees* como buenos ejemplos de prácticas de programación, centrándose fundamentalmente en su aplicación para comprimir imágenes. En [5] se describen diversos algoritmos de coloreado de *quadtrees* y su aplicación práctica en la planificación de computaciones paralelas.

10. Conclusión y Líneas de trabajo

El presente artículo propone un esquema de trabajos prácticos que pretende potenciar la visualización gráfica de los resultados a la vez que se exploran las diversas características de los lenguajes declarativos. Aunque no se ha realizado un estudio sistemático de la reacción de los estudiantes ante este esquema, puede afirmarse que es altamente positiva, con porcentajes de abandono de la asignatura prácticamente nulos y con una tasa de aprobados cercana al 90%. Sin embargo, este tipo de afirmaciones debería contrastarse de una forma rigurosa comprobando, por ejemplo, que los estudiantes expuestos a este tipo de enseñanza realmente resuelven mejor otros problemas de programación.

La evolución más destacable del esquema presentado en [17] al que se ha presentado en este artículo es la utilización del compilador *Zinc* del lenguaje *Curry*, que añade clases de tipos a dicho lenguaje y ofrece una gran similitud con *Haskell*. De esta forma, el salto del paradigma funcional al paradigma lógico se realiza de una forma mucho más gradual que cuando se utilizaba el lenguaje *Prolog*. El principal inconveniente en la actualidad es que la implementación utilizada todavía no incluye satisfacción de restricciones, en cuyo caso podría incluirse una práctica sobre dicha técnica.

Una cuestión que ha surgido con el nuevo nombre de la asignatura se refiere a la propia definición del campo de la programación declarativa. Aunque el grueso de la asignatura se sigue manteniendo en programación funcional y programación lógica, se ha incluido una práctica intermedia utilizando XSLT, que puede considerarse un ejemplo de lenguaje funcional para la transformación de documentos XML [20]. Además, en otros contextos, como en el desarrollo de componentes de negocio, la utilización de XML para describir las relaciones y dependencias entre objetos, se denomina programación declarativa [22].

La generación de mundos virtuales ha supuesto un aliciente en la investigación del grupo IDEFIX [17,18]. Entre las futuras líneas de investigación, con el fin de aumentar la motivación de los estudiantes, se está estudiando la creación de comunidades virtuales donde los estudiantes puedan crear sus propios mundos y conversar con otros estudiantes mediante *chat*. En ese sentido, se está contemplando la inclusión de un último ejercicio práctico que enlace el desarrollo de robots conversacionales al sistema utilizando, por ejemplo, el sistema Jabber que también se basa en XML [13].

Referencias

[1] R. Backhouse, P. Jansson, J. Jeuring, L. Meertens, *Generic Programming – An Introduction,* Advanced Functional Programming, Lecture Notes in Computer Science, vol 1608, S. Swierstra, P. Henriques, J. N. Oliveira (Eds), 1999

[2] T. Bray, J. Paoli, C. M. Sperberg-McQueen, Extensible markup language (1.0), `http://www.w3.org/TR/REC-xml`, Oct 2000

[3] F. W. Burton, J. G. Kollias. *Functional programming with quadtrees.* IEEE Software, 6(1):90-97, Enero, 1989

[4] P. Codognet and D. Diaz. Compiling Constraints in CLP(FD). *Journal of Logic Programming*, Vol. 27, No. 3, June 1996

[5] D. Eppstein, M. W. Bern, B. Hutchings, *Algorithms for coloring quadtrees,* Algorithmica, 32(1), Ene. *2002*

[6] S. Edelman and E. Shapiro, *Quadtrees in concurrent prolog*, Proc. Intl. Conference on Parallel Processing, 1985, pp. 544-551

[7] J. B. Fenwick Jr., C. Norris, J. Wilkes, *Scientific Experimentation via the Matching Game,*SIGCSE Bulletin 34(1), 33th SIGCSE Technical Symposium on Computer Science Education, Marzo, 2002

[8] J. Ferraiolo, J. Fujisawa, D. Jackson, *Scalable Vector Graphics (SVG) 1.2* W3c Recommendation, Enero 2003

[9] J. D. Frens, D. S. Wise, Matrix inversion using quadtrees implemented in gofer. Technical Report 433, Computer Science Department, Indiana University, Mayo 1995

[10] J. Good, P. Brna, *Novice Difficulties with recursion: Do graphical Representations Hold the solution?*, European Conference on Artificial Intelligence in Education, Lisboa, Portugal, Oct.,1996

[11] M. Hanus (editor). *Curry: an integrated functional-logic language,* version 0.8, `http://www.informatik.uni-kiel.de/~mh/curry/report.html`

[12] P. Hudak, *The Haskell School of Expression: Learning Functional Programming through Multimedia,* Cambridge Univ. Press, 2000

[13] Jabber, página Web: `http://www.jabber.org`

[14] R. Jiménez-Peris, S. Khuri, M. Patiño-Martínez, *Adding breadth to CS1 and CS2 courses through visual and interactive programming projects,* ACM SIGCSE Bulletin 31(1), Marzo, 1999

[15] J. Kaasbøll, *Exploring didactic models for programming,* Norsk Informatikk-Konferanse, Høgskolen I Agder, 1998

[16] S. Khuri, H. Hsu, *Interactive Packages for learning image compression algorithms*, ACM SIGCSE Bulletin 32(3), Sept. 2000

[17] J.E. Labra, J.M. Morales, R. Turrado, *Plataforma de enseñanza de lenguajes de programación a través de Internet: Proyecto Idefix,* VIII Jornadas de Enseñanza Universitaria de Informática, JENUI-2002, Cáceres, Junio 2002

[18] J.E. Labra, J. M. Morales, A. M. Fernández, H. Sagastegui, *A Generic e-Learning Multiparadigm Programming Language System: IDEFIX Project*, ACM 34th SIGCSE Technical Symposium on Computer Science Education, Reno, Nevada, USA, Febrero 2003

[19] J. E. Labra, *Representaciones Gráficas y Mundos Virtuales Infinitos en las Prácticas de Programación Lógica y Funcional*, IX Jornadas de Enseñanza de la Informática, Cádiz, Julio 2003

[20] D. Novatchev, *The Functional Programming Language XSLT: A proof through examples*, TopXML, Nov. 2001

[21] Chris Okasaki, *From Fast exponentiation to Square Matrices: An adventure in Types*, ACM SIGPLAN Notices 34(9), Intl. Conference on Functional Programming, pp. 28-35, 1999

[22] E. Roman, R. P. Sriganesh, G. Brose, *Mastering Enterprise Javabeans,* Wiley, 3ª Edición, 2004

[23] H. Samet *The quadtree and related hierarchical data structures.* ACM Computing Surveys, 16(2): 187-260, Junio 1984.

[24] J.Segal, *Empirical studies of functional programming learners evaluating recursive functions,* Instructional Science 22, 385-411, 1995

[25] P. Wadler, *Why no one uses functional programming languages?,* ACM SIGPLAN Notices 33(8): 23-27, Agosto, 1998

[26] D. Wise, *Matrix algorithms using quadtrees.* Technical Report 357, Computer Science Department, Indiana University, Jun., 1992

[27] Web3D. Página Web: `http://www.web3d.org`

[28] Zinc-Project. Página Web: `http://zinc-project.sourceforge.net`

Análisis y diseño de algoritmos
Organización curricular para un primer contacto con la disciplina

Francisco Palomo Lozano, Inmaculada Medina Bulo
Dpto. de Lenguajes y Sistemas Informáticos. Universidad de Cádiz.
Escuela Superior de Ingeniería de Cádiz. C/ Chile, s/n. 11003 Cádiz.
e-mail: {francisco.palomo, inmaculada.medina}@uca.es

Resumen

En este trabajo presentamos nuestra experiencia con una nueva organización curricular para la materia de análisis y diseño de algoritmos en los primeros cursos de los estudios de Informática y realizamos una propuesta docente para una asignatura introductoria.

1. Introducción

En el curso 2002–03 comenzó a impartirse en la Universidad de Cádiz (UCA) un nuevo plan de estudios de la Ingeniería Técnica en Informática de Gestión [13] con diversos cambios en la organización curricular que afectan al estudio de las estructuras de datos y los algoritmos. La nueva organización cuatrimestral dispone cuatro asignaturas para estas materias.

En el primer año, tras un curso de introducción a la programación (IP) y otro de matemática discreta (MD), se cubren las estructuras de datos lineales (ED I), a la vez que se afianzan los conocimientos en metodología y tecnología de la programación (MTP).

El segundo año simultanea el estudio de las estructuras de datos no lineales (ED II) con una primera asignatura de análisis y diseño de algoritmos (ADA I), que posee una segunda parte (ADA II) dedicada al estudio de las principales técnicas de diseño de algoritmos y a su análisis detallado.

Este enfoque, que reconoce el carácter independiente de la Algoritmia, está en consonancia con las últimas recomendaciones de la ACM/IEEE-CS [1], en las que el área AL ha pasado a estar dedicada en exclusiva al estudio de los algoritmos y de la complejidad.

	1er Cuatrimestre		2º Cuatrimestre	
	Asig.	Ctos.	Asig.	Ctos.
1º	IP	3T+4,5P	MTP	3T+3P
	MD	3T+3P	ED I	3T+3P
2º	ED II	3T+3P		
	ADA I	3T+1,5P	ADA II	3T+1,5P

Cuadro 1: Ubicación de las asignaturas.

Algunos de los principios que inspiran esta organización son:

1. El estudio de los algoritmos puede separarse y superponerse al de las estructuras de datos, si bien los enfoques han de ser complementarios y coordinados.

   Hay estructuras de datos y algoritmos que constituyen «conceptos recurrentes» en ambos tipos de asignaturas, pero que se estudian bajo enfoques distintos.

2. Se deben conocer los rudimentos del análisis de algoritmos para apreciar la bondad de las técnicas de diseño.

   Si la eficiencia ha de ser una guía de diseño, es imprescindible conocer lo suficiente sobre análisis como para poder intuir a priori si un diseño va a ser o no adecuado.

De acuerdo con estos principios, debe existir una primera asignatura centrada en el análisis, pero que exponga buenos principios de diseño, que requiera algunos conocimientos de estructuras de datos, pero no todos, y que cimente el posterior estudio de las técnicas de diseño.

Es esta asignatura, ADA I, a la que prestaremos nuestra atención en este trabajo.

2. Asignatura de ADA I

Los objetivos fundamentales que pretende cubrir ADA I se resumen en aprender a:

1. Analizar formalmente la complejidad espacial y temporal de algoritmos básicos.

2. Programar en el laboratorio los algoritmos explicados en clase y contrastar los resultados experimentales obtenidos con la teoría expuesta.

Como se aprecia en el cuadro 1, la asignatura posee en el plan de estudios 3 créditos teóricos y 1,5 prácticos.[1]

Por ello, no tratamos de cubrir mucha materia superficialmente, sino poca en profundidad, dedicando a los aspectos fundamentales y a la resolución de problemas en clase todo el tiempo posible.

El cometido de la asignatura es, pues, sentar una sólida base para el estudio de ADA II, que puede entonces dedicarse a exponer las técnicas de diseño de algoritmos, como «divide y vencerás», algoritmos devoradores y programación dinámica, así como a analizar en detalle ejemplos notables de cada una de ellas.

Los prerrequisitos se encuentran repartidos entre diversas asignaturas. Por un lado, se esperan una serie de conocimientos matemáticos: que el alumno sepa resolver ecuaciones de recurrencia lineales, acotar un sumatorio, emplear el principio de inducción, etc.

Por otro, el alumno debe estar ya familiarizado con el diseño iterativo y recursivo elemental, la abstracción, la modularidad y el empleo de precondiciones, postcondiciones e invariantes. Las estructuras de datos lineales son también necesarias desde los primeros algoritmos.

Por último, cierto dominio del lenguaje C, empleado en primero, es imprescindible para comenzar las prácticas. Utilizaremos un subconjunto apropiado de C++ siguiendo el paradigma de la programación genérica, que nos permite separar con claridad las estructuras de datos de los algoritmos que las manipulan.

[1] Esto corresponde a 30 horas de teoría y 15 de prácticas, pero no es extraño que el calendario lectivo fuerce un número de horas algo inferior.

2.1. Contenidos teóricos

Los contenidos teóricos se organizan como sigue. Téngase en cuenta que el tiempo indicado junto a cada tema incluye el que se dedica a la resolución de problemas en clase, que suele ser de un par de horas por tema.

T1 Introducción y conceptos básicos (4 h)

Problemas, algoritmos y programas. Especificación. Corrección y eficiencia. Ejemplo: multiplicación rusa. Principio de invariancia. Definición informal de orden.

T2 Órdenes asintóticos (4 h)

Órdenes asintóticos. Operaciones.

T3 Complejidad algorítmica (8 h)

Tiempo y espacio. Enfoques en el análisis de algoritmos. Peor caso, mejor caso y promedio. Análisis de las estructuras de control. Ejemplo: algoritmos elementales.

T4 Algoritmos clásicos y su análisis (8 h)

Búsqueda. Métodos directos de ordenación. Ordenación por montículo. Componentes conexas.

T5 Complejidad de los problemas (6 h)

Problemas tratables e intratables. Problemas y árboles de decisión. Cotas inferiores de complejidad. Reducibilidad. Clases de complejidad.

El tema de introducción sirve de enlace con las asignaturas de primero donde el alumno ha estudiado ya superficialmente las nociones de corrección y eficiencia. También permite fijar la notación a emplear y presenta un primer ejemplo con el que se ilustran sistemáticamente todos los conceptos expuestos.

Se elige la multiplicación rusa como primer ejemplo porque sorprende a los alumnos de varias formas: refuta la muy extendida idea de que no hay otra forma de multiplicar que la que aprendimos de pequeños, es uno de los algoritmos documentados más antiguos [2] que se conocen y supone la implementación más simple que se puede realizar de la multiplicación en una computadora digital binaria.

El segundo tema expone con rigor los órdenes O, Ω y Θ y sus operaciones. A diferencia de lo que se hace en muchos textos, se definen los órdenes como conjuntos de funciones, como se predica en [3] y se practica en [4, 12].

Se insiste en las propiedades formales de los órdenes, ya que nos permitirán luego razonar con ellos mecánicamente, con seguridad, en vez de tener que recurrir continuamente a la intuición, como desgraciadamente ocurre en gran número de textos de esta disciplina.

El tercer tema comienza con las nociones de espacio y tiempo algorítmicos, abstractos por naturaleza, y su relación con sus homónimas reales. Tras justificar el análisis por el método de la operación crítica, se expone la necesidad de clasificar los ejemplares según su tamaño y los casos de análisis que surgen al hacerlo.

Por su dificultad intrínseca, prestamos especial atención al análisis en el caso promedio. Los ejemplos que se utilizan son sencillos (cálculo del mínimo, inserción en orden y comprobación de ordenación) y sirven de base a los algoritmos del siguiente tema.

El cuarto tema cubre la búsqueda secuencial y los métodos directos de ordenación (intercambio directo y selección e inserción directas), e incluye el análisis en el promedio de la búsqueda y de la ordenación por inserción.[2]

Nótese que hasta aquí sólo se han necesitado las estructuras lineales de ED I. El tema prosigue con algoritmos más elaborados que requieren algunas estructuras no lineales ya impartidas en ED II a estas alturas del curso.

El primero es un método de ordenación no directo: el algoritmo de Williams u ordenación por montículo. Se analizan con todo rigor dos algoritmos de construcción de montículos, el clásico de Williams y el lineal de Floyd. Se concluye con un algoritmo de cálculo de componentes conexas mediante estructuras de partición, durante cuyo análisis se incide en el impacto que la elección de la estructura de datos subyacente tiene sobre el algoritmo.

[2] La búsqueda binaria, la ordenación por fusión y la de Hoare se reservan a ADA II, ya que forman el canon de las distintas subespecies de algoritmos de «divide y vencerás» [4] y su análisis riguroso se aborda mejor con las técnicas que allí se explican.

El último tema es una introducción estándar a la teoría de la complejidad de los problemas, donde se intenta que el alumno comprenda la diferencia entre la complejidad de un algoritmo y la de un problema, y la existencia de una dificultad intrínseca a algunos problemas, insalvable mediante un aumento lineal de la potencia de cálculo disponible.

2.2. Contenidos prácticos

Las prácticas se organizan en varias sesiones de una hora de duración siguiendo un modelo de laboratorio cerrado en el que se proporciona al alumno un guión que contiene tanto las explicaciones como los ejercicios a desarrollar.

P1 Programación genérica con C++ (5 h)

C++ como un C mejorado. Programación genérica en C++. Biblioteca STL.

P2 Generación de ejemplares de prueba (2 h)

Pruebas de caja negra. Números pseudoaleatorios. Permutaciones pseudoaleatorias.

P3 Medida del tiempo de ejecución (4 h)

Tipos de medida. Formas de medir el tiempo. Factores que influyen en la medida. Elección de los ejemplares de prueba. Ejemplo: números de Fibonacci.

P4 Comparación entre algoritmos de ordenación (4 h)

Métodos directos de ordenación por comparación. Algoritmos de ordenación de la biblioteca del lenguaje.

El enfoque seguido en las prácticas, fundamentado en una combinación de programación genérica con C++/STL, empleo de herramientas de software libre y técnicas experimentales de análisis, ha sido objeto de un artículo específico [9] al que remitimos al lector interesado.

3. Materiales didácticos

Aparte de las habituales transparencias y boletines de problemas, hemos introducido algunos materiales didácticos que estimamos merecen algunos comentarios.

3.1. Curiosidades

Intentamos motivar al alumno proporcionándole toda suerte de referencias históricas y curiosidades, animándole a buscar información adicional en Internet.

Por ejemplo, al estudiar la multiplicación rusa se les habla del papiro de Ahmes [2]. Al hablar de la sucesión de Fibonacci, del canon de belleza griego, de Leonardo da Vinci y su Hombre de Vitrubio, y de la constante aparición del número áureo en la naturaleza.

3.2. Breves repasos

No entraremos en las causas, diversas y complejas, por las que un alumno llega a un segundo curso universitario de las características que nos ocupan sin la formación requerida. Simplemente constatamos la existencia del problema, ante el que tenemos tres opciones:

1. Rechazar el problema. Al fin y al cabo, no es nuestra culpa. Sigamos con el plan docente preestablecido y dejemos bien claro que éste no es nuestro problema.

2. Asumir el problema como nuestro. Al fin y al cabo, puede que el alumno no tenga la culpa. Expliquemos en clase todo aquello que no sepa aun a costa de nuestro plan docente.

3. Asumir el problema solidariamente. Proporcionemos los medios necesarios para que el alumno con su esfuerzo personal y algo de nuestra ayuda supla sus carencias, pero sin sacrificar el plan docente.

Es en este tercer estadio en el que nos encontramos actualmente. Un elemento indispensable para llevarlo a cabo es el «breve repaso».

Un breve repaso es un documento corto,[3] autocontenido, que un alumno puede estudiar holgadamente en una tarde por sí mismo y que ha de servirle para eliminar una laguna muy concreta de su formación sobre un prerrequisito de la asignatura.

Actualmente se han elaborado y puesto a disposición de los alumnos los siguientes:

1. Verificación de algoritmos: corrección, lógica de Floyd-Hoare, verificación a posteriori de algoritmos iterativos.

2. Recursividad lineal: recursividad lineal y final, relación con la iteración, generalización, desplegado y plegado.

3. Acotación de un sumatorio por una integral: técnica de acotación integral, interpretación gráfica, aplicación al cálculo de órdenes asintóticos.

4. Ecuaciones de recurrencia: relación con el cálculo de sumatorios, clasificación, resolución por el método de la ecuación característica, etc.

Los breves repasos se suelen acompañar de una abundante bibliografía de consulta que, sin ser de obligada lectura para su comprensión, ha de servir al alumno interesado para ampliar sus conocimientos.

3.3. Vídeos

University Video Communications (UVC) es una empresa que centra su actividad fundamentalmente en el ámbito académico y que distribuye conferencias de personas de reconocido prestigio internacional en la universidad o en la industria.

Las conferencias son normalmente divulgativas y de una gran calidad técnica, pero se encuentran normalmente en inglés, sin subtitular, lo que suele suponer un inconveniente para el alumno medio, con escasos conocimientos de lenguas extranjeras.

El último tema de teoría, dedicado a la complejidad de los problemas, se ha impartido utilizando un vídeo con una conferencia del Prof. Richard Karp [7], uno de los padres de la teoría de la complejidad, que lleva por título «NP-complete problems». En ella se cubre magistralmente una parte importante de los conocimientos generales que se espera posea un alumno sobre complejidad al terminar la asignatura.

Para romper la barrera del idioma los profesores transcribieron la conferencia en su idioma original, entregando la transcripción a los

[3] Idealmente, de menos de 10 páginas.

alumnos,[4] previamente a la emisión de la conferencia.

3.4. Guiones de prácticas y software libre

Las prácticas de laboratorio son cerradas, por lo que el guión de prácticas juega un papel fundamental para el alumno. No sólo contiene los ejercicios a realizar en cada práctica, sino que, a modo de apuntes, contiene explicaciones sobre el trabajo a desarrollar o sobre técnicas de programación o funciones de biblioteca que pueden ser útiles en su realización.

Pensamos que esto no exime en ningún modo al profesor de dedicar algunos minutos al comienzo de la clase a orientar a los alumnos sobre el desarrollo de sus tareas y las dificultades que puedan encontrarse.

Por otro lado, intentamos promover en nuestros alumnos el empleo de software libre, proporcionando información sobre las herramientas disponibles y un entorno adecuado para la realización de las prácticas en el laboratorio. Utilizamos un sistema operativo GNU/Linux, la última versión estable del compilador GNU C++ y algunas herramientas más, principalmente:

- GNU Plot, que permite automatizar determinadas tareas de análisis y representación gráfica, ya que se programa de manera sencilla y permite obtener gráficas de calidad en una gran variedad de formatos.
- GNU Make, que está especialmente diseñada para ayudar a rehacer trabajos monótonos que deban repetirse cada vez que ciertos ficheros sean modificados. Ésta es justo la situación que nos encontramos en muchas ocasiones en el laboratorio, cuando hay que modificar el código de un experimento, recompilarlo, obtener los nuevos tiempos y redibujar las gráficas.

Para ahorrar trabajo al alumno, suministramos unos esqueletos tipo de los ficheros de entrada para las herramientas anteriores, permitiendo así que se concentre en el código del experimento en sí, y no en su organización.

[4] Deliberadamente sin acompañarla de una traducción a nuestra lengua, ya que creemos que es importante que empleen recursos en lengua inglesa.

4. Campus virtual

Ya que nuestra universidad ha adoptado institucionalmente la plataforma comercial de enseñanza virtual WebCT (`www.webct.com`), los autores decidieron incorporar su asignatura al programa de tutorías electrónicas y docencia en la red promovido por el Vicerrectorado de Ordenación Académica de la UCA.

Sin ser la plataforma ideal para el docente, que ha de pelear con aspectos bastante idiosincrásicos de la misma, hemos de reconocer que es cómoda de utilizar para el alumno y que, en general, ha sido muy bien aceptada.

Los datos de los alumnos se cargan de manera centralizada a través de los disponibles en secretaría liberándonos así de una de las tareas de administración más pesadas. Cada alumno recibe automáticamente un identificador de usuario y una contraseña común a todos los cursos en los que está matriculado.

Seleccionamos y pusimos a disposición de los alumnos varias herramientas de la plataforma:

- Agenda (recordatorios de fechas, cambios de última hora, etc.)
- Herramientas de organización de contenidos (plan docente, apuntes, transparencias, problemas, guiones, listados, etc.)
- Herramientas de comunicación (foros de debate, correo interno y charla en línea)

Sin duda, las herramientas más utilizadas han sido las de comunicación. Curiosamente, hemos constatado que no se emplea en absoluto la charla en línea. Puede que los alumnos deseen charlar en línea a través de sus propios canales, lejos de la mirada del profesor.

En la actualidad mantenemos varios foros: uno principal para dudas generales acerca de la organización docente, dos de contenidos para dudas relacionadas con la materia teórica y la práctica, respectivamente, otro de anuncios importantes de obligada lectura, otro privado para los profesores de la asignatura y, por último, uno de sugerencias.

Inicialmente, manteníamos un único foro para los contenidos teórico-prácticos, pero algunos alumnos expresaron su desconocimiento

acerca de cuestiones aclaradas a otros compañeros en él por contener éste un gran número de mensajes y estimar que carecían del tiempo necesario para buscar lo que les interesaba. Se optó, pues, por dividir el foro de contenidos en dos para facilitar su empleo.

Estimamos que el foro es, probablemente, la herramienta de comunicación más útil ya que permite que los alumnos pierdan el miedo a contestar a las dudas de sus propios compañeros y facilita que el profesor pueda detectar y responder de una vez dudas generales o aclarar conceptos mal aprendidos.

Los profesores de la asignatura mantienen, no obstante, dos horas semanales de tutorías presenciales aparte de atender las tutorías electrónicas.

No obstante, hemos constatado una disminución muy preocupante del habitualmente escaso número de alumnos que asisten a las tutorías presenciales. Parece ser que los alumnos se sienten realmente cómodos con el sistema electrónico de tutorías, lo que nos alegra y nos preocupa a la vez, ya que estimamos que hay dudas y cuestiones que es mejor resolver «cara a cara» con el profesor.

Una herramienta disponible, pero que no hemos utilizado, es la que permite realizar pruebas tipo «test». Esto se debe principalmente a la dificultad que presenta una asignatura como la que nos ocupa, basada fundamentalmente en la resolución de problemas, para establecer pruebas de este tipo que sean de utilidad.

Actualmente, el correo electrónico interno se emplea exclusivamente para la resolución de dudas en privado. Desde que se implantó este sistema, en cada curso se ha contabilizado una media de 192 comunicaciones por esta vía, un número algo inferior a los 254 mensajes gestionados en los distintos foros. Como puede inferirse fácilmente, la gestión de tal cantidad de información (una media de 30 comunicaciones semanales) supone un esfuerzo añadido para los profesores implicados.

5. Método de evaluación

Por razones relacionadas con la implantación gradual del nuevo plan de estudios y la resistencia natural que presentan los alumnos de planes antiguos a la adaptación a un nuevo plan, al menos durante sus primeros años, el primer curso académico del nuevo plan contamos con un número inusitadamente bajo de alumnos en clase (inferior a los 75) para los estándares de nuestra titulación.[5]

Esto nos permitió formar tres grupos prácticos, en torno a los 25 alumnos cada uno, y llevar a cabo una evaluación continua de la parte práctica de la asignatura, no sin un considerable esfuerzo por parte de los profesores implicados.

Para cada práctica, el profesor corregía los errores de programación, concepto e, incluso, estilo. El alumno acudía a una entrevista personal donde se le explican los diversos problemas encontrados y se abre un nuevo plazo para su modificación, entrega y corrección final. Desgraciadamente, el aumento del número de alumnos ha tornado este sistema inviable, por lo que hemos vuelto a un sistema de evaluación tradicional con la obligatoriedad de la entrega final de algunas prácticas seleccionadas.

En cualquier caso, hemos observado que los alumnos apenas leen los guiones de prácticas y que, más bien, tienden a comenzar directamente con la resolución de los ejercicios, en ocasiones sin comprender bien qué es lo que están haciendo o qué se les pide realmente. Encontramos difícil combatir esta tendencia, probablemente, acumulada a lo largo de años.

Para la evaluación de la parte teórica se ha realizado un examen final. El examen final consta principalmente de problemas en los que el alumno tiene que aplicar reiteradamente los conocimientos adquiridos en todo el temario. Esto, que es posible gracias al escaso número de temas, es una forma de evitar que se dejen de estudiar algunos de ellos en la esperanza estadística de que no aparezcan en el examen.

Los alumnos con notas superiores a 4 en el examen teórico, aprobado en prácticas y participación activa en las clases y en el campus virtual obtienen el aprobado global de la asignatura.

[5] La antigua asignatura «Metodología y Tecnología de la Programación II», donde se impartía materia relacionada, llegó a tener 280 alumnos.

6. Evaluación de la propuesta

Los autores participan en un proyecto de formación continua integrado en el Plan Andaluz de Formación del Profesorado Universitario de la Unidad para la Calidad de las Universidades Andaluzas (UCUA).

Este proyecto de formación, de duración anual, se estructuró en su primera edición sobre tres ejes fundamentales:

1. Alumnos: motivación, relaciones alumno-profesor, técnicas de diálogo, y obtención de información sobre su perfil.

2. Actividad docente: exposición en el aula, materiales de estudio, recursos docentes y métodos de evaluación.

3. Planificación docente: coordinación entre asignaturas, revisión y definición del ámbito de conocimientos de cada una.

Además, se consideró el estudio de tres categorías más, de naturaleza transversal, ya que afectan a las señaladas anteriormente: el papel de las TIC en la educación, la adaptación e impacto de los créditos ECTS y los principios éticos que deben regir la actividad del docente, del alumno y del futuro profesional.

Como parte del proyecto, se desarrolló una encuesta exhaustiva (alrededor del centenar de preguntas) sobre diferentes aspectos relacionados con las asignaturas. Pasamos a exponer algunos resultados obtenidos sobre ADA I.

El 70 % de los alumnos encuestados tiene dos o más asignaturas pendientes de primer curso. La tasa de abandono es muy alta: la mitad de los alumnos abandonan la asignatura, indicando como principal causa su dificultad.

En este sentido, el 95 % de los alumnos piensa que la asignatura es difícil o muy difícil, aunque el 65 % piensa aprobarla dentro del curso académico. Sin embargo, y sorprendentemente, más de la mitad afirma que se matricularía si la asignatura no fuera obligatoria. Esto puede deberse al hecho de que el 81 % la consideran interesante. Los alumnos que no esperan aprobar la asignatura reconocen como causas principales su falta de base o el estar cursando demasiadas asignaturas simultáneamente.

Por último, queremos destacar que el 89 % de los encuestados opina que la implantación del campus virtual ha sido muy positiva para el aprendizaje y seguimiento de la asignatura.

No obstante, hemos de advertir que estos datos han de ser interpretados con cierta cautela, ya que la participación en el proceso fue voluntaria y, por consiguiente, no muy alta, estando en torno al 35 % de los matriculados.

7. Conclusiones

Se ha descrito y evaluado una propuesta docente para una asignatura introductoria de análisis y diseño de algoritmos, una disciplina que forma parte de la materia troncal en los planes de estudio de Informática y considerada fundamental en las recomendaciones curriculares internacionales.

Se han introducido algunos recursos didácticos como los breves repasos, el empleo de vídeos con conferencias en versión original, las prácticas estructuradas en guiones con empleo exclusivo de software libre y el campus virtual.

La evaluación de la propuesta, las opiniones de los alumnos y nuestra experiencia durante el curso nos han hecho reflexionar sobre algunas de las posibles mejoras a introducir en sucesivos cursos, que pasamos a describir.

Estimamos que el análisis de las discusiones que se producen en los foros es una fuente magnífica para recabar información acerca de los verdaderos puntos débiles de la formación de nuestros estudiantes. Esta información nos está permitiendo elaborar:

- La lista de preguntas más frecuentes de la asignatura, tanto para la parte teórica como para la práctica.

- Breves repasos de prerrequisitos específicos en los que detectamos lagunas en la formación del alumno.

Creemos que distribuir algunos de los exámenes resueltos por el profesor (pero no todos) haciendo énfasis en el análisis de su enunciado y en el estilo de presentación de la solución para cada problema tipo, ayuda a mejorar los resultados obtenidos.

Estamos convencidos de que la experiencia con el empleo de los vídeos de UVC ha sido tremendamente enriquecedora. Para próximos cursos, tenemos pensado incluir en el primer tema una conferencia del Prof. E. W. Dijkstra [5] titulada «Reasoning about programs» y disponible en la misma colección. Sería interesante también completar las transcripciones de las conferencias con glosarios, aclaraciones, preguntas de comprensión y otros recursos tendentes a aumentar su valor didáctico.

En esta línea, valoramos también la posibilidad de utilizar algunos de los materiales audiovisuales disponibles a través del CD-ROM que acompaña a [10], una obra no muy conocida en nuestro país, pero de indudables cualidades pedagógicas.

Consideramos también interesante recomendar a nuestros alumnos otras obras que no gozan de gran difusión aún, pero que, a buen seguro, pronto lo harán.

A. Levitin propone en [8] una nueva taxonomía para el estudio de las técnicas de diseño de algoritmos, que adereza con numerosos juegos de ingenio y un estilo expositivo claro, riguroso y muy cuidado.

En [6], los autores proponen problemas ligados a aplicaciones reales que no suelen encontrarse en otros textos, así como numerosas notas y ejercicios destinados a estimular la curiosidad del lector y despertar en éste el deseo de buscar recursos en Internet como los que los autores han recopilado para la obra.

Por último, pensamos que [11] puede servir para aumentar el interés de nuestros alumnos por la programación en general. Resulta atractiva la posibilidad de asignar trabajos de programación sobre problemas que ya han sido cuidadosamente preparados y clasificados según su dificultad y popularidad. Por supuesto, muchos de estos problemas requieren estructuras de datos y algoritmos ingeniosos. La entrega, seguimiento y corrección de los trabajos puede realizarse a través de Internet con el mismo sistema que se utiliza en las competiciones internacionales de programación, que emplea «jueces robot». De hecho, esta obra sirve de manual de entrenamiento para la participación en dichos eventos.

Referencias

[1] ACM/IEEE. Joint task force on computing curricula. *Computing curricula 2001.* ACM Press y IEEE Computer Society Press (2001)

[2] Ahmes. *Reglas para obtener el conocimiento de todo lo oculto.* Papiro «Rhind», Museo Británico (1700 A.C.)

[3] Brassard, G. *Crusade for a better notation.* ACM Sigact News, 17(1) (1985)

[4] Brassard, G. y Bratley, T. *Fundamentos de Algoritmia.* Prentice-Hall (1997)

[5] Dijkstra, E. W. *Reasoning about programs.* Stanford University Distinguished Lecture Series. University Video Communications (1991)

[6] Johnsonbaugh, R. y Schaefer, M. *Algorithms.* Pearson (2004)

[7] Karp, R. *NP-complete problems.* Stanford University Distinguished Lecture Series. University Video Communications (1993)

[8] Levitin, A. *Introduction to the design & analysis of algorithms.* Addison-Wesley (2003)

[9] Palomo Lozano, F. y Medina Bulo, I. *Análisis híbrido: una propuesta práctica.* Actas de las IX Jornadas de Enseñanza Universitaria de la Informática. Thomson (2003)

[10] Skiena, S. S. *The algorithm design manual.* Telos (1998)

[11] Skiena, S. S. y Revilla, M. A. *Programming challenges. The programming contest training manual.* Springer (2003)

[12] Peña Marí, R. *Diseño de programas: formalismo y abstracción.* Prentice-Hall, 3ª ed. (2004)

[13] Universidad de Cádiz. *Plan de estudios conducentes al título de Ingeniero Técnico en Informática de Gestión.* BOE Nº 171 de 18 de julio de 2002 (2002)

Una herramienta de apoyo en la enseñanza de Teoría de Autómatas y Lenguajes Formales [*]

Román Sosa y Francisco de Sande
Dep. de Estadística, I.O. y Computación
ETS Ingeniería Informática
Universidad de La Laguna
38271 La Laguna
{alu1970,sande}@etsii.ull.es

Resumen

La herramienta *ToTalf* (Tutorial Online para Teoría de Autómatas y Lenguajes Formales) ha sido diseñada con la finalidad de apoyar la realización de prácticas en la asignatura de *Teoría de Autómatas y Lenguajes Formales*. Las prácticas de laboratorio constituyen una parte esencial de esta asignatura puesto que en los planes de estudios de la Universidad de La Laguna la mitad de sus créditos son de tipo práctico.

ToTalf permite al alumnado experimentar con los conceptos estudiados en las clases de teoría, a través de la ejecución de distintos algoritmos que se aplican sobre los diferentes tipos de modelos de computación que se estudian en las clases teóricas. La herramienta ha sido desarrollada en Java, y está disponible tanto en forma de aplicación como de applet, de modo que los alumnos pueden instalarla localmente en sus ordenadores o bien utilizarla a través de cualquier navegador. Dispone de un sistema de ayuda que facilita el uso de la interfaz de usuario y para cada uno de los diferentes modelos de computación contemplados se ofrecen al usuario ejemplos predefinidos que se pueden cargar directamente y que también ilustran el modo de operación de la herramienta.

[*]El proyecto ToTalf ha sido financiado por el Vicerrectorado de Calidad Docente y Nuevos Estudios de la Universidad de La Laguna en la II Convocatoria de Proyectos de Innovación Docente

Uno de los objetivos esenciales en el diseño de *ToTalf* ha sido que fuera una aplicación abierta de modo que en el futuro sea fácil incorporar nuevas funcionalidades, ejemplos, así como modificar o completar el sistema de ayuda.

1. Introducción

El proyecto *ToTalf* [6] es un sistema de ayuda para el aprendizaje de conceptos básicos en la materia de autómatas y lenguajes formales. Concretamente, ha sido desarrollado para los alumnos que cursan la asignatura de *Teoría de Autómatas y Lenguajes Formales* (TALF) [3] en la Universidad de La Laguna, aunque algunas de sus capacidades son también aprovechables en la asignatura de *Compiladores*. El desarrollo del proyecto tuvo una duración aproximada de seis meses y el primer prototipo fue implantado al final del curso académico 2003-2004.

ToTalf aborda el estudio de la asignatura mediante un enfoque práctico, dividiendo el temario de la asignatura en cuatro módulos principales.

1. Autómatas finitos y expresiones regulares
2. Gramáticas independientes del contexto
3. Autómatas a pila
4. Máquinas de Turing

En cada uno de estos módulos, el usuario puede introducir un ejemplo del concepto correspondiente, y aplicarle los distintos algoritmos

que se encuentran implementados, y que se estudian en la asignatura. De esta forma, el alumno puede aplicar directamente los conocimientos y algoritmos estudiados en las clases de teoría, en vez de tener que resolver los problemas usando lápiz y papel, lo que consume demasiado tiempo, es propenso a los errores, y sumamente tedioso.

En la actualidad es posible encontrar herramientas de características similares a las de *ToTalf*. Es el gran auge de Internet quien ha permitido la proliferación de programas de este tipo. Cada uno suele tener alguna propiedad interesante, pero la carencia de otras los hacen ser herramientas deficientes para nuestros propósitos. En general, la mayoría se centra en un único aspecto: la simulación de autómatas finitos. En el caso de las aplicaciones desarrolladas fuera de España, el hecho de que todos los menús así como el sistema de ayuda esté en inglés, desanima a algunos alumnos a utilizarlas. Revisemos a continuación las aplicaciones más relevantes:

- JFLAP [9] es un paquete de herramientas gráficas diseñado como asistente en la enseñanza de conceptos básicos en TALF. Se trata posiblemente de la herramienta más sofisticada y evolucionada de este tipo. Con respecto a *ToTalf*, la característica más notable es que dispone de una interfaz gráfica que permite al usuario introducir autómatas y visualizar la salida de los diferentes algoritmos.
- ABCEZ (Applet Based Computability Enlightenment Zone) [10] es otra de las herramientas basadas en applets con mayor impacto visual. Igual que JFLAP también posee una cuidada interfaz gráfica, pero en este caso centra su atención exclusivamente en la simulación de autómatas finitos. Una de las características más destacables de este programa es su sistema de ayudas.
- El applet diseñado por J. Calera [1] para la asignatura *Lenguajes, Gramáticas y Autómatas* en la Universidad de Alicante es uno de los pocos ejemplos disponibles de herramientas similares a *ToTalf* en el contexto español. Se centra también en algoritmos orientados a autómatas finitos, y al igual que en nuestro trabajo, en lugar de invertir esfuerzo en el diseño de una interfaz gráfica avanzada, se ha optado por implementar un amplio conjunto de algoritmos.

En las referencias pueden encontrarse otros trabajos [2, 5, 4], sobre simulación de autómatas finitos, y que no aportan grandes diferencias respecto a los ya comentados en los puntos anteriores.

2. Contexto y Objetivos

En la *Escuela Técnica Superior de Ingeniería Informática (ETSII)* de la Universidad de La Laguna se imparten los estudios correspondientes a las titulaciones de Ingeniero Técnico en Informática de Sistemas (ITIS) y de Gestión (ITIG), así como los de Ingeniero en Informática.

La asignatura de TALF figura como troncal en ITIS y como obligatoria en ITIG, y tiene asignados un total de nueve créditos, repartidos en partes iguales entre teoría y prácticas. La asignatura se imparte a lo largo del primer cuatrimestre del segundo curso.

TALF es una asignatura con unos contenidos eminentemente abstractos que se han desarrollado históricamente a partir de los problemas concretos a los que se ha ido enfrentando la comunidad científica en su intento de utilizar lenguajes de alto nivel en la programación de ordenadores, de traducir automáticamente o de matematizar el lenguaje humano para procesarlo automáticamente.

La repercusión más directa de los contenidos de TALF, quizás se produce en las asignaturas del plan de estudios relacionadas con compiladores: *Compiladores* en las ingenierías técnicas, así como *Procesadores de Lenguajes* en la Ingeniería en Informática. Aparte de esta repercusión directa, los contenidos de TALF también son fundamentales en muchos otros campos y asignaturas estudiados en la carrera.

2.1. Objetivos generales de TALF

Con esta asignatura se pretende transmitir al alumno los conceptos fundamentales de la *Teoría de Autómatas y Lenguajes Formales*, presentándole una visión global de la misma y profundizando principalmente en los aspectos más aplicados como son las herramientas básicas que se utilizan en el diseño de analizadores léxicos y sintácticos a partir de expresiones regulares y gramáticas independientes del contexto, respectivamente. Los objetivos principales de la asignatura son:

- Presentar los distintos métodos formales de representación de lenguajes, estudiando métodos generativos (gramáticas) y métodos por aceptación (autómatas).
- Conocer las herramientas que se utilizan para describir, reconocer y caracterizar dichos lenguajes.
- Desarrollar la capacidad de abstracción y análisis teórico en relación con la teoría de lenguajes.

2.2. El programa de la asignatura

El temario de teoría impartido en la asignatura es el siguiente:

1. Introducción al lenguaje C++.
2. Conceptos básicos
3. Lenguajes regulares y autómatas finitos.
4. Gramáticas. Lenguajes independientes del contexto. Autómatas a pila.
5. Máquinas de Turing. Lenguajes recursivos y recursivamente enumerables.
6. Resolubilidad.

Sin desdeñar los contenidos teóricos, indispensables para el desarrollo de cualquier técnica, consideramos que los estudios de Informática son eminentemente prácticos y es por ello que en las asignaturas que impartimos, las prácticas con ordenador constituyen un aspecto fundamental de las mismas como complemento insustituible a las clases de teoría. Semanalmente los alumnos presentan en una sesión de corrección de prácticas una de las prácticas cuyo enunciado se les ha propuesto con antelación. Los contenidos de las diferentes prácticas pueden variar de un año a otro, pero como es natural, siempre están íntimamente relacionadas con los temas que se estudian en teoría.

Los objetivos docentes a alcanzar a través de las prácticas de la asignatura los podemos resumir en:

1. La consolidación a través de las experiencias de laboratorio de los conceptos adquiridos en las clases teóricas.
2. Introducir a los estudiantes en un lenguaje de programación orientado a objetos, que es nuevo para ellos (durante el primer curso de la carrera los alumnos desarrollan sus prácticas usando Pascal).
3. Mejorar el dominio de la programación adquirido en el curso anterior en las asignaturas de *Metodología y Tecnología de la Programación I y II*.

2.3. Objetivos del Proyecto ToTalf

El proyecto *ToTalf* surge como una necesidad del profesor para mejorar la enseñanza de la asignatura. Para este fin, *ToTalf* implementa la totalidad de los algoritmos que se programan en las prácticas (hay que tener en cuenta que las prácticas de la asignatura cambian en cada curso), y muchos otros algoritmos interesantes estudiados en las clases de teoría, incluyendo algunos correspondientes a la asignatura de *Compiladores*. Revisamos a continuación los objetivos que perseguía *ToTalf* en su planteamiento inicial.

Como ya se comentó en la introducción, trata de ser una herramienta para ayudar al alumno en la comprensión de la teoría y los conceptos relacionados con la asignatura. El poder experimentar con dichos conceptos mediante la aplicación de algoritmos debe ser una actividad que facilite esa comprensión. Gracias a *ToTalf*, el alumno puede comprobar directamente si ha realizado de manera correcta un ejercicio teórico. Además, en el caso de problemas de mayor tamaño, el alumno invierte su tiempo en el análisis de los resultados de los algoritmos, en vez de utilizarlo en realizar trazas sobre papel.

ToTalf se diseña como asistente en la realización de prácticas. Los alumnos pueden experimentar con los algoritmos antes de acometer

las prácticas que deben realizar. Pero fundamentalmente, es un marco de comprobación. En general, los alumnos dedican mucho tiempo a la implementación de las prácticas, pero poco a la comprobación de los resultados. Se suelen conformar con comprobar su código con los ejemplos que se le han suministrado, lo cual a veces resulta insuficiente. Con *ToTalf* se pretende que el alumno tenga de una manera sencilla una forma de examinar los resultados que devuelve su programa.

También la herramienta es útil en la realización de prácticas de cierta envergadura, en las que haya que calcular ciertos resultados intermedios, los cuales no son objetivo de estudio de la práctica.

Por último podemos considerarla una herramienta interesante para aquellos alumnos curiosos que gusten de experimentar por su cuenta.

3. ToTalf: la herramienta

3.1. Implementación

ToTalf está programado en Java. La elección de Java como lenguaje de programación responde a la gran ventaja de su ejecución multiplataforma, que permite su uso en multitud de entornos, con la ventaja de poder ejecutarse de manera sencilla y directa en un navegador. Concretamente, *ToTalf* está programado en Java 1.4, y se proporciona tanto en versión applet como en versión aplicación.

Para la ejecución del applet (es decir, la ejecución a través del navegador) es necesario un navegador con soporte para la máquina virtual Java de Sun [7]. Por lo tanto, se necesita disponer del paquete J2RE (entorno de ejecución de Java) o el paquete J2SDK (entorno de desarrollo de Java), disponibles ambos en la sección de descargas.

Sin embargo, para un uso cotidiano, se recomienda la versión aplicación, que no está afectada por las restricciones de seguridad impuestas a los applets. Con la versión aplicación podremos cargar y guardar archivos, y copiar y pegar entre aplicaciones y el applet, o viceversa. Para la ejecución de la aplicación, es necesario, al igual que antes, J2RE o J2SDK, y también descargar el fichero `jar` del proyecto.

Pasemos a estudiar los principales requisitos que se impusieron a la aplicación a desarrollar. El primero es el de disponer de un sistema de ayuda fácilmente modificable para que el usuario pueda consultar cómo operar con la herramienta. Para cumplirlo se optó por realizar la ayuda de *ToTalf* en formato *HTML*. Se trata de una solución estándar ampliamente utilizada, ya que los ficheros de ayuda se puede crear y modificar fácilmente, más aún teniendo en cuenta la multitud de herramientas disponibles en este campo.

Pretendíamos también que la herramienta admitiese con facilidad la incorporación de nuevos algoritmos. En este sentido se puso un especial esfuerzo en el diseño de los objetos que representan los diferentes modelos de computación para permitir la escalabilidad del programa. Cada objeto dispone de abundantes métodos para modificar y acceder a la estructura interna del mismo, lo que simplifica la implementación de nuevos algoritmos.

También fue un requisito que la aplicación admitiese la incorporación de nuevos ejemplos predefinidos para cada modelo de computación. A través de la redefinición de una tabla es posible añadir nuevos ejemplos predefinidos. Cada ejemplo se codifica del mismo modo que se hace en la interfaz de usuario, lo cual facilita la modificación de dicha tabla.

El último de los requisitos del proyecto fue que se desarrollara sobre Linux, en un entorno de desarrollo de código abierto, ya sea en modo texto o gráfico. A la vez, el entorno de desarrollo debía ser de fácil uso, y permitir de forma sencilla la modificación del código fuera de él. Se optó por la herramienta de desarrollo JBuilder8 Personal [8] puesto que se ajusta a estos parámetros y está disponible gratuitamente para usos no comerciales.

3.2. Utilizando ToTalf

El tutorial se compone de cuatro grandes módulos que abarcan los diferentes aspectos abordados en la asignatura: automatas finitos y expresiones regulares, gramáticas independientes

del contexto, autómatas a pila y máquinas de Turing.

En cada módulo el usuario puede introducir una descripción textual del concepto correspondiente. Dicha descripción es igual a la que se utiliza en los enunciados de las prácticas, para así facilitar el uso del programa a los alumnos. Cada descripción puede guardarse en un fichero para su uso posterior, o también puede cargarse una descripción almacenada previamente. Se dispone también de la opción de cargar ejemplos predefinidos, representativos del concepto en cuestión.

A la descripción introducida, se le pueden aplicar los algoritmos implementados, que se corresponden con la totalidad de los algoritmos programados en las prácticas de la asignatura más algunos otros interesantes estudiados en clase.

Describiremos a continuación cada uno de los módulos que componen *ToTalf*.

- Autómatas finitos y expresiones regulares. En este módulo, se contemplan autómatas finitos deterministas (AFD) y no deterministas (AFN), al igual que un subconjunto de las expresiones regulares extendidas. Se encuentran disponibles los siguientes algoritmos:
  - Transformación de AFN a AFD.
  - Eliminación de ϵ-transiciones.
  - Minimización de un AFD.
  - Obtención de un AFN a partir de una expresión regular (Construcción de Thompson).
  - Obtención de una gramática regular a partir de un AFD.
  - Obtención de los AFNs que reconocen $L(M_1) \cup L(M_2)$ y $L(M_1)L(M_2)$ a partir de los autómatas M_1 y M_2.

  En las expresiones regulares, se reconocen los operadores básicos (alternativas, concatenaciones, asociaciones y cierre), junto a varias características de las expresiones regulares extendidas: conjuntos de caracteres (p. ej. `[abc]`, `[a-c]`), el cierre positivo, el operador de interrogación y algunas clases de conjuntos de caracteres (p. ej. `[:alpha:]` y `[:digit:]`).
- Gramáticas independientes del contexto (GIC). En este módulo, los algoritmos implementados son:
  - Cálculo de los conjuntos *first* y *follow*.
  - Algoritmo de Cocke–Younger–Kasami para el análisis sintáctico de una frase.
  - Cálculo de la forma Normal de Chomsky.
  - Simplificación de una GIC.
  - Eliminación de la recursividad por la izquierda de una GIC.
  - Transformación de una gramática regular lineal por la derecha a un AFN.
- Autómatas a pila deterministas. En este módulo el único algoritmo disponible es el de la simulación de un autómata a pila determinista.
- Máquinas de Turing. Sólo se ha implementado la simulación de una máquina de Turing con cinta infinita en ambos sentidos.

```
// gramática que genera expresiones
// regulares simples
ER -> ER + ALT
ER -> ALT
ALT -> ALT PIEZA
ALT -> PIEZA
PIEZA -> ATOMO
PIEZA -> ATOMO *
ATOMO -> t
ATOMO -> ( ER )
```

Figura 1: Descripción de una gramática

3.3. Descripciones

En cada uno de los módulos de *ToTalf*, la entrada a los diferentes algoritmos viene dada por una descripción del modelo de computación correspondiente. Las descripciones tratan de ser lo más general y flexible posible. Así, por ejemplo, el número de caracteres prohibidos intenta ser mínimo, se permite el uso de comentarios en las descripciones, o en el caso

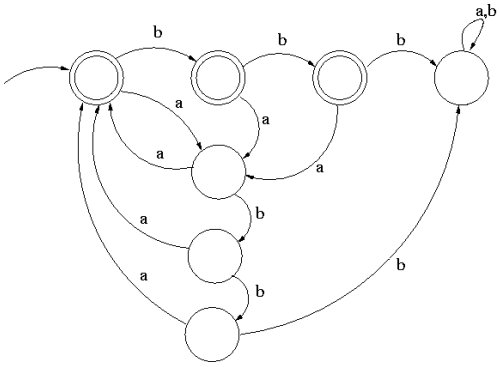


Figura 2: Autómata que reconoce cadenas con un n^{o} par de 'a' y que no contengan la subcadena 'bbb'

de las gramáticas, los símbolos pueden constar de más de un carácter.

Como ejemplo, la Figura 1 muestra el contenido de un fichero que describe una gramática que genera expresiones regulares.

3.4. Un ejemplo de uso con autómatas

Supongamos que el alumno desea resolver el siguiente problema: *Dado el alfabeto* $\Sigma = \{a, b\}$ *hallar un autómata finito con un número mínimo de estados que reconozca las cadenas de* Σ^* *que contengan un número par de símbolos 'a' y no contengan la subcadena 'bbb'.*

Cuando se les plantea este problema a los estudiantes, la mayoría obtienen directamente la solución, que se muestra en la Figura 2.

Sin embargo muchos otros alumnos tratan de obtener la solución siguiendo un proceso alternativo, en el que *ToTalf* es una herramienta muy útil a la hora de simplificar algunos de los cálculos que han de realizarse.

Para resolver este problema, puede resultar cómodo partir de dos autómatas que reconozcan los siguientes lenguajes:

$L_1 = \{w \in \Sigma^* | w$ *contiene un número par de símbolos 'a'*$\}$

$L_2 = \{w \in \Sigma^* | w$ *no contiene la subcadena 'bbb'*$\}$

Los AFD que reconocen L_1 y L_2 son los que aparecen en la Figura 3.

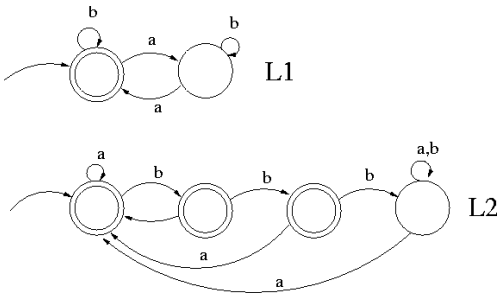


Figura 3: Autómatas que reconocen L_1 y L_2

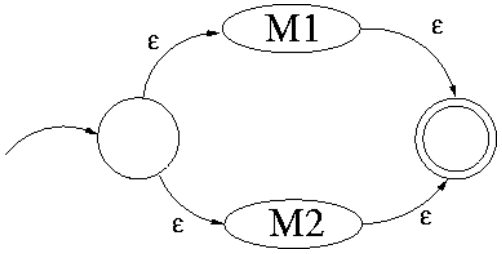


Figura 4: Autómata que reconoce $L(M_1) \cup L(M_2)$

El autómata que se nos pide puede obtenerse teniendo en cuenta la siguiente relación entre lenguajes, consecuencia de la aplicación de una de las leyes de Morgan:

$L_1 \cap L_2 = \overline{\overline{L_1} \cup \overline{L_2}}$

Dado un AFD M que reconoce un lenguaje L, para obtener el AFD que reconoce $\overline{L}$ basta con cambiar en M los estados que son de aceptación por estados que no sean de aceptación y convertir en estados de aceptación aquellos que en M no lo eran.

Por otra parte, partiendo de las descripciones de los autómatas M_1 y M_2 que reconocen $\overline{L_1}$ y $\overline{L_2}$ respectivamente, y utilizando uno de los algoritmos implementados en *ToTalf*, podemos obtener el AFN que reconoce $\overline{L_1} \cup \overline{L_2}$. La Figura 4 muestra el fundamento teórico de esta construcción. Si aplicamos a este autómata el algoritmo de conversión de un AFN en AFD y al resultado le aplicamos la minimización del número de estados de un AFD, obtendremos el AFD que se muestra en la Figura 2, que es la solución del problema planteado.

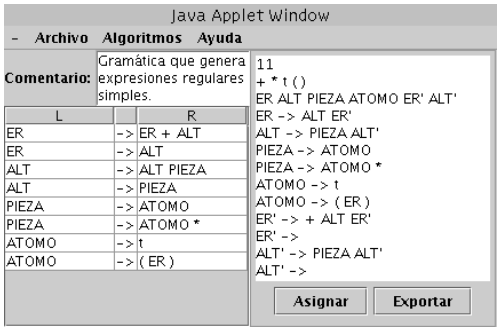


Figura 5: Gramática de expresiones regulares sin recursividad por la izquierda

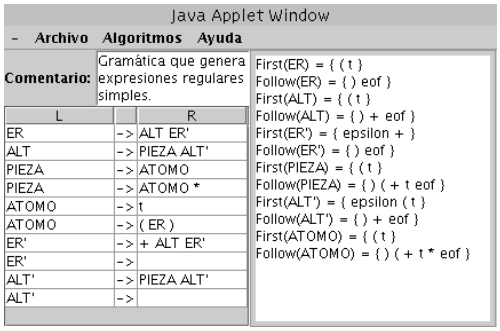


Figura 6: Conjuntos *First* y *Follow* de la gramática de expresiones regulares

3.5. Un ejemplo de uso con gramáticas

Supongamos otro caso real de trabajo: construir un analizador sintáctico descendente recursivo predictivo para la gramática representada en la Figura 1.

El primer paso consiste en introducir la gramática y aplicar el algoritmo de eliminación de la recursividad por la izquierda. Tras seleccionar el algoritmo adecuado en el menú *Algoritmos*, obtenemos el resultado en el marco de la derecha, tal y como se muestra en la Figura 5.

El siguiente paso sería comprobar que con esta gramática sin recursividad por la izquierda se puede construir un analizador descendente recursivo predictivo. Para ello, se debe cumplir que:
$\forall A \in V | A \rightarrow \alpha_i | \alpha_j$:
$First(\alpha_i) \cap First(\alpha_j) = \emptyset, \forall i \neq j$
Si $\alpha_i \Rightarrow^* \epsilon$, entonces
$First(\alpha_j) \cap Follow(A) = \emptyset, \forall j$
No existe actualmente un algoritmo implementado en *ToTalf* que realice esta comprobación, por lo que habría que hacerla a mano. No obstante, sí está implementado el cálculo de conjuntos *first* y *follow* de una gramática, por lo que la tarea es mucho más sencilla. Los conjuntos de *first* y *follow* se muestran en la Figura 6. Tras la correcta comprobación de que un analizador descendente recursivo predictivo puede funcionar con la gramática, el alumno puede empezar a codificar dicho analizador.

4. Conclusiones y trabajo futuro

La valoración que hacemos de los resultados del primer año de utilización de *ToTalf* en la docencia de *TALF* es absolutamente positiva. Las encuesta que hemos realizado entre el alumnado refleja que una mayoría de ellos han utilizado la herramienta y también mayoritariamente quienes la han utilizado están satisfechos con los resultados obtenidos. Para el próximo curso, una vez cubierta la etapa en la que la primera versión de *ToTalf* ha sido probada satisfactoriamente, planeamos utilizarla de forma más intensiva en el desarrollo de los contenidos de la asignatura.

Nos gustaría también destacar que *ToTalf* está siendo utilizado no sólo por los alumnos matriculados en *TALF* sino también por los alumnos de *Compiladores*, asignatura cuyos contenidos están muy cercanos a los desarrollados en *TALF*.

Por otra parte consideramos que el trabajo realizado hasta ahora debe ser considerado como un primer prototipo de la herramienta. Nos gustaría poder seguir perfeccionándola y sobre todo añadiéndole funcionalidades. La lista de posibles ampliaciones es tan amplia como los contenidos de las asignaturas de *TALF* y *Compiladores*, pero en concreto, tenemos ya en mente las siguientes extensiones:

- Obtención del autómata finito (AF) que reconoce el lenguaje complementario del reconocido por otro AF.

- Equivalencia de AFs.
- Simulación de Autómatas a pila no deterministas (APND).
- Obtención del APND asociado con una gramática independiente del contexto (GIC).
- Cálculo de la tabla LL(1) para una GIC.
- Obtención de la GIC que genera el lenguaje reconocido por APND.
- Simulación de máquinas de Turing con varias cintas.

Pero sin duda, una de los aspectos más interesantes a incluir es una interfaz gráfica para la definición de los autómatas, en vez del actual esquema de texto. La creación de autómatas sería más intuitiva, y el análisis de los autómatas resultado de los algoritmos, mucho más sencillo, lo que conseguiría un mejor entendimiento por parte del alumno.

Tal como ha sido concebido, el prototipo actual de *ToTalf* no es más que el núcleo básico de una herramienta que tiene toda la potencialidad de proseguir su desarrollo. En el caso de herramientas similares, lo único que podemos hacer es recomendar a nuestros alumnos su utilización, mientras que *ToTalf* es un prototipo sobre el que tenemos control total. También estamos considerando la posibilidad de liberar el código fuente de la aplicación, haciéndolo de dominio público, de forma que toda la comunidad académica de habla hispana pueda contribuir al desarrollo de la herramienta.

En resumen, consideramos satisfactorio el esfuerzo invertido en este proyecto de innovación docente. Pensamos que *ToTalf* es en estos momentos una de las herramientas de estas características más completas desarrollada en España, y que se ajusta plenamente a los objetivos con los que fue concebida.

Referencias

[1] J. Calera. *Applet Java para realizar diversas operaciones con autómatas finitos.* Departamento de Lenguajes y Sistemas Informáticos de la Universidad de Alicante. Disponible electrónicamente en `http://www.dlsi.ua.es/asignaturas/lga/applet/Afapplet.html`.

[2] Nicolas Christin. DFApplet a deterministic finite automata applet simulator. Disponible electrónicamente en `http://www.cs.virginia.edu/~nc2y/dfa/`.

[3] Francisco de Sande. *Páginas web de Teoría de Autómatas y Lenguajes Formales.* Escuela Técnica Superior de Ingeniería Informática. `http://asignaturas.pcg.ull.es/etsii/talf/`.

[4] Thomas Dunn. Interactive animation of finite-state automata. Disponible electrónicamente en `http://www.ccs.neu.edu/home/tdunn/COM1350/honors/`.

[5] Dynalab. Finite State Automaton Applet. Disponible electrónicamente en `http://www.cs.montana.edu/~dynalab/fsa/fsa.html`.

[6] Román Sosa González. ToTalf tutorial online de teoría de autómatas y lenguajes formales. Disponible electrónicamente en `http://asignaturas.pcg.ull.es/etsii/talf/ToTALF/Applet_ToTalf.html`.

[7] Java. Página principal: `http://java.sun.com`.

[8] Jbuilder downloads. Disponible electrónicamente en `http://www.borland.com/products/downloads/download_jbuilder.html`.

[9] Susan H. Rodger. JFLAP 4.0b10 version. Disponible electrónicamente en `http://www.cs.duke.edu/~rodger/tools/jflap/index.html`.

[10] MindSpring Software. ABCEZ (applet based computability enlightenment zone). Disponible electrónicamente en `http://www1.cs.columbia.edu/~zeph/ComputabilityApplets/abcez.html`.

ÍNDICE DE AUTORES