Mejora de la comprensión de las estructuras de datos

Raquel Lacuesta, Karmelo Urzelai
Departamento de Informática e Ingeniaría de Sistemas
Escuela Universitaria Politécnica de Teruel
Universidad de Zaragoza
lacuesta@unizar.es, karmelo@unizar.es

Resumen

El artículo presenta el desarrollo de animaciones interactivas para dar soporte a la docencia de la asignatura Estructuras de Datos, en una ingeniaría técnica en informática.

La asignatura consta de un amplio temario que hace que al alumno no le resulte fácil asimilar todos los modelos; el objetivo propuesto es facilitar su aprendizaje, así como animarle al autoestudio, poniendo en internet animaciones de soporte para su consulta.

Las animaciones desarrolladas se han clasificado en 8 grupos, cada uno de ellos asociado a un tema de la asignatura: costes computacionales, pilas, colas, listas, tablas, árboles, grafos y skip-lists. Para cada uno de estos grupos se describen las animaciones realizadas, mostrándose imágenes de algunas de ellas.

Se presenta un enfoque didáctico centrado en la motivación y refuerzo mediante animaciones interactivas que ayuden a la compresión de la asignatura.

1. Introducción

La asignatura de Estructuras de Datos se imparte en Teruel en el primer curso de la Ingeniería Técnica en Informática de Gestión. Los alumnos llegan a la asignatura únicamente con los conocimientos adquiridos en la asignatura de Programación I impartida en el primer cuatrimestre.

Los resultados de los primeros años de implantación de la titulación han sido poco satisfactorios debido al gran número de abandonos y a al escasa número de aprobados finales.

Analizando el problema, y dejando de lado motivos ajenos a la propia asignatura y su docencia en los que no se puede actuar, o en los que la resolución incumbe a otros entornos, las causas resultantes se han concentrado en dos aspectos: la amplitud del temario, y el todavía escaso nivel de programación y consecuente falta de capacidad de abstracción.

Analizando ambos aspectos, el temario, a pesar de ser amplio, aparte de ser coherente con la misma asignatura dada en otros centros, se considera adecuado a su número de créditos. De todas formas se ha realizado una adecuación en el tiempo dedicado a cada tema, quitando peso a algunos temas menos importantes, y dando más otros.

Por otro lado el nivel ofrecido en la asignatura previa de programación se considera correcto, y al menos de momento no se ha tomado en consideración desplazar la asignatura en el plan de estudios para dar una asignatura adicional intermedia.

2. Descripción de la asignatura.

La asignatura se divide en nueve temas, de los cuales cinco se dedican a la presentación de diferentes estructuras de datos, y donde para cada una de ellas se muestran varias implementaciones.

Los Tipos Abstractos de Datos (TAD) mostrados son: estructuras de datos lineales (pilas, colas, listas), tabla (*hashing*), árboles, grafos y *skip-lists*. [2][8]. Todos los conceptos teóricos se refuerzan en prácticas de laboratorio, que se desarrollan en el lenguaje de programación Ada 95 [1], donde deben implementarse variantes de los diferentes TADs explicados en las clases de teoría.

Lo que se pretende a lo largo del temario impartido es facilitar un esquema lógico para manipular los datos en función del problema que se deba tratar y el algoritmo que se utilice. Es importante escoger la estructura de datos y su

implementación más adecuadas, por lo que es fundamental conocerlas y comprenderlas.

Cabe destacar que se tratan tanto estructuras de datos estáticas (tamaño en memoria fijo) como dinámicas (tamaño en memoria variable), y que el planteamiento a la hora de explicarlas se debe adaptar a sus peculiaridades.

3. Motivación

El objetivo principal para la mejora de resultados, dentro del ámbito docente de la asignatura, ha sido intentar mejorar el entendimiento por parte de los alumnos de los diversos temas, intentado facilitarles la visualización de las diferentes estructuras de datos, y hacerles más ameno y llevadero el temario.

La explicación de algunas estructuras de datos implica mostrar al alumno cómo se mueven los datos, cómo cambian ciertos valores, como evolucionan por ejemplo los punteros, y todo ello es difícil hacer con medios estáticos como es la pizarra o las transparencias, que cómo mucho pueden mostrar imágenes inanimadas.

Se han desarrollado ya diversas aplicaciones para mejorar la compresión de algunas estructuras [7], queriendo aquí ampliar las posibilidades de una forma muy visual.

Por tanto tras considerarse insuficientes los medios utilizados hasta ahora, se decidió incorporar las nuevas tecnologías para mostrar la evolución de las estructuras de datos de forma animada. A su vez se consideró que una vez que se realizaba el esfuerzo sería interesante que estas animaciones no se vieran de forma pasiva sino que sería conveniente la implicación del alumno, haciendo que éstas fueran interactivas exigiendo al menos un mínimo de actuación por su parte, y en algunos casos añadiendo ventanas explicativas de los pasos seguidos.

La interactividad de las animaciones facilita un mejor entendimiento de los conceptos, frente a aquellas animaciones que sólo proporcionan una visualización del dinamismo de la estructura, ya que entre otras cosas permiten al alumno seguir la animación paso a paso de acuerdo a sus propias necesidades, y a su velocidad de asimilación.

Las animaciones han sido desarrolladas siguiendo el proceso de aprendizaje de un alumno que desconocía la asignatura, para plasmar desde una perspectiva más realista las dificultades encontradas y expresar de forma más cercana al alumno las ideas [6]. Lógicamente en todo momento el profesor supervisaba la corrección de las operaciones mostradas en las animaciones y su aspecto pedagógico. Así mismo es el profesor quien con su experiencia ha escogido las animaciones más representativas y las estructuras de datos e implementaciones que más dificultad presentan habitualmente para su aprendizaje.

Por otro lado se ha ajustado el conjunto de animaciones a las estudiadas dentro de la asignatura, abarcando todo el temario. Y no sólo se han desarrollado animaciones pensando en el comportamiento del T.A.D., sino que en algunos cosas la animación se ha centrado más en una posible implementación, que es la que más dificultad de comprensión tenía. Y todo ello se ha realizado buscando siempre la perspectiva más didáctica posible.

El empleo de animaciones se ha utilizado también en otras materias, por ejemplo para la animación y simulación de algoritmos paralelos de exploración de grafos[3]. También se han hecho animaciones para representar estructuras de datos por ejemplo en [7], no adecuándose a la asignatura Estructuras de Datos sino a la de Programación, por lo que resultan insuficientes para impartir un temario completo de la asignatura Estructuras de Datos.

4. Objetivos

Con esta iniciativa nos proponemos:

- Ofrecer al alumno un material de apoyo que facilite el aprendizaje. Todas las animaciones están puestas a disposición de los alumnos para repasar cuantas veces necesite el temario impartido. Cada animación irá adjunta al tema correspondiente.
- Incentivar al estudiante mediante apoyos visuales a preparar la asignatura. La unión de los temas teóricos a las animaciones gráficas facilita y ameniza el aprendizaje.
- Promover el uso de nuevas tecnologías por parte del alumno. Tanto los temas de la asignatura como las animaciones están en la página *web* de la asignatura con lo que el alumno puede acceder fácilmente a todo el material, tanto desde casa como desde la universidad
- Mejorar la compresión de conceptos durante la clase. El profesor puede utilizar las diversas animaciones durante las clases para hacer

comprender a los alumnos los temas explicados.

Como conclusión, la base principal para la mejora docente está en el apoyo que las animaciones pueden dar tanto durante las clases de teoría y prácticas, como durante la fase de autoaprendizaje que posteriormente el alumno lleve a cabo para asentar los conceptos explicados previamente.

Y por otro lado las animaciones son un gran apoyo para todas aquellas personas que no pudiendo asistir a las clases presenciales desean seguir la asignatura, ya que muchas veces los apuntes de otros alumnos resultan totalmente insuficientes para seguir una explicación oral que mostraba el flujo de datos dentro una estructura.

5. Animaciones

El entorno utilizado para el desarrollo de las animaciones ha sido Flash [4][5] debido a su facilidad de uso en la creación de películas dinámicas e interactivas, consiguiendo, además, una optimización en el tamaño de cada animación mediante la utilización de gráficos vectoriales.

Para la visualización de las animaciones desde las páginas *web* de la asignatura será necesario el programa *Flash Player*, gratuito, de fácil acceso y disponible en casi todos los navegadores.

Cada estructura de datos tiene su correspondiente animación, y algunas de ellas disponen de varias animaciones ya sea por su dificultad, o por la variedad de implementaciones. Dentro de cada animación se podrán realizar las operaciones pertinentes a cada estructura, consiguiendo así la interactividad del alumno con la animación y por lo tanto un refuerzo personalizado.

Para cada una de ellas se estudiaron las diferentes posibilidades de representación, escogiéndose la más intuitiva. El objetivo perseguido fue el de conseguir un efecto de autoaprendizaje y una fácil asimilación de los contenidos durante su visualización.

Dentro de las animaciones hay algunas cuyo objetivo es mostrar de forma detallada el comportamiento de la estructura y otras cuya única función es dar al estudiante la idea intuitiva de la estructura. Por ejemplo en la explicación de las colas una primera animación representa el comportamiento de una cola de coches en una parada, pudiendo añadir coches y quitar coches, lo que ya de por sí puede dar la idea del comportamiento de una cola mejor que ciertas explicaciones formales. Una segunda animación representa ya una secuencia de datos en los que podemos realizar las operaciones de *encolar* y *desencolar*.

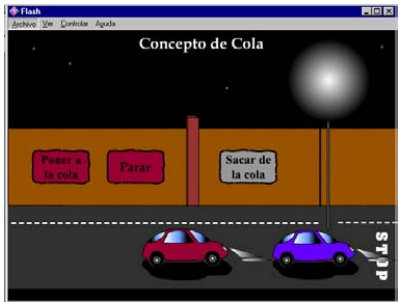


Figura 1. Concepto de cola

Todas las animaciones disponen de una serie de botones que permiten, dentro de ciertas restricciones, decidir la evolución de los datos, ver la animación paso a paso, reiniciar, o elegir entre diversas opciones. Algunas de ellas además disponen de cuadros de texto en los que se va explicando paso a paso las acciones que se están realizando y la fase en la que se encuentra.

En las animaciones que trabajan directamente con datos se ha mantenido una interfaz uniforme para darle homogeneidad, facilitar su seguimiento y no distraer al alumno.

Las animaciones realizadas [6] han sido:

1. Costes computacionales: La problemática de este tema es por un lado que su formalismo hace que el alumno sea más reacio a su estudio y por otro considerar que la eficiencia es un tema menor a causa de la creciente potencia de los ordenadores. La animación desarrollada por tanto incide en ambos aspectos de modo que por un lado presenta gráficamente los diferentes tipos de costes, y a continuación de una forma visual muestra las grandes diferencias entre unas y otras, para que el alumno sea consciente de la importancia de elegir un algoritmo de un coste razonable.
2. Pilas: Las animaciones de estructuras comienzan con ésta primera, muy sencilla, y

que introduce al alumno en la utilización de las animaciones que encontrará más tarde, que serán más complejas y elaboradas. En este caso una primera animación ofrece la idea intuitiva de pila por medio del apilamiento y desapilamiento de unos cubos, y otra animación posterior ya muestra el comportamiento más formal de una estructura de datos *pila*, con sus operaciones de *apilar*, *desapilar*, *cima* y *vacía*. Esta segunda animación ya introduce al alumno en la estética general de las animaciones.

3. Colas: A pesar de ser también una estructura muy sencilla se han definido dos animaciones, al igual que en las pilas una primera para aportar la idea intuitiva de *cola* (Figura 1), y una segunda para la definición más formal. Se intenta así con estas primeras lecciones atraer al alumno a "jugar" con las animaciones, de modo que posteriormente con animaciones más complejas le resulte más natural.
4. Listas con punto de interés: Para este caso, siguiendo la línea de las dos anteriores, se ha creado una primera animación para entender el concepto por medio de un pequeño editor de texto simulado. Y por otro se han construido una serie de animaciones orientadas a mostrar el comportamiento de una *lista* implementada de forma encadenada. Pese a explicarse en el temario también otro tipo de implementaciones se ha considerado innecesario realizar animaciones para ellas debido a su simplicidad.

   De las animaciones realizadas se ha incidido en los aspectos más críticos como la creación de la lista, para que se entiendan los conceptos de apuntador y de pila de posiciones libres. Otras de las animaciones muestran las operaciones más habituales de las listas y que pueden resultar difíciles de seguir sólo con una explicación oral, como el avance del *Punto de Interés* (Figura 2), la inserción de un nuevo elemento o el borrado. Las animaciones avanzan paso a paso, bajo el control del alumno, para que vea los cambios producidos tanto en el encadenamiento de la lista como en la gestión del espacio libre.

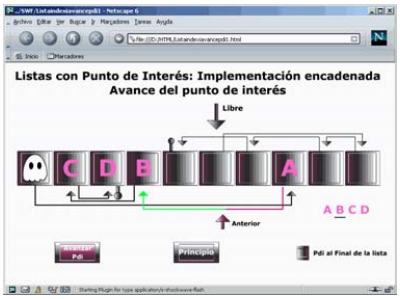


Figura 2. Listas con Punto de Interés

5. Tablas *Hash*: Parte de la dificultad de esta estructura de datos es mostrar el comportamiento de los tres tipos básicos de implementaciones que pueden darse a las tablas de dispersión: encadenamiento indirecto, encadenamiento directo y no encadenadas. Por ello se han realizado animaciones que muestran cómo se pueden implementar cada una de ellas y su comportamiento, mostrándose por ejemplo para las primeras, en tres animaciones, la inserción de un elemento en la tabla, el borrado de elementos y la consulta de si un elemento pertenece a la tabla (Figura 3).

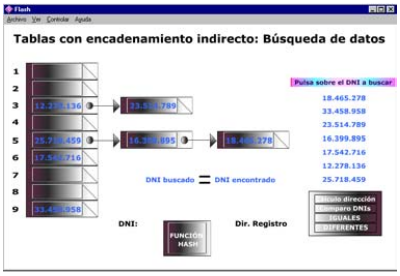


Figura 3. Búsqueda en tabla *hash*

6. Árboles: En el tema de árboles a modo de ejemplo se ha realizado una animación, que llama a otras tres animaciones, que muestran los tres tipos de recorrido en profundidad (Figura 4), y otra animación para mostrar el recorrido en anchura. A causa de que la implementación del recorrido en anchura de los árboles es un poco más complejo por la necesidad de una cola

que permita visitar los nodos en el orden correcto, en la animación a medida que se recorre el árbol se muestra también paralelamente paso a paso la evolución de la cola que nos permite realizar dicho recorrido

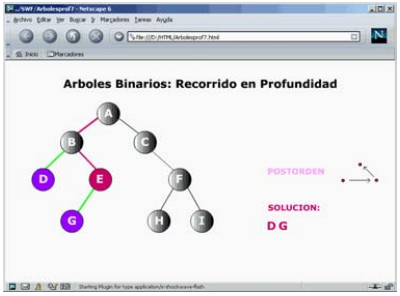


Figura 4. Recorrido en árboles

7. Grafos: Como ejemplo de utilización de grafos se ha realizado una animación que muestra la ejecución del algoritmo de Dijkstra. En dicha animación se avanza poco a poco seleccionando el siguiente nodo del camino de coste mínimo, y calculando los costes de los caminos a los demás nodos. Se muestra de forma explícita la tabla de costes y los caminos que los generaron, al mismo tiempo que se representa el grafo y se van marcando dinámicamente los diferentes caminos que se están calculando.
8. Skip List: Para mostrar esta estructura de datos, se ha dibujado una *skip-list* sobre la que se ha realizado el proceso de inserción de un elemento (Figura 5). Para la inserción en primer lugar se hace la búsqueda del punto de inserción, con lo que se está mostrando también cómo funciona ésta operación, pero para la posterior inserción además se van guardando los enlaces para poder redireccionarlos después.

6. Ejemplo de Animación

A modo de ejemplo se muestra paso a paso una de las animaciones asociadas a la estructura más simple de las realizadas: la pila.

Como ya se ha dicho, hay una primera animación que muestra la idea intuitiva de pila por medio de unos cubos etiquetados con letras, y la que aquí se explica es la segunda, que muestra, con la estética general de las animaciones, el comportamiento del TAD pila y sus operaciones.

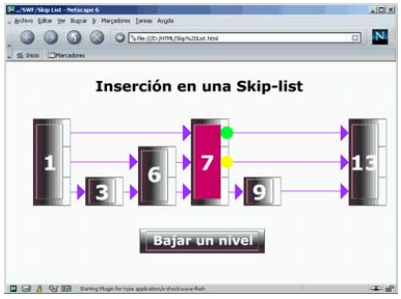


Figura 5. Skip-List

La animación comienza con una pila vacía, y el indicador *Pila Vacía* parpadeando para mostrar que está activado. El marcador de la cima de la pila está vacío al no haber ningún elemento en ella (Figura 6).

Figura 6. Pila vacía

Si se pulsa el botón de *Apilar*, el primer dato aparece por la izquierda y se desplaza hasta entrar en la pila y colocarse como cima. El indicador de *Pila Vacía* se desactiva y el dato introducido pasa a mostrarse en la ventana *Cima de Pila* (Figura 7).

Figura 7. Apilado de un dato

Si se pulsa *Apilar* más veces el proceso se repite llenándose poco a poco la pila, y modificándose el marcador de la cima de la pila (Figura 8).

Figura 8. Apilado de tres datos

En cualquier momento se puede pulsar también el botón de *Desapilar*, en cuyo caso el elemento de la cima, sale de la pila y desaparece por la izquierda (Figura 9).

Figura 9. Desapilando un dato

En el caso de desapilarse todos los elementos vuelve a encenderse el indicador de *Pila Vacía*.

El resto de animaciones pueden verse en http://eupt.unizar.es/eda/temario/Temario.html [1]

7. Conclusiones

Ante la problemática planteada en la asignatura Estructuras de Datos en la que los alumnos abandonaban fácilmente a causa de una temática amplia y la introducción de algunos conceptos nuevos, se ha decido apoyar tanto las clases como el autoestudio con animaciones.

Las animaciones se han realizado en Flash por su versatilidad, facilidad de uso y por ser prácticamente un estándar de *Internet*, ya que uno de los destinos principales de dichas animaciones es colocarlas en las páginas web de la asignatura, de modo que los alumnos puedan consultarlas en cualquier momento. También podrán ser utilizadas durante las clases para las explicaciones.

Las animaciones han sido realizadas por una persona que desconocía previamente la asignatura de modo que pudiera plasmar sus propias dudas y problemas durante el aprendizaje en las mismas animaciones. Lógicamente dichas animaciones en todo momento han estado guiadas por el profesor de la asignatura que además de aportar la experiencia sobre las dudas más habituales de los alumnos y controlar la corrección de todo lo realizado, volcaba su experiencia docente en la búsqueda de la representación más pedagógica posible

A lo largo del segundo cuatrimestre del curso 2002-2003 se van a utilizar las animaciones desarrolladas como herramienta docente, esperando una buena recepción del esfuerzo desarrollado, siendo positiva la impresión de los alumnos que ya las han visto. De todas formas con la realimentación de la reacción y la opinión de los alumnos se espera mejorar, adecuar y ampliar este primer paso.

Por último, consideramos que con este desarrollo no se completa ningún trabajo, sino al contrario la experiencia nos ha mostrado que se abren las puertas para la utilización de animaciones en muchas otras asignaturas, en áreas que pueden ser completamente diferentes.

[1] Prohibida la reproducción, copia o utilización de cualquier información de la web, sin la expresa autorización de su autor

Agradecimientos

Los autores de este artículo desean agradecer a Celia Marzo su dedicación y desarrollo de las animaciones. También desean agradecer a Elvira Mayordomo, del mismo Dpto., la iniciativa y desarrollo del proyecto, llevando ella posteriormente una línea totalmente independiente a pesar de compartir objetivos.

Referencias

[1] Barnes J. "Programmimng in Ada 95". Addison Wesley. 1999

[2] Franch X. "Estructuras de datos. Especificación, diseño e implementación". Edición UPC.

[3] José Fco. Cachairo González, Manuel Díaz Rodríguez, Antonio Vallecillo Moreno . "Animación y Simulación de Algoritmos Paralelos de Exploración de Grafos".

[4] Macromedia. "Manual Macromedia Flash 5. Guía de Consulta de Actionscript". Macromedia Inc.

[5] Macromedia . "Manual Macromedia Flash 5. Utilización de Flash". Macromedia Inc.

[6] Mazo C. "Animaciones Flash para la asignatura Estructura de Datos". Trabajo Fin de Carrera. Escuela Universitaria Politécnica de Teruel.

[7] Salamó M., Camps J., Vallespí C., Vernet D., Llorá X., Bernadó E., Garrell J.M., González X.. "Iniciativas para motivar a los alumnos de Programación". Universitat Ramon Llull.

[8] Weiss M.A. "Estructuras de datos y algoritmos". Addison-Wesley-Iberoamericana.

Análisis híbrido: una propuesta práctica

Francisco Palomo Lozano, Inmaculada Medina Bulo
Dpto. de Lenguajes y Sistemas Informáticos. Universidad de Cádiz.
Escuela Superior de Ingeniería de Cádiz. C/ Chile, s/n. 11003 Cádiz.
{francisco.palomo, inmaculada.medina}@uca.es

Resumen

Este artículo defiende la necesidad de introducir el *análisis híbrido* en las prácticas de las asignaturas en las que se imparte análisis y diseño de algoritmos secuenciales, y presenta un entorno de trabajo adecuado para su realización en el laboratorio.

Esta técnica de análisis es una mezcla de las técnicas tradicionales de *análisis teórico y empírico*: en el análisis híbrido se emplea la primera de ellas para establecer un modelo y la segunda para obtener datos experimentales que, por último, permiten ajustar el modelo mediante regresión.

1. Introducción

En las recomendaciones ACM/IEEE-CS del año 1991 [1] se sugiere explícitamente que dentro del área AL de *algoritmos y estructuras de datos* y, en concreto, en las unidades AL4 (análisis de la complejidad), AL5 (clases de complejidad) y AL6 (ordenación y búsqueda), se realice la

> *...medida de los tiempos de algoritmos seleccionados de distintas clases de complejidad...*

y también la

> *...corroboración de la complejidad teórica mediante el empleo de métodos experimentales...*

como parte de las prácticas de laboratorio a desarrollar siguiendo un modelo de *laboratorio cerrado*.

Actualmente, tras la última revisión realizada en el 2001 [2], el área AL ha pasado a ser *algoritmos y complejidad* y en AL1 (análisis básico de algoritmos) aparece explícitamente recomendada la

> *...medida empírica del rendimiento de los algoritmos*.

Como expondremos a continuación, limitar los métodos experimentales a un *análisis empírico* presenta claros inconvenientes.

Por otro lado, pensamos que un enfoque más integrado en el que se emplee un método de *análisis híbrido* ayuda a que el alumno aplique en el laboratorio una parte mayor de los conocimientos sobre *análisis teórico* adquiridos en las clases de teoría.

Nos centraremos en el análisis del tiempo de ejecución, aunque el método descrito es aplicable al análisis de otros recursos computacionales (como el espacio) siempre que exista una manera sencilla de medir su consumo en la práctica.

2. Técnicas de análisis

Siguiendo a [4], un texto de análisis y diseño de algoritmos de amplio uso, las dos técnicas principales de análisis de algoritmos son el *análisis teórico* y el *análisis empírico*. De la mezcla de ambas surge el *análisis híbrido*.

2.1. Análisis teórico

En el análisis teórico se fija una determinada operación patrón y se calcula una función matemática que mide el número de instrucciones de dicho tipo que ejecuta el algoritmo frente a una determinada medida del tamaño de su entrada.

Decimos que una operación es *crítica* cuando su función asociada tiene al menos el mismo orden que la correspondiente a cualquier otra operación del algoritmo y *elemental* si en su implementación su tiempo de ejecución tiene un orden constante.

Cuando el algoritmo se analiza respecto de una operación crítica, el *principio de invariancia* garantiza que el orden de la función que mide el tiempo de ejecución de cualquier implementación del algoritmo (en la que dicha operación sea elemental) coincide con el del algoritmo.

Es decir, el resultado es independiente de la máquina, del lenguaje de programación y del programador. En tanto en cuanto se mantengan las condiciones descritas, los resultados podrán extrapolarse a la implementación: el orden del tiempo abstracto del algoritmo corresponderá con el del tiempo físico de ejecución del programa.

La principal desventaja radica en que un análisis teórico preciso puede ser muy complejo. Por ejemplo, el análisis de la familia de algoritmos de ordenación de Shell no se ha completado, pese a ser un método clásico sobre el que se ha investigado exhaustivamente.

Afortunadamente, ocurre en muchos casos que, aunque el análisis preciso de la función puede ser muy complicado, la obtención de su orden puede realizarse con un esfuerzo bastante menor.

2.2. Análisis empírico

En el análisis empírico se mide el tiempo de ejecución de una implementación para un número suficiente de datos de entrada. La gráfica resultante de representar los tiempos frente al tamaño de la entrada puede emplearse para comprobar la tendencia del algoritmo y compararlo con otros.

Su principal ventaja es que proporciona datos reales de ejecución para una determinada implementación. Las desventajas de este método son evidentes. Desde un punto de vista teórico, y a falta de otra información, no podemos predecir nada acerca del comportamiento del algoritmo fuera de los valores observados.

Desde un punto de vista práctico, se puede perder una gran cantidad de tiempo en implementar un algoritmo muy ineficiente que no nos interesa en absoluto. En muchas ocasiones este tiempo no compensará al que se hubiera empleado en realizar un análisis teórico con el fin de rechazar el algoritmo como impracticable.

Incluso si el algoritmo es ineficiente pero interesante, como en el caso de los que resuelven problemas inherentemente complejos (por ejemplo, problemas NP-completos), este enfoque presenta sus inconvenientes, ya que un análisis empírico requerirá ingentes cantidades de tiempo para recabar los datos experimentales.

2.3. Análisis híbrido

En el análisis híbrido se realiza primero un análisis teórico encaminado a determinar la forma aproximada de la función de tiempo y posteriormente un análisis empírico para recabar datos. Finalmente se completa nuestro conocimiento de la función con los datos extraídos experimentalmente.

Se observa que podemos descomponer el proceso de análisis en tres etapas: modelado, experimentación y regresión.

Modelado

Es necesario establecer un modelo aproximado del comportamiento de la función de tiempo.

Una forma inicial de establecer dicho modelo es emplear un criterio asintótico. La ventaja de este enfoque es que únicamente necesitamos conocer el orden del algoritmo.

Supongamos que el tiempo del algoritmo viene dado por $t(n)$, siendo n el tamaño de la entrada, y que hemos determinado que $t(n) \in \Theta(f(n))$, pese a desconocer la expresión exacta de $t(n)$.

En la gran mayoría de los casos que se nos presentan en la práctica $t(n) \in \Theta(f(n))$ porque

$$\lim \frac{t(n)}{f(n)} = a \in \mathbb{R} \ .$$

En tales casos, podemos tomar $\hat{t}(n) = af(n)$ como aproximación de $t(n)$ para valores de n lo suficientemente grandes, es decir, $\hat{t}(n)$ será un modelo de la función desconocida $t(n)$.

La desventaja es que, al ser un criterio asintótico, la estimación sólo será realmente precisa para valores de n lo suficientemente grandes, es decir, a partir de un cierto umbral. Con frecuencia, este umbral es pequeño y se obtienen buenos resultados incluso para valores pequeños de n.

Si el algoritmo es lo suficientemente sencillo y no sólo se conoce el orden de $t(n)$ sino que también se conoce su expresión exacta, pueden emplearse modelos más detallados, con más parámetros.

Experimentación

El algoritmo debe implementarse con sumo cuidado, de manera que el programa obtenido refleje fielmente la idea subyacente sin introducir costes ocultos adicionales que no hayan sido tenidos en cuenta durante el modelado.

Pero esto no es suficiente. Cualquier hipótesis sobre los datos de entrada que haya sido considerada durante el modelado (por ejemplo, al calcular el orden) debe ser reflejada al seleccionar los datos de entrada para el experimento.

Por ejemplo, al analizar en el caso promedio un algoritmo de ordenación por comparación para vectores, es muy común que se suponga que todos los elementos del vector de entrada son distintos y sus permutaciones equiprobables. A menos que los vectores de entrada para el experimento se seleccionen de acuerdo con dichos criterios, el resultado no será significativo.

La medida de los tiempos también ha de realizarse con cuidado. Es posible que los tiempos que estemos midiendo sean menores que la resolución de nuestro aparato de medida. Por ejemplo, hemos comprobado que, en nuestro sistema, la resolución de la función estándar `clock()` es muy baja: aproximadamente $0{,}01$ s. Si no disponemos de un instrumento más preciso, podemos realizar una medida indirecta, midiendo el tiempo total de un número suficiente de repeticiones del experimento y promediando el resultado. Es muy importante que para cada repetición de un mismo experimento se empleen exactamente los mismos datos de entrada.

Siguiendo con el ejemplo, basta medir el tiempo total de 10 repeticiones de un mismo experimento y dividirlo entre 10 para obtener su tiempo con una precisión de $0{,}001$ s.

Regresión

Una vez que se dispone de los datos experimentales, hay que estimar los parámetros de regresión que aparecen en el modelo. Nótese que, salvo en casos muy sencillos, el modelo no es lineal. Por lo tanto, una regresión lineal simple no es aplicable.

Los métodos de tipo LLS (*linear least squares*) permiten, a partir de una función $\hat{y}(x) = ax + b$ y un conjunto de observaciones (x_i, y_i), obtener los parámetros a y b que proporcionan el mejor ajuste del modelo a las observaciones en el sentido de minimizar el error cuadrático medio [6, 9].

El método se denomina lineal no porque lo sea el modelo respecto de x, sino porque lo es respecto de a y b. De hecho, puede generalizarse para tratar modelos del siguiente tipo:

$$\hat{y}(x) = \sum_{k=1}^{n} a_k f_k(x) \ ,$$

donde las $f_k(x)$ son funciones arbitrarias.

Los métodos de tipo NLLS (*non-linear least squares*) son más generales y permiten trabajar prácticamente con cualquier tipo de modelo. El más utilizado por los diversos paquetes especializados es el de Levenberg-Marquardt [7, 8, 9].

3. El entorno de trabajo

Nuestro entorno de trabajo en un laboratorio equipado para realizar prácticas de las características descritas está constituido íntegramente por herramientas de software libre. Todas pueden encontrarse en cualquiera de las distribuciones habituales de LINUX.

Sobre todo empleamos GNU MAKE, GNU C++ y GNU PLOT, que pertenecen todas al proyecto GNU. Son herramientas potentes, estables, muy probadas y, sobre todo, gratuitas, algo muy importante si queremos que el alumno pueda trabajar en casa sin tener que realizar un costoso desembolso adicional.

MAKE nos ayuda a organizar el trabajo, no sólo de compilación, sino de experimentación. Nuestra experiencia demuestra que emplear un poco de tiempo en explicar esta herramienta, y suministrar a los alumnos unos apuntes no muy extensos sobre ella, ayuda a mejorar el rendimiento de éstos en el laboratorio.

GNU PLOT nos permite representar gráficamente los resultados experimentales, contenidos en ficheros de texto, frente a los modelos, suministrados como funciones, y comparar «a ojo» la bondad del ajuste (orden `plot`). Además, implementa internamente el método de Levenberg-Marquardt con lo que nos permite ajustar los parámetros de regresión de modelos muy complejos (orden `fit`) sin tener que emplear herramientas externas.

Otra ventaja que presenta es que es programable mediante *guiones* que pueden ser ejecutados desde la línea de órdenes o, automáticamente, desde MAKE.

3.1. ¿Por qué C++?

C++ [5, 10] es un lenguaje multiparadigma que permite programar en una variedad de estilos. Podemos diferenciar tres paradigmas básicos dentro del lenguaje que pueden mezclarse entre sí para lograr aún más flexibilidad. Esto hace que el lenguaje se adapte bien a una gran variedad de itinerarios curriculares.

Por un lado, puede emplearse como un lenguaje estructurado. En este sentido, puede considerarse que C++ es un C mejorado con un sistema de tipos más estricto que el de C, espacios de nombres separados, un mecanismo de control de excepciones y sobrecarga de funciones y operadores.

Por otro lado, implementa el paradigma de la programación genérica a través del concepto de *plantilla*. Las plantillas son definiciones paramétricas que permiten un alto grado de abstracción.

Por último, permite programar utilizando orientación a objetos. A diferencia de los lenguajes orientados a objetos puros, C++ posee herencia múltiple y el enlace es estático por omisión.

La posibilidad de sobrecargar el operador de llamada a función dentro de una clase, permite además la creación de *objetos función* que pueden pasarse como parámetro dotando a C++ de características propias de los *lenguajes de orden superior*.

No es necesario para el alumno dominar el lenguaje para sacar partido de él. Mostrarle un subconjunto adecuadamente escogido en relación a sus conocimientos previos puede ser suficiente. En una asignatura donde se enseña análisis y diseño de algoritmos, los alumnos aprecian la eficiencia del lenguaje como un valor añadido. C++ fue diseñado expresamente para tal fin.

Otra razón de peso para elegir C++ es la existencia de una *biblioteca estándar de plantillas* asociada al lenguaje que permite trabajar cómodamente con contenedores y algoritmos genéricos. Se trata de la STL [3, 5]: la única biblioteca que conocemos que forma parte de un estándar internacional y que incluye como parte de su especificación (normativa para todas las implementaciones) requisitos de complejidad. Esto la hace especialmente apropiada para su empleo en un laboratorio de análisis y diseño de algoritmos.

La STL se incorporó al lenguaje tras aprobarse una propuesta de A. A. Stepanov y M. Lee en una reunión del comité ANSI/ISO para la estandarización de C++ celebrada en julio de 1994. El estándar ISO/IEC [5] se aprobó finalmente en 1998.

3.2. ¿Por qué la STL?

La biblioteca estándar de plantillas es una biblioteca de clases contenedoras, iteradores y algoritmos. Proporciona implementaciones muy eficientes de estructuras de datos y algoritmos que se necesitan habitualmente en el desarrollo de programas más complejos y es el resultado de años de investigación desarrollada por sus autores sobre *programación genérica*.

El que la STL sea una biblioteca especialmente diseñada para la programación genérica se refleja en el hecho de que prácticamente todos sus componentes son paramétricos.

Existen otras bibliotecas escritas en C++, algunas muy completas, como LEDA, que podrían utilizarse para nuestros propósitos. Frente a ellas, STL presenta la ventaja de formar parte de cualquier implementación de C++ que cumpla con el estándar. La mayoría de los fabricantes de compiladores tienden a que sus productos se encuentren en dicho estado, con lo que incluyen implementaciones de calidad de la STL integradas en la distribución de sus compiladores. Con esto se facilita la creación de código transportable entre distintas plataformas.

Pero, sin duda, una de las características más sorprendentes e innovadoras de la STL es que como parte de su especificación incluye la complejidad asintótica temporal. Esto significa que cualquier fabricante de compiladores o bibliotecas que desee cumplir el estándar de C++ está obligado a proporcionar implementaciones de sus algoritmos que cumplan ciertos requisitos de eficiencia.

4. Conceptos clave de la STL

A continuación se exponen de manera general, los aspectos más relevantes que presenta la STL y que facilitan nuestra tarea a la hora de analizar y diseñar algoritmos escritos en C++.

4.1. Especificaciones de complejidad

Todas las operaciones de la STL poseen una especificación de complejidad. Ésta puede venir dada por un orden asintótico o, en los casos más simples, por una función concreta. La mayoría de las veces se especifica la complejidad del peor caso y, en ocasiones, la del caso promedio.

Otras veces, la complejidad temporal se especifica mediante un *análisis amortizado*. Esto es útil cuando el tiempo de una operación puede sufrir grandes variaciones a lo largo de una secuencia de operaciones. En este caso un análisis en el peor caso podría ser excesivamente pesimista y un análisis en el promedio, poco significativo, por la gran desviación de los tiempos.

4.2. Contenedores

Los contenedores son objetos que se utilizan para almacenar otros objetos (incluso otro contenedor) proporcionando operaciones para su manipulación.

La STL posee diversas clases contenedoras agrupadas en dos categorías: secuencias (como los vectores) y contenedores asociativos ordenados (como los conjuntos). También posee adaptadores de secuencia (como las pilas) que se comportan como contenedores especializados. Los nombres de las clases y su descripción aparecen en el siguiente cuadro:

`vector`	Vector
`deque`	Cola doble
`list`	Lista
`set`, `multiset`	Conjunto y multiconjunto
`map`, `multimap`	Asociación mono/multivalor
`stack`	Pila
`queue`	Cola simple
`priority_queue`	Cola simple de prioridades

4.3. Iteradores

Los iteradores son una generalización del concepto de puntero y se emplean principalmente para recorrer un contenedor. Muchos algoritmos manejan rangos de iteradores.

El rango $[i, j)$ representa a todos los elementos comprendidos entre los iteradores i y j sin incluir al último. Si `c` es un contenedor, todos sus elementos pueden representarse mediante el rango $[$`c.begin()`, `c.end()`$)$.

Existen distintos tipos de iteradores y cada contenedor proporciona los apropiados para que puedan emplearse algoritmos eficientes sobre ellos.

Los vectores y las colas dobles proporcionan iteradores de acceso directo. Esto significa que se puede acceder a cualquier elemento en tiempo constante, es decir, que el acceso se puede considerar como una operación elemental. Sin embargo, los iteradores de las listas y los contenedores asociativos son únicamente bidireccionales, ya que no es posible realizar acceso directo a un elemento de estos contenedores en tiempo constante.

Así, un algoritmo genérico diseñado eficientemente para emplear acceso directo, como es el caso del algoritmo de ordenación `sort()` de la STL, sería ineficiente si se aplicara a una lista que proporcionara iteradores de «acceso directo» de complejidad $O(n)$. Por lo tanto, las listas no proporcionan tales iteradores y para compensar poseen su propia función `sort()`.

4.4. Algoritmos

Hemos visto que la STL define contenedores que incluyen algoritmos específicos, pero también define algoritmos que son independientes del contenedor, en el sentido de que pueden emplearse sobre cualquiera que cumpla unos determinados requisitos.

La genericidad de los algoritmos independientes del contenedor puede descomponerse en tres factores ortogonales de diseño: son paramétricos, reciben iteradores (en lugar de contenedores) y pueden recibir objetos función.

5. Algunos ejemplos concretos

Vamos a aplicar las técnicas previamente descritas al análisis en el promedio de tres algoritmos de ordenación tal y como debería hacerlo un alumno en el laboratorio.

Las funciones en C++ que se presentan a continuación implementan distintos algoritmos genéricos que ordenan un rango $[i, j)$ de iteradores de acceso directo. En adelante, $n = j - i$ representará el número de elementos del rango.

La función `merge_sort()` implementa el algoritmo de ordenación por fusión. La STL define `inplace_merge()` que funde dos rangos ordenados $[i, k)$ y $[k, j)$ en no más de $n - 1$ comparaciones si hay memoria suficiente. Supondremos que la hay, en caso contrario, `inplace_merge()` puede emplear hasta $O(n \log n)$ comparaciones.

```
template <typename I>
void merge_sort(I i, I j)
{
  if (j - i > 1) {
    I k = i + (j - i) / 2;
    merge_sort(i, k);
    merge_sort(k, j);
    inplace_merge(i, k, j);
  }
}
```

La función `median_quick_sort()` implementa una variante del algoritmo de ordenación rápida, de Hoare. En esta variante se emplea la mediana como pivote. La STL define `nth_element()` que emplea un tiempo promedio lineal en reorganizar los rangos $[i, k)$ y $[k, j)$ de forma que los elementos del primer rango no sean mayores a los del segundo y que en k quede el elemento que ocuparía dicha posición si $[i, j)$ estuviera ordenado.

```
template <typename I>
void median_quick_sort(I i, I j)
{
  if (j - i > 1) {
    I k = i + (j - i) / 2;
    nth_element(i, k, j);
    median_quick_sort(i, k);
    median_quick_sort(k, j);
  }
}
```

La función `heap_sort()` implementa el algoritmo de ordenación por montículo, de Williams. La STL define `make_heap()`, que construye un montículo en no más de $3n$ comparaciones, y también `sort_heap()`, que lo ordena en $n \log_2 n$ comparaciones.

```
template <typename I>
void heap_sort(I i, I j)
{
  make_heap(i, j);
  sort_heap(i, j);
}
```

Primero se elige el modelo. Estamos interesados en un análisis en el promedio. No es difícil deducir, aun sin conocer la expresión exacta de $t(n)$, que los tres algoritmos realizan $\Theta(n \log n)$ comparaciones en tal caso. Así, según el criterio asintótico, podemos tomar $\hat{t}(n) = an \ln n$ como modelo.

En segundo lugar, hemos de realizar los experimentos. Hay que cronometrar el tiempo entre dos puntos de un programa, para lo que creamos una clase que emplea la función `clock()` de la biblioteca para medir el tiempo por diferencia. Aquí `CPS` es `double(CLOCKS_PER_SEC)`, el número de unidades internas de tiempo por segundo.

```
class Cronometro {
  clock_t t0;
public:
  Cronometro() { t0 = clock(); }
  double tiempo()
  { return (clock() - t0) / CPS; }
};
```

Con el siguiente programa obtenemos los datos para un análisis en el promedio. Por cada n se crea un vector de n elementos que se rellena con los valores $1, \ldots, n$. Entonces, se permuta aleatoriamente obteniendo una de las $n!$ permutaciones posibles. Así es como se selecciona la entrada para cada experimento.

```
int main()
{
  for (int n = N0; n <= N; n += I) {
    vector<double> v(n);
    generate(v.begin(), v.end(), G());
    random_shuffle(v.begin(), v.end());
    vector<double> t = v;
    long int r = 0;
    Cronometro c;
    do {
      ORDENAR(v.begin(), v.end());
      v = t;
      ++r;
    } while (c.tiempo() < 1.0);
    cout << n << '\t'
         << c.tiempo() / r << endl;
  }
}
```

La medida del tiempo es adaptativa. Cada experimento se repite durante, al menos, 1 s. Esto asegura que si el experimento dura menos de $0{,}01$ s se repita al menos 100 veces, permitiendo obtener una precisión de $0{,}1$ ms.

En el programa, `ORDENAR` representa a cualquier algoritmo que ordene un rango. Por otro lado, `N0` es el tamaño inicial, `N` el final e `I` el incremento. Para el experimento escogimos los valores `500`, `50000` y `1000`, respectivamente.

Como se observa, `generate()` requiere un objeto función para poder generar la secuencia $1, \ldots, n$. Para ello creamos la clase `G`, en la que se sobrecarga el operador de llamada a función.

```
class G {
  double i;
public:
  G(): i(1.0) {}
  double operator ()() { return i++; }
};
```

Por último, un sencillo guión para GNU PLOT permite realizar la regresión y obtener una gráfica del resultado frente a las observaciones. Por ejemplo, para `heap_sort()` hacemos:

```
set dummy n
t(n) = a * n * log(n)
fit t(n) "heap_sort.dat" via a
plot "heap_sort.dat", t(n)
```

En este caso `fit` calcula que $a = 83{,}0 \cdot 10^{-9}$, con lo que podemos estimar que el programa tarda un tiempo de $83{,}0\, n \ln n$ ns $\approx 57{,}5\, n \log_2 n$ ns. Análogamente, se obtiene $a = 133{,}1 \cdot 10^{-9}$ para `median_quick_sort()` y $a = 192{,}3 \cdot 10^{-9}$ en el caso de `merge_sort()`.

En los tres casos el sistema nos informa de que la desviación típica de los errores es inferior a $0{,}002$ y de que el error asintótico para el parámetro estimado es inferior al $1\,\%$. Esto da una idea de la bondad del modelo. En la figura 1 se presenta una gráfica comparativa que muestra los datos experimentales y su regresión.

6. Evaluación de la propuesta

Actualmente utilizamos exclusivamente análisis empírico en nuestras prácticas. Parece que ésta es la situación general en la Universidad española, donde no tenemos conocimiento de experiencias docentes de implantación del método aquí propuesto.

La principal desventaja de este enfoque es que no resulta fácil para los alumnos contrastar los datos experimentales obtenidos en el laboratorio con los resultados teóricos presentados en clase. Se produce así un salto entre ambos niveles que influye negativamente en la formación del alumno.

Si bien suele ser fácil para ellos comprobar cualitativamente que dos algoritmos tienen complejidades diferentes, no lo es comparar dos algoritmos de similar complejidad ni comprobar el impacto de ciertas mejoras como la elección del umbral óptimo en algoritmos de «divide y vencerás».

No obstante, realizamos una experiencia piloto con la ocasión de un curso de verano y las encuestas establecieron un grado elevado de satisfacción por parte de los alumnos. Entre las ventajas apreciadas al utilizar análisis híbrido cabe destacar que:

- Permite plantear prácticas más realistas a los alumnos.
- Facilita la comparación de algoritmos de similar complejidad.
- Permite estimar la constante oculta en la notación asintótica.
- Permite realizar predicciones cuantitativas sobre la evolución temporal.

En concreto, hemos observado que el alumno queda muy satisfecho cuando se le suministra un algoritmo desconocido y es capaz por sus propios medios de establecer un modelo cuantitativo de su complejidad temporal en el laboratorio.

7. Conclusiones

Hemos diseñado un entorno de trabajo adecuado a la realización de prácticas de análisis híbrido en el laboratorio. Este entorno está construido a partir de herramientas potentes y estables de software libre, lo que reduce los costes de implantación y facilita su uso al alumno.

Se ha ilustrado paso a paso el desarrollo con dichas herramientas de una práctica de laboratorio de estas características en la que se comparan diversos algoritmos de ordenación y se han expuesto las ventajas que aporta la técnica de análisis híbrido frente a un análisis meramente empírico.

Pretendemos implantar esta propuesta en el próximo curso, en las asignaturas *análisis y diseño de algoritmos I y II* de los nuevos planes de estudio de Informática en nuestra universidad. Estas asignaturas se impartirán en segundo curso, durante el primer y segundo cuatrimestres, respectivamente.

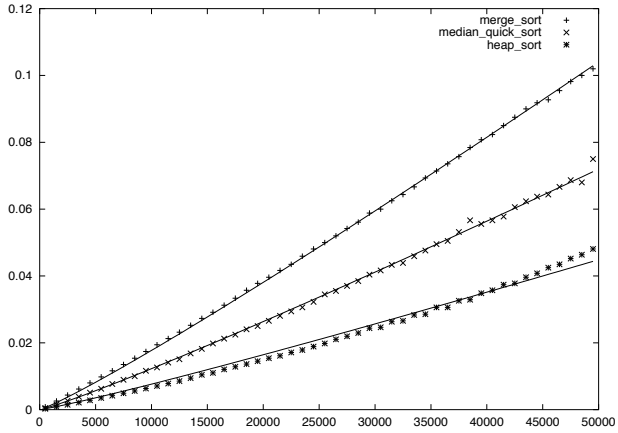


Figura 1: Resultados del análisis híbrido de los tres algoritmos

Referencias

[1] Joint Task Force on Computing Curricula. IEEE/ACM. *Computing Curricula 1991*. ACM Press and IEEE Computer Society Press (1991)

[2] Joint Task Force on Computing Curricula. IEEE/ACM. *Computing Curricula 2001*. ACM Press and IEEE Computer Society Press (2001)

[3] Austern, Matthew H. *Generic Programming and the STL: Using and Extending the C++ Standard Template Library*. Addison-Wesley (1998)

[4] Brassard, Gilles & Bratley, Paul. *Fundamentos de Algoritmia*. Prentice-Hall (1997)

[5] ISO/IEC 14882:1998. *Programming Languages – C++*. (1998)

[6] Jain, Raj. *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation and Modeling*. Wiley (1991)

[7] Levenberg, Kenneth. *A Method for the Solution of Certain Non-linear Problems in Least-Squares*. Quarterly of Applied Mathematics **2**(2) (1944)

[8] Marquardt, Donald W. *An Algorithm for the Least-Squares Estimation of Nonlinear Parameters*. SIAM Journal of Applied Mathematics **11**(2) (1963)

[9] Press, William H.; Teukolsky, Saul A.; Vetterling, William T. & Flannery, Brian P. *Numerical Recipes in C++: The Art of Scientific Computing*. Cambridge University Press. 2ª ed. (2002)

[10] Stroustrup, Bjarne. *The C++ Programming Language. Special Edition*. Addison-Wesley (2000)

Robótica e informática industrial

Docencia de la Planificación y el Desarrollo de un Proyecto de Informática Industrial

Lorenzo Moreno, Evelio González, Jonay Toledo, Leopoldo Acosta, Alberto Hamilton, J. Albino Méndez, Sergio Hernández, Marta Sigut, Nicolás Marichal, Santiago Torres
Grupo de Computadoras y Control
Departamento de Física Fundamental y Experimental, Electrónica y Sistemas
Universidad de La Laguna
28307 La Laguna, S/C Tenerife
e-mail: {lorenzo, evelio, leo, marta, jonay, nico, alberto, jamp, sergio, santiago}@cyc.dfis.ull.es

Jesús F. Montañez
Alumno de la Universidad de La Laguna y Becario de TOTALBAR, S.L.
email: jesusfm@cyc.dfis.ull.es

Resumen

Este trabajo elabora un plan de docencia de la planificación y desarrollo de un proyecto de Informática Industrial. Se basa en un proyecto real de colaboración de la ULL con una empresa del entorno. Dicho plan consiste en guíar al alumno a través de la mayor cantidad posible de fases: planificación, obtención de la placa de circuito impreso,... Aunque a fecha de redacción de esta ponencia, los alumnos no han pasado por todos los módulos propuestos, los resultados en cuanto a interés mostrado y conocimientos adquiridos es positivo. Animamos a emplear otros proyectos similares con idénticos fines docentes.

1. Introducción

El presente trabajo constituye un proyecto de docencia que pretendemos realizar en la asignatura de Informática Industrial y que pretende mostrar a los alumnos todos los aspectos que conlleva la realización de un proyecto comercial en esta disciplina: electrónica de potencia, electrónica digital, fabricación de circuitos impresos, verificación y validación del primer prototipo y optimización del producto final. No obstante, se planea extender estos contenidos a asignaturas de Diseño de Sistemas Digitales, Proyecto o Informática Industrial en los estudios de Ingeniería Eléctrica, Ingeniería Técnica Industrial e Ingeniería en Informática Especialidad de Sistemas, pudiendo entonces elaborar una distribución en horas por módulo según la cantidad de créditos a cubrir.

Para ello, nos basamos en un proyecto real de un desarrollo industrial realizado por nuestro departamento con una empresa del entorno. Hemos elegido este caso por:

- lo completo del proyecto que abarca todos los aspectos señalados anteriormente
- la clara diferenciación de las fases del proyecto, lo cual lo dota de un alto valor pedagógico.
- los abundantes recursos disponibles (documentación, prototipos iniciales y mejorados, producto final) que se derivan de un proyecto de estas características.
- nuestra experiencia de que los alumnos suelen mostrar un mayor interés cuando analizan situaciones y dispositivos de la "vida real"

A pesar de lo aparentemente concreto de esta aplicación, pensamos que es fácilmente extensible a otros proyectos reales que pueden ser aprovechados para ser mostrados a los alumnos, incluso en la docencia de otras áreas de la Informática.

No pretendemos por tanto, describir de forma exhaustiva este proyecto particular, sino guíar a los alumnos a través de la mayor cantidad posible de aspectos que surgen en un desarrollo comercial. Tampoco perseguimos que el alumno reproduzca el producto final (lo cual sería además imposible con la limitación temporal de la asignatura) sino que trabaje y asimile la mayor parte posible de las fases de un proyecto.

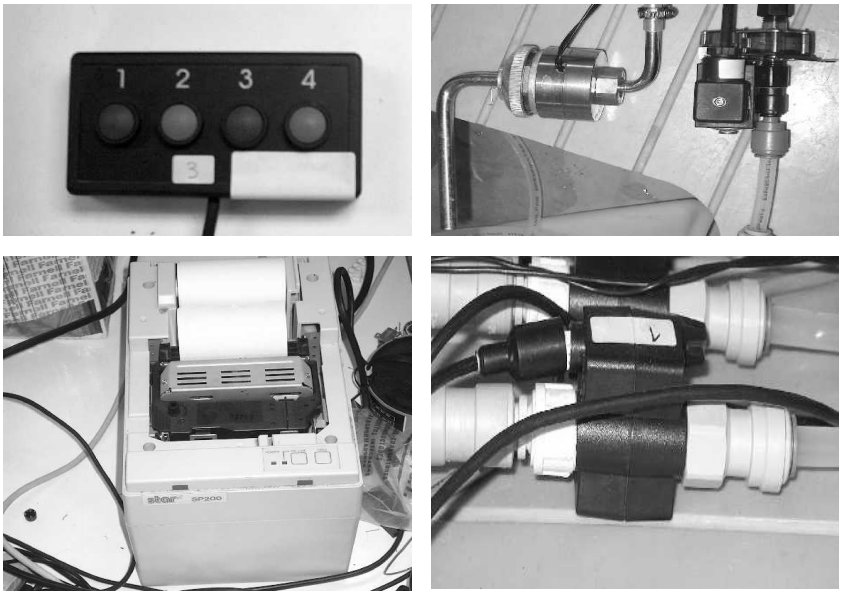


Figura 1. Elementos del proyecto presentado: botonera, electroválvulas, impresora y caudalímetro

El resto de la ponencia se estructura del siguiente modo. En primer lugar describiremos las especificaciones del proyecto, para posteriormente detallar las fases del mismo y cómo se reflejan en su docencia. Concluiremos esta ponencia evaluando el impacto observado en los alumnos de nuestra asignatura.

2. Descripción del proyecto

El proyecto que queremos desarrollar con los alumnos consiste en lo siguiente:

Se desea gobernar automáticamente el proceso de llenado de líquido de recipientes de diferente capacidad seleccionados mediante una botonera (figura 1 arriba izq.), activando una electroválvula (figura 1 arriba der.). El sistema debe asímismo poder envíar a una impresora (figura 1 abajo izq.) información del número de recipientes llenados durante el tiempo de funcionamiento.

Las cantidades de líquido a servir deben ser fijadas previamente en un proceso de calibración, ya sea por tiempo de apertura de la electroválvula o por caudal (método más exacto). La información de este caudal es sumistrado por un caudalímetro (figura 1 abajo der.) que suministra un tren de pulsos proporcional a la cantidad de líquido que circula por unas mangueras.

Finalmente, la información tanto de la calibración de las cantidades a servir como de la cantidad de recipientes llenados deben conservarse para la próxima vez que se encienda el dispositivo.

Podemos por tanto, listar las especificaciones de este proyecto del siguiente modo:

- Características Hardware
  - Modelo de electroválvula impuesto por la empresa
  - Modelo de caudalímetro impuesto por la empresa

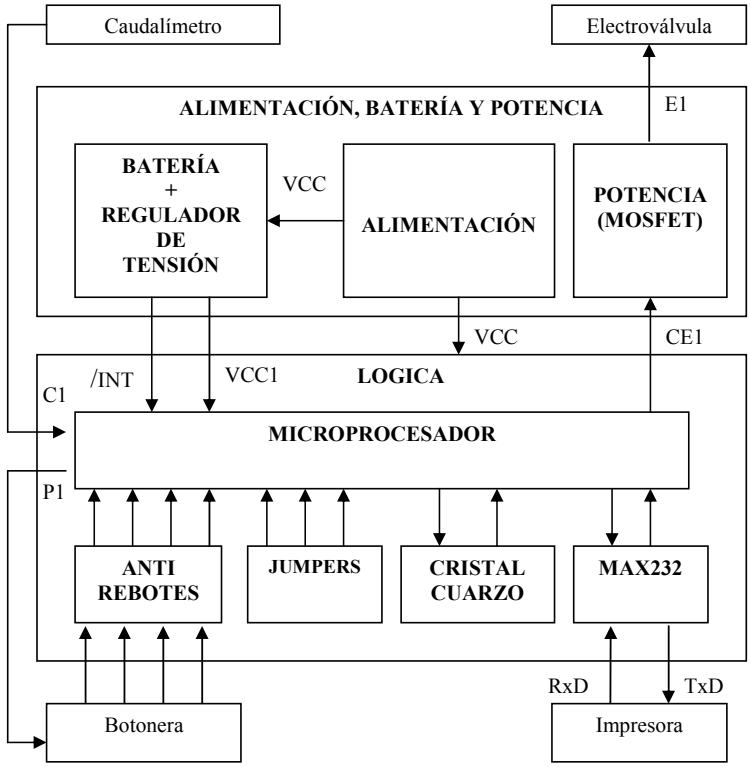


Figura 2. Esquema de bloques hardware del montaje, integrando los componentes suministrados por la empresa con dos placas que separan la parte de lógica de las de alimentación y potencia para activar la elevtroválvula. Se suministra al alumno para su estudio

- Microcontrolador a emplear
- Sistema de alimentación: batería para situaciones de power-fail
- Modelo de botonera impuesto por la empresa
- Limitación del tamaño del producto final

- Características Software
  - Modos servir por tiempo/servir por caudal
  - Modos calibración/operación/informe
  - Grabación de la información
- Tiempo de realización del proyecto
- Coste del producto

Como se puede observar en estas especificaciones, el sistema electrónico a diseñar contempla todos los aspectos de la electrónica que el alumno ha estudiado en anteriores asignaturas: electrónica digital, electrónica de potencia, alimentación y finalmente programación en lenguaje ensamblador para microcontroladores.

3. Planificación y Diseño del Primer Prototipo

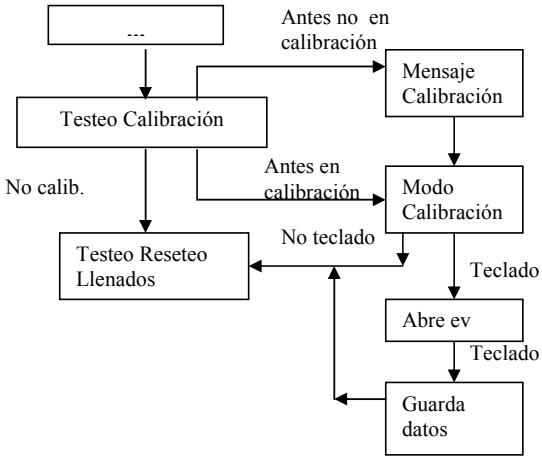


Figura 3. Detalle del diagrama de flujo del software del dispositivo, referido a la calibración del sistema. En el bucle principal se testea si el sistema se encuentra en modo de calibración. Si lo está y anteriormente no se encontraba en ese modo, lanza un mensaje a la impresora indicando que se acaba de entrar en ese modo, del que sale si no se detecta pulsación en las botoneras, pasando a la siguiente comprobación de modo (el de reseteo) . Al detectar una pulsación, el sistema abre la electroválvula y espera a que se pulse otra vez el botón, lo que indica que se ha llenado el recipiente con la cantidad de líquido deseada, almacenando el dato. Con este tipo de diagramas, los alumnos deben programar rutinas clave como la gestión de los pulsos o el almacenamiento de los datos.

Tras la descripción de las especificaciones del proyecto a realizar, los alumnos pasan al estudio de los apectos relacionados con la planificación del proyecto.

En primer lugar se les remarca la importancia de un cuidadoso proceso de planificación lo más realista posible, incluyendo todos los aspectos involucrados en el diseño del prototipo. Los alumnos deben saber extraer de este proceso de planificación las tareas a desarrollar y en qué plazos de tiempo, secuenciándolas a través de un diagrama de Gantt [2], poniendo de manifiesto las tareas que resultan ser críticas en la realización del proyecto. Existen numerosas aplicaciones para la elaboración de estos diagramas siendo la más empleada el Microsoft Project. No obstante ésta no es la única. Una simple búsqueda por Internet nos permite acceder a otras herramientas que se ajusten más a nuestra forma de enseñar estos aspectos. Cuando los alumnos, divididos en grupos de 2 ó 3, han realizado sus correspondientes diagramas, se hace una puesta en común, que permite enriquecer esta fase. Así los alumnos se dan cuenta de aspectos que habían pasado por alto o de suposiciones realizadas poco realistas.

Una vez realizado este proceso de planificación, empezaremos con el diseño del prototipo inicial. Dividiremos este proceso en las siguientes fases:

3.1. Elección de dispositivos

En esta fase se procede a la elección del procesador que gobernará el sistema (microcontrolador, PLC o procesador convencional) y la del resto de componentes: transformador, elementos de potencia, fuentes

conmutadas, módulos de transmisión serie, batería, rectificadores, relés,...

Respecto a la primera elección, se presentan a los alumnos las ventajas y desventajas (precio, funcionalidades, facilidad de programación) de cada una de los posibles procesadores. Tras ello se justifica la elección de un microprocesador de la familia ATMEL.

Para la segunda, cada grupo se encargará de la elección de los componentes de una parte del sistema. Estas partes son escogidas de modo cuidadoso para que no resulten excesivamente complicadas para el alumno y no tengan relación directa con la elección de los componentes de otras partes. Cada grupo dispondrá para esta elección de abundante documentación (catálogos y hojas técnicas de diversos componentes) que faciliten su labor. Una vez finalizado este estudio, el profesor de la asignatura lo evaluará.

3.2. Obtención del diagrama de bloques

La figura 2 presenta un diagrama de bloques del sistema. Dicho diagrama incluye cada uno de los elementos que debe ser diseñado en el prototipo y es suministrado a los alumnos para su estudio, lo cual permitirá visualizar el proyecto y ayudará en fases posteriores. En este caso, para el prototipo inicial se ha optado por dos placas que separen la parte de lógica de la de potencia y alimentación, para evitar en lo posible todo tipo de interferencias.

3.3. Diseño del circuito impreso

Se emplea un paquete de software como ORCAD u otro similar para el diseño del plano esquemático con el SDT (Schematic Design Tool). A partir del fichero esquemático, se genera la placa de circuito impreso o PCB (Printed Circuit Board) con el correspondiente módulo del programa (PCB layout en el caso del ORCAD), para posteriormente definir los ficheros GERBER [3] (formato ampliamente empleado y que contiene información sobre la anchura de las pistas y los puntos de inicio y final de cada una).

A nivel docente, en esta fase se pretenden que los alumnos trabajen con estos programas e investiguen todas las posibilidades que ofrecen. Como la placa global tiene demasiados componentes, se divide en módulos interconectados más pequeños, cuyo esquemático se suministra a cada grupo. Los alumnos obtienen a continuación el PCB y GERBER de cada módulo.

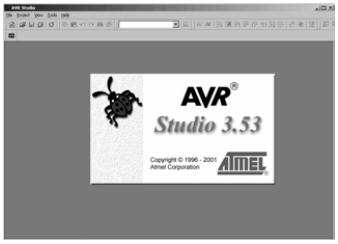


Figura 4. Entorno de programación AVR Studio

3.4. Diseño Software

La fase final del diseño del apartado anterior se puede simultanear con el diseño software: diagrama de bloques, código y una primera comprobación del mismo mediante un simulador, si existe, del microcontrolador o autómata elegido. En nuestro caso, ATMEL [1] proporciona un completo entorno de programación para una amplia gama de microprocesadores: el AVR Studio (Figura 4). Este entorno es gratuito y se complementa con una abundante documentación sobre el mismo y sobre los microprocesadores.

A nuestros alumnos se les suministra el código fuente del dispositivo en ensamblador y el diagrama de flujo del mismo (detalle en Figura 3), a falta de unas rutinas principales (por ejemplo la gestión de los pulsos envíados por el caudalímetro o la detección de la caída del suministro eléctrico). La cantidad de trabajo a realizar dependerá del tiempo disponible en la asignatura.

3.5. Obtención de la placa de circuito impreso

Una vez que se dispone de la placa de circuito impreso, la cual ha sido realizada en nuestro laboratorio, se procede al testeo de la misma, detectando y corrigiendo todos los errores encontrados. En un prototipo como el nuestro, que dispone tanto de elementos de potencia

(electrónica analógica) como elementos digitales, los errores más comunes detectados son:

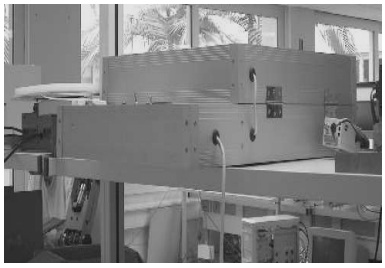


Figura 5. Diversos métodos de obtención de placas de circuito impreso: insoladora (izq.) y fresadora (der.). En el laboratorio se muestra a los alumnos brevemente ambos métodos

- Errores producidos por un diseño térmico deficiente. Se puede disipar demasiado calor en determinados componentes, lo que reduce su vida media
- Errores intermitentes: falsos contactos debidos a una mala soldadura, problemas de ruido producidos en el entorno que afectan al sistema, problemas de tierra, etc...
- Falta de optimización en el prototipo inicial: exceso de cableado, exceso de componentes, elementos redundantes, etc...

En relación con los alumnos, inicialmente se les comenta brevemente los procesos de elaboración del circuito impreso: el artesanal mediante insoladora y el de fresado. Esto se realiza en el laboratorio, mostrándoles el instrumental correspondiente que puede verse en la Figura 5.

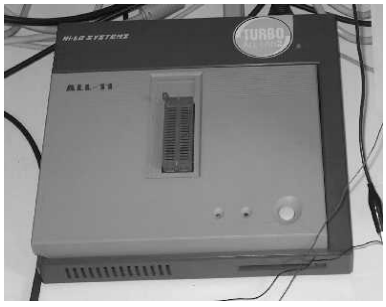


Figura 6. Programador ALL-11 para el microcontrolador.

Al contar con los ficheros originales del proyecto, podemos replicar placas del prototipo inicial que son testeadas por los alumnos (una pequeña parte por cada grupo), incidiendo en los diferentes clases de errores posibles.

3.6. Finalización del primer prototipo

Disponiendo ya de la placa depurada, se integra en ella la primera versión de software depurada con simulador. Se inicia entonces otro proceso de testeo y depuración de errores, tanto hardware como software, hasta finalizar un primer prototipo.

Al final de esta fase, el alumno se habrá familiarizado con la programación y depuración de código ensamblador para microcontroladores. Cuentan para ello con un programador ALL-11 (Figura 6), que permite programar un amplio conjunto de microprocesadores.

3.7. Presentación del prototipo y proceso de mejora

Este primer prototipo cumple con las especificaciones iniciales del proyecto, pero no supone el fin del mismo. No es extraño, más bien todo lo contrario, que una vez que se presente el sistema a la empresa, ésta introduzca nuevas especificaciones. Éstas pueden derivarse de razones tan variadas como por ejemplo de un nuevo estudio de mercado, nuevas funcionalidades que se quiere asignar al dispositivo o de componentes complicados de

conseguir en el lugar de comercialización (recordamos que se desea comercializar el producto en otros países).

En paralelo, pueden localizarse posibles mejoras en el diseño (por ejemplo un diseño térmico más eficiente) y que a la vez faciliten el replicado del producto. En este sentido cabe citar:

- el prototipo inicial normalmente presenta conexiones adicionales (derivadas del testeo y verificación) que complican el replicado. Por tanto se realizan nuevas placas con dichas modificaciones.
- mejorar el conexionado, sustituyendo en lo posible las soldaduras por conectores modulares
- englobar el sistema en una única placa, ahorrándose de este modo las conexiones entre placas

En cuanto a nivel docente se plantean dos actividades:

- Se plantean algunas de las especificaciones reales sugeridas por la empresa en esta fase y se solicita a los alumnos que determinen qué modificaciones implican en el dispositivo.
- Se presentan en el laboratorio el prototipo inicial y diversos prototipos evolucionados, donde los alumnos puedan comprobar fácilmente las mejoras introducidas. En la Figura 7, pueden verse tres prototipos. En el inicial (arriba), un estudio térmico inicial no optimizado conduce a un disipador de grandes dimensiones. Esto fue reparado en el segundo prototipo (medio), pero las excesivas conexiones con cables dificultan el replicado. Finalmente el prototipo final (abajo) ha unificado el sistema con una única placa con conectores modulares.

3.8. Generación de la Documentación

Tras diversas realimentaciones desde el empresario en el punto anterior, el producto se encuentra listo para su comercialización. Pero antes debe ser complementado con una documentación imprescindible: el manual de usuario y el manual de instalador.

El plan de docencia termina proporcionándole a los alumnos copia de ambos manuales, incidiendo en algunos aspectos de redacción (detalles de las figuras, orden, completitud...)

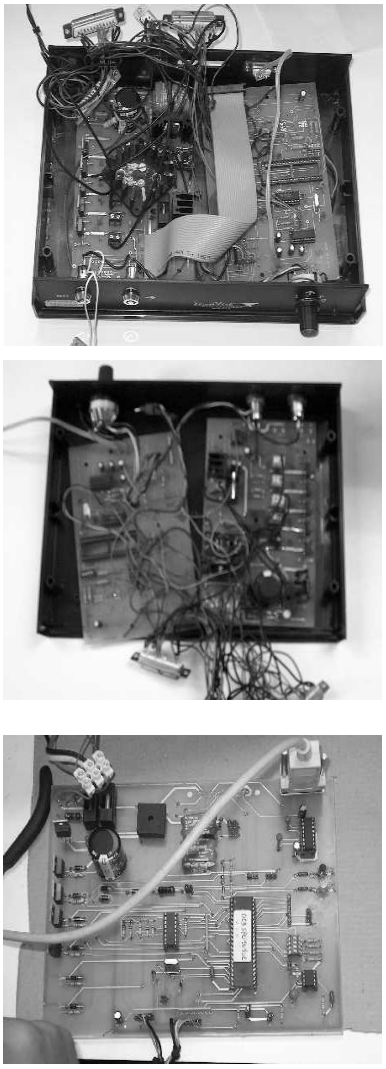


Figura 7. Evolución de los prototipos

4. Evaluación del impacto de la docencia

A fecha de redacción de esta ponencia no se ha podido valorar de un modo global este proyecto de docencia. Por razones de tiempo, los alumnos solamente han realizado algunos de los módulos descritos en esta ponencia, como la programación con simulador y la depuración código de placa, tras suministrarle todos los diagramas descritos aquí. No obstante, se planea extender estos contenidos a asignaturas de Diseño de Sistemas Digitales, Proyecto o Informática Industrial en los estudios de Ingeniería Eléctrica, Ingeniería Técnica Industrial e Ingeniería en Informática Especialidad de Sistemas, pudiendo entonces elaborar una distribución en horas por módulo según la cantidad de créditos a cubrir.

Sin embargo, los alumnos han mostrado un notable interés por este proyecto y por las prácticas que han realizado. Agradecen trabajar siguiendo los pasos de diseño y fabricación de dispositivos que se están comercializando con bastante éxito, e incluso llegan a pedir más documentación o analizar más la placa.

Desde los autores consideramos altamente instructivo esta propuesta pues los alumnos tienen una visión global de lo que supone un proyecto de colaboración con una empresa, englobando numerosas áreas como la electrónica de potencia, electrónica digital, fabricación de circuitos impresos, elaboración de documentación,...

5. Conclusiones

Se ha propuesto un proyecto de docencia en asignaturas de Informática Industrial. El objetivo de este plan es guíar a los alumnos por las diferentes fases de un proyecto de colaboración con empresa. Este proyecto está basado en un proyecto real de la ULL con una empresa del entorno.

Los alumnos van pasando por diversos módulos que reflejan las diferentes fases: planificación de tareas, diseño de prototipo, elección de los componentes, diseño del circuito impreso, desarrollo del software, obtención de la placa, mejoras en el prototipo,...

Aunque no ha sido puesto en marcha todos los módulos, la opinión entre los alumnos es positiva. Por otra parte, los autores consideran este proyecto altamente instructivo pues los alumnos tienen una visión global de lo que supone un proyecto de colaboración con una empresa englobando un gran número de disciplinas.

Desde aquí se anima al lector a aprovechar proyectos similares para fines docentes.

Agradecimientos

Los autores desean mostrar su agradecimiento a los componentes del servicio técnico del Departamento de Física Fundamental y Experimental, Electrónica y Sistemas de la ULL, D. Roberto Betancor Bonilla, D. Manuel Fernández Vera y D. Agustín Padrón Name por sus horas de trabajo en este proyecto. Asimismo queremos agradecer a D. José León González, representante de la empresa TOTALBAR S.L., originaria del proyecto, por su confianza en nosotros para sacar adelante este producto.

Referencias

[1] ATMEL. Página principal: http://www.atmel.com

[2] Cos Castillo, Manuel de *Ingeniería de Proyectos*. Madrid, 1993.

[3] Page J., Delgado A., Blanco C. *Banco de CAD-CAM del laboratorio de circuitos de alta frecuencia de la ETSIT-UPM.* Actas URSI'96. http://www.etc.upm.es/alcaf.htm

Propuesta para la Integración de Prácticas de Laboratorio en Intensificaciones de la titulación de ITIS[1]

Vicent Lorente, Sílvia Terrasa, Salvador Petit y Alfons Crespo
Departamento de Informática de sistemas y computadores
Universidad Politécnica de Valencia
{vlorente, sterrasa, spetit, alfons}@disca.upv.es

Resumen

Los cambios en los planes de estudios de las ingenierías técnicas hacen que el profesorado en general se plantee cambios en la forma de impartir la docencia. En el presente artículo se propone la utilización de una plataforma común para la realización de actividades prácticas para asignaturas pertenecientes a una misma intensificación. Esto permite, por una parte, aunar esfuerzos a la hora de realizar la puesta en marcha de las prácticas, y, por otra, ofrecer al alumno una visión global a un problema complejo.

1. Introducción

En la actualidad, la Universidad Politécnica de Valencia está efectuando, desde el pasado curso 2001-2002, cambios en los planes de estudio de algunas titulaciones técnicas. Dos de estas titulaciones son las de Ingeniero Técnico en Informática de Gestión (ITIG) y la de Ingeniero Técnico en Informática de Sistemas (ITIS). Uno de los cambios más interesantes, desde el punto de vista de la formación del alumnado, es la aparición del concepto de intensificación. Con éste término, lo que se intenta es ofrecer al alumno la posibilidad de realizar una especialización durante el último curso que pueda acreditarse de alguna forma. Para ello lo que se intenta es reunir asignaturas optativas con una temática común. La idea es que cuando un alumno decida elegir una intensificación, deberá cursar obligatoriamente las asignaturas centrales de la intensificación (normalmente tres asignaturas) y luego escoger dos entre un conjunto de entre 6 a 8 asignaturas de temática relacionada para cubrir un mínimo de créditos de la intensificación y que esta se pueda acreditar.

Esta nueva forma de concebir las materias optativas hace que muchos profesores intenten aunar esfuerzos para efectuar su docencia de la forma más adecuada. Hasta ahora, las asignaturas optativas eran en cierta forma asignaturas independientes, en las que sus profesores podían cambiar contenidos o reestructurarlos sin que ello tuviera mayores consecuencias. Sin embargo, a partir de ahora se deberá actuar con más cautela, y se tendrán que revisar los contenidos de las asignaturas optativas de una determinada intensificación para asegurar que ni se repiten conceptos, ni se queda ningún concepto por impartir.

A parte de lo que es meramente teoría, uno de los objetivos más interesantes es que las sesiones prácticas sean lo más completas posibles, intentando cubrir todos los objetivos prácticos de la intensificación. Para poder llevar a cabo este objetivo de forma adecuada es necesario tener en cuenta las limitaciones temporales que plantean estas asignaturas, ya que, en el mejor de los casos, sólo disponen de dos horas de laboratorio a la semana, y la mayoría sólo disponen de dos horas de laboratorio cada quince días. Estas limitaciones hace que, en muchas ocasiones las prácticas no cubran todos los aspectos necesarios. Por otra parte, también suelen existir limitaciones de

[1] Ingeniero Técnico en Informática de Sistemas

espacio y de dinero a la hora de adquirir un material de prácticas un poco más sofisticado.

El presente artículo propone un método de diseño de las sesiones prácticas para que las limitaciones antes expuestas se vean aliviadas. La idea fundamental es la de utilizar el mismo material de soporte de prácticas en varias de las asignaturas de la intensificación. De esta manera, se pueden unir los esfuerzos de los profesores implicados para, por una parte, conseguir un material de apoyo más sofisticado y robusto, y, por otra parte, montar prácticas relacionadas entre si que cubran todos los aspectos interesantes de la intensificación en cuestión.

La propuesta está orientada a una intensificación en concreto, la intensificación de Informática Industrial de la titulación de ITIS. Se proponen, a modo de ejemplo, unas prácticas combinadas para las asignaturas de Sistemas de tiempo real, Automática Industrial y Control y la de Sistemas Robotizados, todas ellas pertenecientes a esta intensificación.

El resto del capítulo se organiza como sigue: en la sección 2 se describe el marco de las intensificaciones de la titulación de ITIS, y en concreto de la intensificación de informática industrial. La sección 3 está dedicada a describir el material de soporte que se va a utilizar en las sesiones prácticas. La sección 4 se plantearan las sesiones prácticas de las distintas asignaturas de la intensificación. Finalmente en la sección 5 se proporcionaran algunas conclusiones.

2. Intensificaciones de la titulación de ITIS.

En los planes de estudio que se están poniendo en práctica en la Escuela Superior de Informática Aplicada de la UPV[2] , el número de créditos optativos que tiene que cursar un alumno para la obtención del título de Ingeniero Técnico en Informática de Sistemas es de 34,5. La totalidad de dichos créditos se encuadran en el tercer curso de acuerdo al plan de estudios. Ello permite ofrecer un marco de intensificaciones destinado a imprimir un perfil más profesional a los futuros titulados.

Las materias optativas están agrupadas por intensificaciones. Cada intensificación está constituida por un Núcleo de Intensificación (18 créditos) y un conjunto de materias optativas afines.

Para obtener el título de ITIS un alumno deberá cursar obligatoriamente al menos un Núcleo de Intensificación perteneciente a una de las siguientes intensificaciones:

- Administración de Sistemas y Redes
- Informática Industrial.
- Ingeniería de computadores.
- Multimedia.
- Tecnologías y Servicios Web

A continuación se pasa a describir más en profundidad la intensificación de Informática Industrial, objeto del presente artículo.

2.1. Intensificación de Informática Industrial.

La intensificación de Informática Industrial tiene como objetivo principal el formar a profesionales con sólidos conocimientos de las técnicas, dispositivos y herramientas propias del ámbito industrial que le capaciten para la especificación, diseño, montaje, depuración y mantenimiento de sistemas informáticos de control y su integración en el ámbito de las redes industriales de área local, así como el desarrollo de aplicaciones de tiempo real y de software en general para el control de procesos industriales a través de computador.

Como se ha comentado anteriormente, todo alumno que desee cursar la intensificación de Informática Industrial, ha de cursar obligatoriamente las asignaturas correspondientes al núcleo de la intensificación y al menos 2 asignaturas de las materias complementarias de la misma Las asignaturas correspondientes a esta intensificación son:

Núcleo de la intensificación:

- Automática Industrial y Control.
- Sistemas de Tiempo Real.

[2] Universidad Politécnica de Valencia

- Sistemas de Entrada / Salida.

Materias complementarias de la intensificación:

- Análisis aplicado.
- CAD / CAM
- Configuración, administración e interconexión de redes de área local.
- Control estadístico de calidad.
- Diseño de sistemas lógicos.
- Fundamentos físicos de la robótica.
- Gestión y mantenimiento de empresas industriales.
- Periféricos e interfaces.
- Sistemas robotizados.
- Técnicas de inspección y de mantenimiento.

Como se comentó en la introducción la propuesta que se realiza en este artículo está relacionada con tres de las asignaturas de esta intensificación, dos de ellas pertenecientes al núcleo (Sistemas de tiempo real y Automática industrial y control), y la tercera perteneciente a las materias complementarias (Sistemas robotizados). En las siguientes secciones se describe las prácticas propuestas.

3. Diseño y montaje de los prototipos hardware.

A la hora de abordar el reto de realizar prácticas conjuntas, lo primero que se debía tener claro era el material que se iba a utilizar. Lo que se intentaba era disponer de un material común sobre el cual realizar distintas actividades. Debido a la temática de las asignaturas se optó por el montaje de la figura 1. Como se puede observar, el montaje consta de dos brazos robots, de cuatro grados de libertad y de una cinta transportadora.

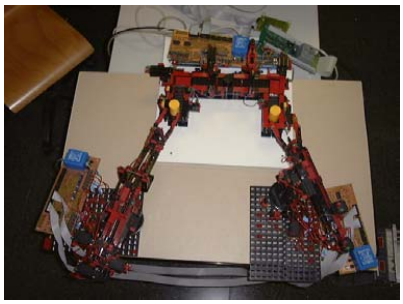


Figura 1. Montaje propuesto para las prácticas.

Como se ha comentado, los brazos robot disponen de cuatro articulaciones, cada una de ellas conectadas a un motor. También dispone de los finales de carrera para cada articulación (figura 2). Todos la entradas y las salidas del brazo robot son digitales.

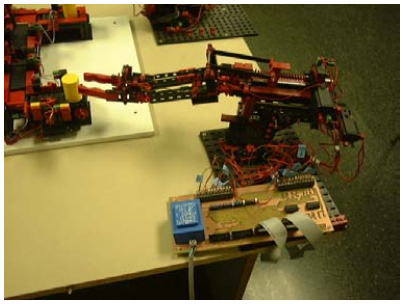


Figura 2. Brazo Robot.

La cinta transportadora (figura 3), por su parte dispone de sensores de posición así como de mecanismos para desplazar los objetos que se transporten. Al igual que en los robots todas las entradas y salidas son digitales.

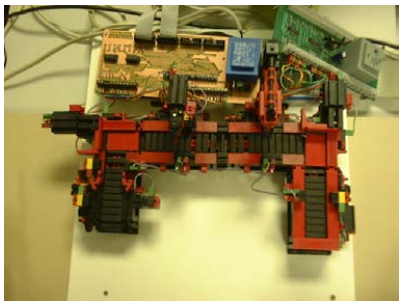


Figura 3. Cinta transportadora.

Para cada uno de los elementos (los dos brazos robots y la cinta transportadora) se han diseñado y realizado una serie de prototipos *hardware* para hacer posible la conexión a una tarjeta de adquisición de datos (PCLAB), de forma que sea posible su control desde un ordenador.

Este montaje nos permite disponer de un abanico bastante amplio de actividades prácticas. Por una parte, para la asignatura de sistemas de tiempo real, el simple hecho de disponer de un proceso físico a controlar ya proporciona el marco adecuado para la realización de sus prácticas, tal y como se veía en [1]. Además, con este montaje, disponemos de tres procesos distintos que pueden cooperar en un momento dado.

Respecto a la asignatura de Automática industrial y control, se pueden realizar muchas actividades relacionadas tanto con la fase de diseño de control, diagramas de estado, etc, hasta la implementación de algoritmos de control para controlar el proceso de producción.

Finalmente, en la asignatura de Sistemas Robotizados el simple hecho de disponer de brazos robots ya supone un marco ideal para realizar las actividades prácticas, ya que se puede desde realizar el modelado de los brazos robots hasta resolver el problema de la cinemática directa o inversa, y luego comprobar sobre el propio robot le funcionamiento de las ecuaciones calculadas.

4. Planteamiento de las sesiones prácticas según la asignatura.

Una vez descrito el marco común de las actividades prácticas, en este apartado se va a describir las actividades prácticas para cada una de las asignaturas.

4.1. Sistemas de tiempo real.

El descriptor de la asignatura de sistemas de tiempo real dice que los objetivos de la misma son:

- Saber las características de los sistemas de tiempo real (STR) y la de los sistemas operativos de tiempo real.
- Conocer los requisitos de los lenguajes para STR.
- Ser capaz de programar en Ada.
- Conocer tanto las metodologías de diseño de los STR como conceptos sobre su planificación.

Como se comentaba en [1], la evaluación de la asignatura se realiza mediante un trabajo de asignatura que consiste en el control de un proceso físico real. A este respecto, el montaje descrito en la sección anterior es un buen ejemplo de proceso a controlar.

Para la elaboración tanto del trabajo como de las actividades prácticas, se dispone de un entorno cruzado de desarrollo de aplicaciones empotradas, proporcionado por el sistema operativo de tiempo real MarteOS desarrollado por el departamento de Electrónica y Computadores de la Universidad de Cantabria [3]. Este entorno nos ofrece la posibilidad de desarrollar código en un entorno Linux y luego ejecutarlo en un sistema operativo de tiempo real (para más información sobre este punto consultar [2]).

Las actividades prácticas se deben organizar de forma que la dificultad vaya incrementándose de forma paulatina. De forma que se intente primero actuar de forma independiente sobre los distintos elementos del montaje. Una vez claro el comportamiento de los elementos por separado ya se plantea en una última práctica la el control coordinado de toda la plataforma.

En estas actividades se debe hacer especial hincapié en los aspectos propios de los sistemas de tiempo real, utilizándose para tal efecto un sistema operativo de tiempo real (MarteOS) y un lenguaje de programación también de tiempo real (Ada95).

4.2. Automática industrial y control.

En los descriptores de la asignatura de Automática industrial y control (AIC) aparece como objetivos principales los de :

- Conocer los controladores lógicos programables.
- Saber realizar el modelado y análisis de sistemas continuos, discretos y muestreados
- Entender el bucle de control por computador.
- Saber diseñar de reguladores discretos.
- Ser capaz de realizar una implementación práctica.

Si nos fijamos en el montaje de la figura 1, bajo el punto de vista de esta asignatura, se puede ver el conjunto como una planta industrial en la cual se ofrece la visión de un proceso industrial completo. En esta planta se pueden distinguir tres elementos:

1. Un brazo robot destinado a alimentar el proceso con piezas,
2. Una cinta transportadora dedicada a transportar y manipular las piezas.
3. Un brazo robot destinado a ir apilando el producto final.

Con esta visión del montaje, las actividades prácticas se pueden orientar de forma que los alumnos realicen primero un modelado del sistema discreto. A partir del modelado, sus conocimientos del control en bucle cerrado, sería interesante que fueran diseñando y aplicando distintos tipos de reguladores con el fin de realizar el control de la planta.

4.3. Sistemas robotizados.

Finalmente los descriptores de la asignatura de Sistemas Robotizados (SR) incluyen los siguientes objetivos:

- Conocer los componentes y la estructura de los sistemas robotizados.
- Ser capaz de programar robots.
- Saber como realizar el modelado y control de robots.
- Tener nociones sobre sensorización en robótica.
- Saber realizar aplicaciones.

Como se puede deducir de los objetivos de la asignatura, ésta intenta dar una visión de lo que es un sistema robotizado, desde los aspectos más técnicos (componentes y estructura) hasta los aspectos de más alto nivel (modelado y control). A este respecto, el montaje propuesto es ideal ya que es en sí mismo un sistema robotizado.

Las actividades prácticas propuestas para esta asignatura incluyen por orden de dificultad:

1. Realizar un modelado de los brazos robot y de la cinta transportadora.
2. Realizar un control de forma independiente de cada uno de los elementos.
3. Finalmente realizar un control coordinado.

La diferencia fundamental entre los algoritmos de control de esta asignatura y los vistos en la asignatura de AIC, son básicamente que los de esta asignatura están centrados en los SR, mientras que los de la asignatura de AIC son mucho más generales.

5. Conclusiones

Una de las ventajas que tienen los cambios de estudios en las titulaciones de informática es que obligan en cierta manera al profesorado a estar al día en cuanto a la temática de sus asignaturas optativas, a aparte de introducir cambios en la propia estructura del plan de estudios.

Con esta motivación, y gracias a los nuevos planes de estudio de la titulación de ITIS de la

Universidad Politécnica de Valencia, ha surgido un nuevo concepto: el concepto de intensificación.

El presente artículo ha intentado ofrecer una propuesta sobre cómo mejorar la docencia, al menos en cuanto a actividades prácticas se refiere, aprovechado este nuevo concepto de intensificación. Básicamente lo que se intenta es, por una parte, mejorar los recursos utilizados en las actividades prácticas ya que el hecho de compartir los montajes en distintas asignaturas puede hacer que éstos sean un poco más costoso y, además, hay más profesores implicados en la puesta de marcha de los mismos. Por otra parte, el alumno es capaz de relacionar mejor los conocimientos adquiridos en distintas asignaturas, pudiendo obtener una visión global del problema, ya que, aunque cada asignatura se centra en un aspecto concreto, el hecho de realizar las actividades prácticas sobre la misma plataforma le permite asociar conceptos más fácilmente.

Referencias

[1] S. Terrasa, P. Balbastre, A. Crespo. "La importancia del uso de procesos físicos reales en la enseñanza universitaria en la ingeniería". JENUI'2000, pp.304-310

[2] S. Terrasa, P. Balbastre, A. Crespo. "Experiencia docente en el desarrollo de aplicaciones empotradas con MarteOS", JENUI'2002, pp.304-310

[3] MarteOS : http://marte.unican.es

[4] "Titulación de Ingeniero Técnico en Informática de sistemas (plan 2001)" B.O.E. nº 259 de 17/10/2001

Sistemas distribuidos y paralelos

Domótica y Edificios Inteligentes en la Universidad de Alicante

Jorge Azorín, Andrés Fuster, Francisco Maciá, Francisco J. Ferrández
Dpto. Tecnología Informática y Computación
Universidad de Alicante
Apdo. Correos 99. E-03080. Alicante
e-mail: {jazorin,fuster,pmacia,fjferran@dtic.ua.es}

Resumen

En este artículo se presenta el enfoque de la asignatura Domótica y Edificios Inteligentes impartida en las Ingenierías Informáticas de la Universidad de Alicante. La propuesta subraya el papel del Ingeniero en Informática en el campo de la domótica, con un planteamiento motivado por la evolución de este campo de aplicación tecnológica hacia la integración de servicios de comunicaciones y computación. Inicialmente, se exponen las motivaciones de la incorporación de la asignatura a los planes de estudio y los objetivos orientados a la aplicación en este campo del cuerpo de conocimiento de los estudios de informática. A continuación se propone el contenido teórico y práctico con un marcado carácter aplicado, finalizando con una propuesta de evaluación continua del alumno.

1. Introducción

La incorporación de la asignatura Domótica y Edificios inteligentes se sitúa en la reforma de los planes de estudio de Informática que se produce en el curso 2001/2002 en la Universidad de Alicante [1][2][3]. En los nuevos planes de estudio de Informática, la optatividad representa un porcentaje importante en el total de créditos que han de cursar los alumnos, concretamente un 20% para la Ingeniería Informática, 18,66% para la Ingeniería Técnica en Informática de Sistemas y 16% para la Ingeniería Técnica en Informática de Gestión. Dentro de este plan de estudios y concretamente para el área de conocimiento Arquitectura y Tecnología de Computadores, se contemplan asignaturas optativas concebidas con carácter complementario al cuerpo de conocimiento impartido en materias obligatorias y troncales (Sistemas Operativos, Arquitectura de Computadores, redes, etc) y otras con un marcado carácter aplicado. Domótica y Edificios Inteligentes es una asignatura optativa de primer y segundo ciclo para las tres carreras de informática y forma parte del enfoque aplicado de las materias ofertadas a los estudiantes de informática. Se trata de una asignatura cuatrimestral y consta de tres créditos de teoría y tres de prácticas.

La domótica o el concepto más amplio de edificios inteligentes es un campo de aplicación tecnológica que está irrumpiendo con fuerza desde hace relativamente muy poco tiempo y donde el Ingeniero en Informática tiene un campo de trabajo poco explotado en la actualidad. Se trata de un campo multidisciplinar, si bien, en el enfoque clásico se ha tratado como un problema de automatización. Estos hechos están presentes en el contexto de la universidad española (ver tabla 1), donde solamente dos universidades, la Universidad de Oviedo [4] y la Universidad Politécnica de Madrid [5], ofertan una asignatura específica para este campo, aunque con un planteamiento distinto del que nosotros queremos ofrecer.

El enfoque que nosotros planteamos está motivado principalmente por la transformación que este campo está experimentado: desde un enfoque de automatización no estructurado hacia una integración de servicios de procesamiento de información y comunicaciones en el marco de las IST (Information Society Thecnologies – Tecnologías de la Sociedad de la Información) [6]. En esta transformación, el Ingeniero en Informática tiene un papel fundamental. La asignatura pretende fomentar esta visión, marcando unos objetivos y contenidos que doten al alumno de conocimiento, habilidades y

Universidad	Asignatura	Créditos	Contenido
U. Oviedo	Domótica y Edificios Inteligentes (2002)	4.5 + 1.5	Nociones generales de electrotecnia. Estudio de componentes y sistemas para la gestión técnica de la edificación
U. Politécnica de Madrid	Domótica y Edificios Inteligentes (1996)	3 + 1.5	Generación y transporte. Red de baja tensión. Iluminación. Electricidad estática. Arquitecturas del sistema de control de un edificio. Sistema de climatización. Sistema de detección y protección contra incendios. Sistema de seguridad. Sistema de control de ascensores. Estudio de proyectos concretos de control de edificios.

Tabla 1. Domótica y Edificios Inteligentes en otras universidades.

actitudes dentro de este campo concreto. El conocimiento que se aporta al alumno, en algunos temas, no es novedoso sino fundamentalmente de aplicación de modelos y técnicas propias de la disciplina informática en el contexto del control inteligente de viviendas.

En los siguientes apartados mostraremos la motivación y los objetivos propuestos para la asignatura, expondremos los contenidos teóricos y prácticos, y por último, finalizaremos el artículo con las conclusiones que se han extraído de la experiencia docente.

2. Motivación y objetivos

La domótica es un campo de naturaleza multidisciplinar donde áreas como la electrónica, la automática, las comunicaciones y la computación aportan parte de su conocimiento. La aplicación de estos cuerpos de conocimiento tiene como objetivo proporcionar servicios en el entorno del hogar y de forma más genérica en la edificación. Este campo se entiende como el área de automatización de las actividades y los sistemas del hogar y de incorporación de inteligencia en los edificios.

El ámbito de la domótica[1] según su enfoque clásico se limita a la automatización puntual de servicios a pequeña escala, lo que lleva asociado una falta de estructura. Sin embargo, el ámbito de la domótica es mucho más amplio y más abierto que la automatización de edificios. El enfoque actual hace énfasis en la integración de servicios generales de procesamiento de información y de comunicaciones dentro del marco de las IST. Esta tecnología debe abordar la automatización de los sistemas del hogar actuales y soportar los servicios futuros relacionados con el trabajo, la salud, la educación, etc (e-work, e-health, e-learning,…)

Las características generales de los sistemas domóticos son:

- Las mismas de los sistemas generales de redes de computadores: estructuración, modularidad, transparencia (encapsulación), escalabilidad.
- Comprenden todos los niveles, desde la capa física hasta la interfaz de usuario.
- Constituyen sistemas informáticos y de comunicaciones multidisciplinares, con concurso de todas las áreas de la informática y de otras áreas.
- Los subsistemas, los dispositivos y los componentes son muy diversos y hay gran heterogeneidad en requerimientos, prestaciones y costes.

La gran evolución que han experimentado las IST, y concretamente la disciplina informática, ha repercutido en todos los ámbitos de la sociedad. Los sistemas informáticos presentan nuevas posibilidades a la hora de abordar cualquier tipo de problema. Estas posibilidades han marcado la evolución de los sistemas domóticos.

El nuevo planteamiento requiere la revisión y diseño de nuevos objetivos. El alumno debe tener conciencia de la evolución de esta tecnología y el papel que el Ingeniero en Informática desempeña. Debe conocer cuales son las tecnologías que existen actualmente y ser capaz de enfrentarse a la especificación y diseño de estos sistemas

[1] Según el texto literal del diccionario de la Real Academia Española, domótica se define como: (Del lat. *domus*, casa, y *informática*). 1. f. Conjunto de sistemas que automatizan las diferentes instalaciones de una vivienda.

domóticos. Concretamente, los objetivos generales de la asignatura Domótica y Edificios Inteligentes los detallamos a continuación:

- Ubicar el área domótica en el marco de las IST.
- Introducir los conceptos generales de la domótica y su ámbito de aplicación.
- Estudiar la metodología general de diseño de los sistemas domóticos.
- Conocer los subsistemas domóticos y los dispositivos de percepción y actuación.
- Estudiar los estándares actuales y las tecnologías de comunicación en edificación.
- Diseñar un sistema domótico basado en agentes.
- Plantear sistemas avanzados y líneas futuras de desarrollo

3. Contenidos teóricos

Con el fin de cubrir los objetivos que se han marcado en la sección anterior, el contenido teórico de la asignatura se desarrolla en tres bloques temáticos. A continuación describimos cada uno de los bloques y los temas que comprende.

Bloque I. Generalidades

Este bloque introduce al alumno en la domótica explicando qué es lo que se espera de esta tecnología y cuál es el papel que desempeña el Ingeniero en Informática en este contexto. También, se cubren los aspectos legales.

Tema 1. Ámbito de la domótica

Objetivos: Introducir las metas de la Sociedad de la Información, revisar las tecnologías integradas en la IST y ubicar la domótica en la IST.

Contenido: Conceptos. Ámbito. Características de los sistemas domóticos. Criterios de diseño.

Tema 2. Marco legal y competencial

Objetivos: Introducir los conceptos generales de la domótica y su ámbito de aplicación, revisar el marco legal de la domótica y ubicar la competencia del Ingeniero en Informática.

Contenido: Marco legal. Competencias. El Ingeniero en Informática y la domótica.

Bloque II. Tecnología domótica

El bloque II se presenta como el de mayor carga de la asignatura contando con más del 50% del total de créditos teóricos. Aquí se analizan los distintos servicios que se esperan de un sistema domótico (confort, seguridad, comunicaciones y ahorro energético) incidiendo en los aspectos tecnológicos de captura de información y actuación. Más tarde, se revisa la arquitectura de control de un edificio mediante las normas y los estándares específicos existentes en el mercado, mostrando, también, los diferentes sistemas implantados con dichas normas. Una vez revisado y valorado el estado actual de esta tecnología, identificando sus virtudes y debilidades, entramos en uno de los temas fundamentales: las redes del hogar. Las comunicaciones juegan un papel muy importante en la evolución de la tecnología domótica. En este tema se introducen las necesidades de las redes del hogar, estudiando las distintas tecnologías existentes: adecuación de las redes de datos clásicas (ethernet) y las redes específicas del hogar (mediante cable eléctrico, teléfono,...). Acabamos el bloque con la integración de distintas tecnologías de comunicaciones y las especificaciones middleware.

Tema 3. Sistemas domóticos y componentes

Objetivos: Estudiar los servicios de los subsistemas domóticos revisando los sensores y actuadores utilizados, y revisar las particularidades de sensorización y actuación de cada subsistema.

Contenido: Análisis del problema. Subsistemas domóticos. Adquisición. Actuación.

Tema 4. Normativa

Objetivos: Entender la automatización y el control como un subconjunto de los servicios de un edificio inteligente, entender la arquitectura del sistema de control de un edificio, revisar las normas y estándares del mercado para el control de edificios, y valorar el estado de la tecnología actual identificando debilidades.

Contenido: Home Automation & Networking. Arquitectura de Sistemas domóticos: topología, soporte físico, nodos de control y protocolos (sistemas por corrientes portadoras y sistemas en bus).

Tema 5. Sistemas domóticos actuales

Objetivos: Estudiar diferentes sistemas implantados en la actualidad con los estándares estudiados y estudiar los servicios de estos sistemas domóticos.

Contenido: Sistemas basados en estándares de buses domóticos. Sistemas propietarios. Sistemas basados en autómatas.

Tema 6. Redes domóticas

Objetivos: Introducir las necesidades de redes en el hogar, discutir los objetivos del diseñador de redes en el hogar, estudiar las tecnologías de comunicación en la edificación, conocer especificaciones de alto nivel para red del hogar, y valorar las ventajas e inconvenientes de las diferentes tecnologías.

Contenido: Justificación e impacto de las redes. Arquitectura de las redes del hogar. Alternativas tecnológicas. Soluciones de alto nivel.

Bloque III. Especificación y diseño

En el último bloque, el alumno ya es consciente de los objetivos y de la evolución domótica, y conoce las herramientas tecnológicas disponibles. Por tanto, es momento de plantear nuevos paradigmas más robustos, expresivos y potentes para poder abordar la concepción y el diseño de este tipo de sistemas utilizando metodologías más cercanas a las IST. Para ello, en el tema siete presentamos el paradigma de agentes como modelo para especificación y diseño de sistemas heterogéneos, proporcionando al alumno las claves para el diseño de un sistema domótico basado en agentes. Finalizamos el contenido teórico de la asignatura pretendiendo que el alumno conciba los servicios del hogar desde una perspectiva global, incidiendo en los sistemas domóticos avanzados y servicios de alto nivel. También, en este último tema, los alumnos exponen los trabajos que han elaborado (en el apartado de evaluación mostraremos la finalidad de los mismos).

Tema 7. Modelado mediante agentes

Objetivos: Entender la problemática en el diseño de un sistema domótico, introducir el modelo de agentes y sistemas multiagentes, y diseñar un sistema domótico basado en agentes.

Contenido: Problemática en el diseño. Agentes. Sistemas multiagente. Modelado del sistema.

Tema 8. e-Home

Objetivos: Estudiar sistemas domóticos avanzados y servicios de alto nivel.

Contenidos: Los contenidos de este último tema versan sobre aplicaciones o modelos avanzados de domótica, ya sean tecnologías específicas o tecnologías existentes en otras áreas que se aplican con éxito en este campo (biometría, realidad virtual, robótica, etc).

En la tabla 2 se muestra la distribución temporal de la materia impartida en teoría.

4. Prácticas de laboratorio

Debido al carácter aplicado del conocimiento que debe poseer el alumno con respecto a la domótica, el componente práctico desempeña un papel importante, concretamente, el 50 % de los créditos totales de la asignatura.

Dentro de los objetivos generales, en el laboratorio se pretende reforzar el contenido descriptivo, haciendo hincapié en las habilidades y actitudes del alumno frente a los problemas que se presentan en la domótica. Para ello se proponen seis prácticas donde el alumno se enfrente a la especificación de requerimientos de un sistema domótico y al diseño de distintas partes del mismo. También, incidimos en los aspectos

Bloque	Tema	Duración
Generalidades	Tema 1. Ámbito de la domótica	2 h.
	Tema 2. Marco legal y competencial	2 h.
	Subtotal	**4 h.**
Tecnología domótica	Tema 3. Sistemas domóticos y componentes	4 h.
	Tema 4. Normativa	4 h.
	Tema 5. Sistemas domóticos actuales	4 h.
	Tema 6. Redes domóticas	6 h.
	Subtotal	**18 h.**
Especificación y diseño	Tema 7. Modelado mediante agentes	4 h.
	Tema 8. e-Home	4 h.
	Subtotal	**8 h.**
	Total	**30 h.**

Tabla 2. Cuadro resumen contenido teórico de Domótica y Edificios Inteligentes

tecnológicos de los estándares existentes en el mercado. Para llevar a cabo estos objetivos utilizamos distintas herramientas y recursos para el modelado e implementación de estos sistemas.

Con el fin de incidir en el diseño general utilizamos Simulink® [7]. Ésta es una herramienta interactiva que permite, a partir de la construcción de diagramas de bloques gráficos, modelar, analizar y simular sistemas dinámicos de tiempo continuo y discreto. Simulink®, concretamente, es una *toolbox* de MatLab®. El concepto de *toolbox* se debe a que es una colección especializada de funciones que permiten trabajar en clases particulares de problemas. Por tanto, Simulink® añade muchas características específicas a los sistemas dinámicos, mientras conserva toda la funcionalidad de propósito general de MatLab®. Sin embargo, el entorno de trabajo que proporciona, un entorno de trabajo visual, permite el uso de esta toolbox sin necesariamente conocimientos previos de MatLab®.

Otra de las herramientas que utilizamos es el sistema operativo QNX® [8]. Las características más destacables de este sistema operativo son: arquitectura de micro-kernel con API POSIX que permite implementar sistemas embebidos, escalabilidad, tiempo real e integración de los recursos de la red. El propósito fundamental de este sistema operativo es proporcionar sistemas abiertos de una forma robusta y escalable adecuada para una amplia gama de sistemas, desde sistemas de recursos restringidos a entornos de computación distribuidos. Todas estas características hacen de QNX® un soporte ideal para la arquitectura de un sistema domótico.

Como caso de estudio de plataforma Middleware se propone JINI™ [9]. La tecnología JINI™ está construida a partir de JAVA mediante librerías de clases. Esta tecnología afronta el problema de computación distribuida utilizando un conjunto simple de interfaces y protocolos. Permite crear *federaciones* de recursos (dispositivos, datos, aplicaciones, etc) disponibles a través de la red de forma transparente al usuario. En general puede ser utilizado para compartir tanto servicios software como hardware, ambos representados como objetos y accesibles mediante Interfaces Java. Las características más importantes y relacionadas íntimamente con los sistemas domóticos son la capacidad para transformar una red heterogénea de dispositivos en una homogénea, ser independiente de la arquitectura de los dispositivos aislados y la adaptación dinámica a los dispositivos.

Como dispositivos, disponemos del IPC@CHIP® Single Chip Embedded-Webserver de Beck [10]. Se trata de un controlador embebido diseñado para aplicaciones en red. El controlador incorpora el hardware y software necesario para tal fin. El hardware consiste en una CPU 186 de 16 bits, memoria RAM y Flash, Ethernet, Watchdog y detección de falta de alimentación. El software preinstalado es un sistema operativo en tiempo real (RTOS) con sistema de ficheros, la pila TCP/IP, un servidor Web, Ftp y Telnet. Este controlador está montado en el kit DK-40 de la misma empresa, mediante el cual tenemos 8 entradas y salidas digitales, 2 interfaces puertos serie (TTL) e interfaz Ethernet (10BaseT). Las características de este controlador permite el uso de interfaces remotas para el control de distintos dispositivos.

Una vez analizadas las herramientas y recursos que utilizamos, a continuación, mostramos el contenido práctico de la asignatura describiendo cada una de las prácticas:

Práctica 1. Recopilación de información

La primera práctica está destinada a que el alumno recopile información sobre la domótica y términos afines. Con esta labor, pretendemos iniciar al alumno en qué es la domótica, cuando y por qué se aplica este término. El estudiante en todo momento ha de ser capaz de valorar la información recopilada.

Para llevar a cabo esta tarea, se plantea la redacción de un informe valorando las referencias recopiladas y clasificándolas según el criterio que estime más oportuno (empresas, organizaciones, estándares, investigación, normativa legal, etc).

Práctica 2. Introducción a Simulink

Antes de abordar el trabajo de la siguiente práctica, es necesario que el alumno disponga de los conocimientos necesarios de la herramienta escogida para realizar el diseño general de un sistema domótico.

Con este objetivo, se dan las nociones básicas de Simulink® sin incidir en MatLab®. Concretamente, se repasan las librerías de que dispone la herramienta, reforzando los bloques que puede necesitar el alumno. A partir de aquí se

realizan una serie de modelos para que el alumno comprenda la metodología de modelado con esta herramienta. Acabamos la descripción de la herramienta con utilidades para la implementación del sistema (máscaras) y con la definición de bloques a partir de funciones de sistema, aplicando un lenguaje imperativo de programación (s-functions).

Práctica 3. Diseño de un sistema domótico mediante Simulink

El objetivo principal de esta práctica es que el alumno sea capaz de analizar el problema de implantar un sistema domótico para una vivienda unifamiliar. Que sea capaz de especificar los requerimientos para esa vivienda en concreto a nivel de los subsistemas de seguridad, confort, comunicaciones y de ahorro energético, y de diseñar el sistema oportuno.

Para la realización, se propone al alumno que diseñe el sistema domótico para una vivienda unifamiliar tipo con dependencias como dormitorios, salón, cocina, jardín, etc. En primer lugar debe especificar los requerimientos del sistema. La fase de diseño se divide en dos partes: la construcción de una librería domótica y el diseño concreto de la vivienda. En la fase de construcción de la librería se pide la construcción de cuatro sublibrerías: generación de eventos, control, adquisición y actuación. Con esto se persigue que el alumno se centre de forma independiente en las magnitudes a controlar (térmicas, eléctricas, ópticas, etc), el propio control (gestión a partir de la adquisición de información), la sensorización y la actuación. El nivel de detalle que se persigue es optativo, condicionado al interés del alumno, por ejemplo si queremos medir la temperatura, el bloque encargado podría realizarse mediante una función matemática que devolviese un valor con respecto a la temperatura, o por ejemplo, que el bloque implementase la adquisición mediante una resistencia NTC. En el diseño concreto de la vivienda se persigue la interconexión de los diferentes bloques diseñados por el alumno. Por último se pide al alumno que implemente las pruebas necesarias que determinen la corrección del sistema diseñado.

Práctica 4. Diseño de un sistema embebido para el control de servicios mediante un bus domótico

Los objetivos que nos marcamos para esta práctica son dos y serán abordados en diferentes niveles de realización. El primero de ellos es la aproximación a la utilización de un protocolo de control basado en bus domótico, que permitirá al alumno extrapolar los conocimientos particulares del protocolo, para comprender la filosofía de funcionamiento de los buses de control domótico. Se pretende estudiar tanto los protocolos de control como los controladores de los sistemas, así como sus interfaces. El segundo objetivo es la introducción al diseño de sistemas embebidos que permitirán el desarrollo de arquitecturas integradas, para la implementación de servicios de control concretos, que podrán ser ofrecidos en diferentes puntos de la red de control domótico. Estos sistemas embebidos podrán implementar tanto el control de la red con el protocolo pertinente como el interfaz con el usuario.

El contenido de la práctica plantea, para el primer objetivo, el estudio de X10, uno de los protocolos de control basados en bus domótico más difundidos. Éste utiliza la línea de corriente para la transmisión de las señales de control. Los controladores permitirán el envío de comandos a través de la red para implementar las funciones deseadas, existiendo interfaces para arquitectura PC que permiten la comunicación, monitorización y configuración del sistema. Los alumnos deberán realizar la implementación del protocolo y la interfaz para el controlador que comunica con el PC. Para el segundo objetivo, utilizando la implementación anterior, se pretende que el alumno sea capaz de abordar el diseño del protocolo y la interfaz para un sistema embebido utilizando QNX®.

Práctica 5. Desarrollo de servicios basados en tecnologías web

En la práctica cinco se fija como objetivo familiarizar al alumno sobre las posibilidades de integración de Internet en el desarrollo de redes de control especializadas en tareas de supervisión y control. Estas tecnologías ofrecen un valor añadido a las actuales soluciones propietarias que ofrece el mercado.

Para ello, se diseñará y realizará un prototipo, proponiendo una solución que utilice los protocolos, interfaces y lenguajes de programación sobre los que se sustentan las soluciones que propone el IPC@CHIP® para

servicios Web. A partir del kit DK-40 que como hemos comentado anteriormente presenta una matriz de entradas y salidas digitales, se propone la conexión de distintos dispositivos para comprobar su correcto funcionamiento.

Práctica 6. Modelado mediante agentes

Esta última práctica tiene como objetivo la especificación y el diseño del sistema utilizando agentes que permitan la conectividad con sistemas generales de procesamiento de información y de comunicaciones.

El alumno ha de ser capaz de especificar y diseñar los posibles agentes involucrados en tal objetivo. En esta práctica hacemos uso de la herramienta JINI™. La implementación de todo el sistema presentaría una amplia dilatación en el tiempo y probablemente fuera de los objetivos docentes, donde se enfatiza en la habilidad y actitud del alumno. Por esto, se pide la implementación de dos o tres agentes representativos del sistema.

Estas dos últimas prácticas que hemos descrito no fueron elaboradas en la primera propuesta de la asignatura. Sin embargo, consideramos oportuno la inclusión de éstas en el segundo año que impartiremos la asignatura para lograr una mayor conexión con el contenido teórico. En la tabla 3 se muestra la distribución de las prácticas de la asignatura y su duración junto con la distribución teórica.

5. Evaluación de la asignatura

El carácter descriptivo de la asignatura en su parte teórica indica la conveniencia de la realización de ejercicios por parte del alumno, que serán entregados para su revisión y evaluación. Este método permite la evaluación continua de la evolución del alumno. Para ello, al finalizar cada sesión de teoría se plantean una serie de casos prácticos (cuestiones y problemas) sobre la materia explicada. Las prácticas de laboratorio serán de asistencia obligatoria para observar la progresión de los alumnos, que además deberán entregar memorias del trabajo realizado. Además se propone la elaboración de trabajos optativos en relación a la materia impartida. Estos trabajos se caracterizan por presentar servicios domóticos de alto nivel que englobamos dentro del tema e-Home. Entre estos trabajos, en el curso anterior se presentaron *Sistemas de Reconocimiento de Voz, Robótica en el hogar y de consumo, Reconocimiento de huellas dactilares en el ámbito de la domótica, Entornos inteligentes (Realidad Virtual), Teletrabajo, etc.*

Toda esta tarea desarrollada de forma continua durante el curso, será evaluada y calificada siguiendo los siguientes criterios: la calificación final C_F de la asignatura se obtendrá de la nota obtenida en teoría, C_T, (evaluación continua y/o examen final), mediante la nota obtenida en prácticas, C_P, y por la realización del trabajo optativo, C_{OPT}.

Teoría	Práctica	Duración
Tema 1. Ámbito de la domótica	Práctica 1. Recopilación de información	4 h.
Tema 2. Marco legal y competencial		
Tema 3. Sistemas domóticos y componentes	Práctica 2. Introducción a Simulink	2 h.
Tema 4. Normativa	Práctica 3. Diseño de un sistema domótico mediante Simulink	6 h.
Tema 5. Sistemas domóticos actuales	Práctica 4. Diseño de un sistema embebido para el control de servicios mediante un bus domótico	6 h.
Tema 6. Redes domóticas	Práctica 5. Desarrollo de servicios basados en tecnologías web	6 h.
Tema 7. Modelado mediante agentes	Práctica 6. Modelado mediante agentes	6 h.
Tema 8. e-Home		

Tabla 3. Diagrama temporal de los contenidos prácticos de la asignatura con respecto al contenido teórico.

$$C_F = 0{,}4C_T + 0{,}4C_P + 0{,}2C_{OPT} \quad (1)$$

De esta forma, al ser una asignatura optativa el alumno puede aprobar la asignatura con la realización de los problemas y las prácticas. Sólo los alumnos que muestren un interés por la asignatura podrán optar a calificaciones de sobresaliente y matrícula de honor.

Aquellos alumnos que no superen la evaluación continua a partir de los supuestos prácticos o no la utilicen podrán realizar un examen final. La calificación de teoría se obtiene mediante la realización del examen final de la asignatura, C_E, o mediante la evaluación continua de los problemas planteados, C_{ECP}, al finalizar las sesiones teóricas.

$$C_T = \max\{C_E, C_{ECP}\} \quad (2)$$

Si el alumno opta por la realización de los casos prácticos, la calificación se reparte de forma pondera a la importancia del tema asociado, proporcional a la duración de los mismos. La calificación de las prácticas, también se reparte de forma pondera a la importancia y duración de las mismas.

6. Conclusiones

En este artículo se ha presentado el enfoque de la asignatura Domótica y Edificios Inteligentes de los planes de estudio de las Ingenierías Informáticas de la Universidad de Alicante. Se ha expuesto la motivación de la incorporación de esta asignatura y los objetivos, orientados por la transformación de la domótica hacia un enfoque con énfasis en la integración de servicios de procesamiento de información y de comunicaciones. Este enfoque da cabida al Ingeniero Informático en un campo donde no había intervenido tradicionalmente. El objetivo a nivel de transmisión de conocimientos tiene un marcado carácter aplicado. A nivel práctico los objetivos enfatizan las habilidades y actitudes del alumno que le permitirán enfrentarse a problemas de ámbito domótico. El método de evaluación propuesto, por su parte, permite una mayor relación profesor-alumno y una evaluación continua de la progresión.

La aceptación de la asignatura en el primer curso en el que se ha impartido ha resultado positiva. En el presente curso académico, la matriculación ha sido superior. Además, alumnos que cursaron la asignatura están realizando el proyecto fin de carrera sobre esta materia. La valoración de la experiencia docente en base a la acogida de los alumnos que está siendo muy positiva, unida a la aceptación a nivel de proyectos fin de carrera, provoca la reflexión sobre el interés de intensificar nuestra actividad investigadora en esta línea.

Referencias

[1] *Plan de Estudios conducente al título de Ingeniero en Informática de la Escuela Politécnica Superior de la Universidad de Alicante.* BOE número: 230-2001.

[2] *Plan de Estudios conducente al título de Ingeniero Técnico en Informática de Sistemas de la Escuela Politécnica Superior de la Universidad de Alicante.* BOE número: 230-2001.

[3] *Plan de Estudios conducente al título de Ingeniero Técnico en Informática de Gestión de la Escuela Politécnica Superior de la Universidad de Alicante.* BOE número: 230-2001.

[4] *Adaptación del Plan de Estudios de Ingeniero Técnico en Informática de Sistemas, de la Escuela Universitaria de Ingeniería Técnica Informática de Oviedo, a los Reales Decretos 614/1997, de 25 de abril, y 779/1998, de 30 de abril*. BOE número: 181-2002

[5] *Resolución de 25 de septiembre de 1996, de la Universidad Politécnica de Madrid, por la que se ordena la publicación del Plan de Estudios para la obtención del titulo de Ingeniero en Informática*. BOE número: 253-1996

[6] Programa de trabajo 2003-2004 del VI Programa Marco de la Unión Europea. *http://www.cordis.lu/fp6/home.html*.

[7] Documentación para productos de Mathworks. *Simulink*. *www.mathworks.es*

[8] Documento técnico. *System Architecture*. QNX RT Platform. *http://www.qnx.com/*

[9] Baker, M.; Juhasz, Z. *Distributed Computing with Java and Jini*. Euro-Par 2001. Manchester. 2001

[10] Schlösser, E.; Gatrost, J. *SC12 Getting Started*. Oct. 2001 *www.bcl.de*

Complejidad Algorítmica: de la Teoría a la Práctica

I. Dorta, C. León, C. Rodríguez, G. Rodríguez, A. Rojas
Universidad de La Laguna
Edificio de Física y Matemáticas
c/ Astrofísico Fco. Sánchez s/n
38271 La Laguna. Tenerife

Resumen

En este trabajo se presentan las herramientas CALL y LLAC que facilitan el análisis de complejidad de aplicaciones tanto secuenciales como paralelas. La descripción del modo de uso se realiza mediante el estudio de un caso práctico. El ejemplo elegido ha sido el Problema de la Mochila 0/1. La resolución del mismo se aborda mediante la técnica de Ramificación y Acotación, presentando dos opciones de implementación: recursiva y paralela.
La principal aportación del uso de estas herramientas en la docencia es establecer un puente entre el análisis teórico de la complejidad de los algoritmos y el análisis práctico de los parámetros de rendimiento de los programas, simplificando la labor de ejecución, recolección y estudio de resultados computacionales.

1. Introducción

En los planes de estudio vigentes en la Universidad de La Laguna se cuenta con asignaturas de contenidos relacionados con la Programación Paralela y Distribuida [4] que se concretan en las asignaturas optativas: *Programación en Paralelo I* y *Programación en Paralelo II* y *Programación Distribuida*.

En ellas, los alumnos deben realizar prácticas de laboratorio que implican el uso de los dos paradigmas de programación de máquinas paralelas estándar: el Paso de Mensajes y la Memoria Compartida. Ello supone la realización de experimentos que permitan confirmar que el algoritmo paralelo que se ha diseñado proporciona alguna ventaja frente al algoritmo secuencial. El uso de las herramientas CALL y LLAC facilita esta labor.

En este trabajo, se presenta un ejemplo para ilustrar cómo se usarían las herramientas CALL y LLAC en la docencia de "*Programación en Paralelo*". El problema que se ha considerado es el Problema de la Mochila 0/1 [3]. Se ha optado por un problema de optimización combinatoria porque tienen una gran trascendencia en todos aquellos sectores productivos y usuarios de software que se dediquen al desarrollo de aplicaciones de ayuda a la toma de decisiones y optimización de recursos. Dado que el objetivo de este trabajo es mostrar la utilidad del software en la docencia de optimización matemática, sólo se incluyen los conocimientos elementales necesarios para empezar a trabajar con ambos paquetes.

En el apartado siguiente se presentan las herramientas CALL y LLAC. En la tercera sección se define el caso de estudio y se presentan dos aproximaciones para su resolución: secuencial y paralela, se describe cómo se instrumenta el código y cómo interpretar los resultados. Finalmente se exponen las conclusiones y trabajos futuros.

2. Las herramientas CALL y LLAC

El análisis de complejidad de un algoritmo produce como resultado una "*función de complejidad*" que da una aproximación del número de operaciones que realiza. La herramienta CALL [5] permite anotar con dicha fórmula de complejidad el código C que implanta el algoritmo. Por ejemplo, para el clásico producto de matrices cuadradas C = A×B, donde A y B son matrices de dimensión N×N, el código de la llamada al procedimiento se anotaría de la siguiente forma:

```
#pragma cll mp mp[0] + mp[1]*N + \
                mp[2]*N*N + mp[3]*N*N*N
  MatrixProduct(A, B, C, N);
#pragma cll end mp
```

Donde C_0 = mp[0], C_1 = mp[1], C_2 = mp[2] y C_3 = mp[3] son las constantes asociadas a la fórmula de complejidad del algoritmo ($C_0 + C_1 N + C_2 N^2 + C_3 N^3$).

Los valores de dichas constantes, para una arquitectura dada, se calculan mediante regresión lineal por LLAC. LLAC es una herramienta estadística basada en R [2] que analiza los datos aportados por la ejecución del código instrumentado mediante CALL.

3. Caso de estudio: El Problema de la Mochila 0/1

Consideremos la siguiente definición del Problema de la Mochila 0/1:

"Se dispone de una mochila de capacidad C y de un conjunto de N objetos. Se supone que los objetos no se pueden fragmentar en trozos más pequeños, así pues, se puede decidir si se toma un objeto o si se deja, pero no se puede tomar una fracción de un objeto. Supóngase además, que el objeto k tiene beneficio b_k y peso p_k, para $k=1,2,...,N$. El problema consiste en averiguar qué objetos se han de insertar en la mochila sin exceder su capacidad, de manera que el beneficio que se obtenga sea máximo".

Su formulación como un problema de optimización es:

$$max \sum_{k=1}^{N} b_k x_k$$

sujeto a: $$\sum_{k=1}^{N} p_k x_k \leq C$$

$$x_k \in \{0,1\} \quad k \in \{1,...,N\}$$

La Ramificación y Acotación [1] es un método general que permite resolver un amplio rango de problemas de optimización combinatoria. La solución de un problema consiste en un vector de enteros que cumple un número de restricciones y optimiza una función objetivo. La función objetivo debe ser maximizada o minimizada.

Dado que el área de soluciones contiene un número de elementos exponencial, es imposible explorar todas las soluciones en un tiempo razonable para problemas grandes. La técnica de Ramificación y Acotación intenta reducir el número de soluciones factible por exploración sistemática del área de solución. Los algoritmos de Ramificación y Acotación dividen el área de solución paso por paso y calculan una cota del posible valor de aquellas soluciones que pudieran encontrarse más adelante. Si la cota muestra que cualquiera de estas soluciones tiene que ser necesariamente peor que la mejor solución hallada hasta el momento, entonces no necesitamos seguir explorando esta parte del árbol. Por lo general, el cálculo de cotas se combina con un recorrido en anchura o en profundidad.

En los párrafos siguientes se presentan dos implementaciones, una secuencial en C y una paralela con MPI, de un algoritmo que resuelve el Problema de la Mochila mediante la técnica de Ramificación y Acotación.

3.1. Implementación Secuencial

El Algoritmo 1 muestra el código para resolver el problema de forma recursiva. El número de objetos a insertar en la mochila se almacena en la variable N, en las variables w y p se guardan los pesos y los beneficios de cada uno de ellos, mientras que M representa la capacidad. Puesto que se trata de un problema de maximización se da el valor inicial de $-\infty$ a la variable que almacena la mejor solución encontrada hasta el momento *bestSol* (líneas 1 a 4).

Entre las líneas 8 y 18 se define la función de acotación (*lowerUpper*). Se utiliza la misma función tanto para calcular la cota inferior como la cota superior. La cota inferior se define como el máximo beneficio que se puede obtener para un subproblema dado. Mientras que la cota superior incluye la parte proporcional del beneficio del último objeto que no se pudo insertar en la mochila.

La función *knap* (líneas 20-35) implementa el algoritmo de Ramificación y Acotación de forma recursiva. En la línea 29 se realiza la llamada que estudia la posibilidad de insertar el objeto k, mientras que en la línea 30 se considera su no inclusión. De las líneas 22 a la 26 se implementa la condición que actualiza los valores de la mejor solución (*bestSol*) encontrada hasta el momento.

Estamos interesados en conocer el comportamiento del algoritmo, concretamente la pregunta que nos gustaría responder es ¿cuántos nodos del espacio de búsqueda se visitan para encontrar la mejor solución?.

```
1     /* …ficheros de cabecera */
2     number N, M;
3     number w[MAX], p[MAX];
4     number bestSol = -INFINITY;
5
6     #pragma cll code double numvis;
7
8     void lowerUpper(number k, number C, number P,number *L, number *U) {
9       number i, weig, prof;
10      if (C < 0) {*L = -INFINITY; *U = -INFINITY;  }
11      else {
12        for(i=k, weig = 0, prof = P; weig <= C; i++)
13        {weig += w[i]; prof += p[i];}
14        i--;
15        weig -= w[i]; prof -= p[i];
16        *L = prof;  *U = prof+(p[i]*(C-weig))/w[i];
17      }
18    }
19
20    number knap(number k, number C, number P) {
21      number L, U, next;
22      if (k < N) {
23        lowerUpper(k,C,P,&L,&U);
24        if (bestSol < L) {
25          bestSol = L;
26        }
27        if (bestSol < U) { /* L <= bestSol <= U */
28          next = k+1;
29          knap(next, C - w[k], P + p[k]);
30          knap(next, C, P);
31    #pragma cll code numvis += 2 ;
32        }
33      }
34      return bestSol;
35    }
36
37    int main(int argc, char ** argv) {
38      number sol;
39      readKnap(data);
40    #pragma cll code double numvis = 0.0;
41    #pragma cll kps kps[0]*unknown(numvis) posteriori numvis
42                  /* obj. sig., capacidad rest., beneficio */
43      sol = knap(   0,          M,                0);
44    #pragma cll end kps
45      printf("\nsol = ", sol);
46    #pragma cll report all
47      return 0;
48    }
```

Algoritmo 2. Implementación Secuencial recursiva anotada

Para ello anotamos el código con directivas de CALL. La línea 6 le dice a CALL que se va a definir una variable de tipo `double` para almacenar el número de nodos visitados, *numvis*. A dicha variable se le da inicialmente el valor cero (línea 40). La directiva *code* de CALL proporciona la posibilidad de insertar código fuente al programa anotado. Este código no modifica el programa original, sólo se utiliza en las sentencias CALL. La cadena *cll* que se coloca después de la palabra reservada `#pragma` le indica al compilador de C que se trata de una directiva para el compilador de CALL. Si no se dispone del mismo, simplemente se interpreta como un comentario.

La línea 41 crea un experimento CALL típico. El identificador *kps* especifica su nombre. Este experimento mide el tiempo de ejecución de las sentencias entre las directivas de inicio y finalización (línea 44). El `pragma cll end` va seguido del nombre del experimento y hace que el cronómetro de CALL se pare, se guarden los tiempos y se prepare para la próxima ejecución.

La fórmula que sigue al identificador del experimento *kps* (línea 41) ha de sintetizar la

fórmula de complejidad que nosotros creemos que describe el comportamiento del tiempo. Asociadas a la fórmula se tienen las constantes ksp[0],…,kps[i]. La sintaxis de CALL requiere que se utilice el nombre del experimento para indexar las constantes asociadas con su fórmula de complejidad. Estas constantes pueden ser evaluadas con la herramienta LLAC.

Nuestra intuición nos dice que el tiempo invertido por el algoritmo de Ramificación y Acotación está relacionado con el número de nodos del espacio de búsqueda que se visitan. Así pues, nuestra fórmula se ha de escribir en términos de la variable *numvis* - número de nodos visitados. Sin embargo, el valor final de *numvis* sólo se conocerá una vez resuelto el problema, esto es cuando acabe la llamada a *knap(0,M,0)*. Por lo tanto, en la fórmula de complejidad se indica que dicho valor es desconocido (`unknown`) y que se evaluará al final (`posteriori`).

El número de nodos visitados (*numvis*) se incrementa cuando se tratan los dos nuevos subproblemas que se obtienen a partir del que se esté estudiando en cada momento, esto es, cuando recursivamente se resuelven los problemas que "si" incluyen al siguiente objeto (línea 29) y "no" lo incluyen (línea 30). Este incremento hemos de indicárselo a CALL insertando la directiva correspondiente en la línea 31.

Finalmente, se ha de añadir la directiva `report` después de la especificación de todos los experimentos. Esta directiva le dice al compilador de CALL que genere un fichero con todos los resultados obtenidos durante la ejecución. En nuestro caso, la colocamos justo antes de la sentencia `return` de la función *main* (línea 46). Se ha utilizado el calificador `all` que guarda los resultados de todos los experimentos definidos en el programa. Alternativamente, se puede especificar una lista con los identificadores de los experimentos de los cuales se quieren almacenar los resultados.

Una vez anotado el código fuente con las directivas de CALL se puede compilar con cualquier compilador de C y se seguirá ejecutando exactamente como si no se hubiera modificado. Esto es debido a que el compilador ignora todas las directivas CALL tratándolas simplemente como comentarios.

Supongamos que el Algoritmo 1 está almacenado en el fichero *kpr.c*. Procedemos a procesarlo con el compilador de CALL utilizando el comando:

```
> call kpr.c
```

Como resultado se producen dos ficheros de salida: *kpr.cll.c* y *kpr.cll.h*. Para saber qué código añade CALL al programa original sólo hay que echarle un vistazo a estos ficheros. Ahora se procede a compilar estos ficheros con un compilador de C, indicándole dónde están los ficheros de CALL que es necesario incluir (`/usr/local/CALL/include`):

```
>cc -o kpr kpr.cll.c
```

Al ejecutar el programa resultante (`kpr`) se obtiene un fichero con los resultados del experimento definido: *kpr.c.dat*. La Figura 1 muestra el contenido de este fichero. El programa se ejecutó sobre una mochila de capacidad M = 1.254.202 y número de objetos N = 5000. Fijemos nuestra atención en las dos últimas líneas del fichero. En ellas aparecen los resultados de nuestro experimento. Puesto que se trata de una ejecución secuencial, el número de CPUS usadas es 1. Por defecto, el nombre de la CPU es 0. El número de nodos visitados (*numvis*) ha sido de 261.134 y el tiempo en el que se ha ejecutado la llamada a la función *knap()* ha sido de 0,16 segundos. En el manual de usuario de CALL [5] se puede encontrar una descripción detallada del resto de la información que aparece en el fichero.

```
EXPERIMENT: "kps"
BEGIN_LINE: 115
END_LINE: 119
FORMULA: p 0 p 1 v 0  * +
INFORMULA: kps[0]+kps[1]*numvis
MAXTESTS: 131072
DIMENSION: 2
PARAMETERS:
NUMIDENTS: 1
IDENTS: numvis
OBSERVABLES: CLOCK
COMPONENTS: 1 numvis
POSTFIX_COMPONENT_0: 1
POSTFIX_COMPONENT_1: v 0
NUMTESTS: 1
SAMPLE:
  CPU  NCPUS  numvis    CLOCK
    0      1  261134.0  0.16491100
```

Figura 1. Contenido del fichero knr.c.dat

```
1        busy[0] = 1; for i = 1, nProcs { busy[i] = 0;}

2        idle = nProcs - 1;
3        //Send initial subproblem to first idle slave
4        auxSp = sp.initSubProblem();
5        outputPackect.send(firstIdle,
6                           auxSp,   // initial subproblem
7                           bestSol, // bestSolution
8                           sol);    // current solution
9        idle--;
10       IDLE2WORKING(busy,firstIdle);  // mark this slave like working
11       while (idle < (groupSize-1)) { // while there are working slaves
12          recv(source, flag);
13          while(flag) {
14              if (SOLVE_TAG) {        // receive the final solution
15                inputPacket.recv(source,
16                                 bestSol,  // best solution
17                                 sol);     // current solution
18              }
19              if (BnB_TAG) {              // receive a slave request
20                inputPacket.recv(source,
21                                 high,     // upper bound
22                                 nSlaves); // num. of reuiered slaves
23                if ( high > bestSol){      // problem to branch
24                  total= ((nSlaves <= idle)?nSlaves:idle);
25                  for i = 1, total { idle--; IDLE2WORKING(busy,i); }
26                  outputPacket.send(source,
27                                    total,  // num. of assigned slaves
28                                    bestSol,     // best Solution
29                                    1,..., total // slaves identifiers
30                                   );
31                }
32                else { // the problem must be bounded
33                  outputPackted.send(source, DONE);
34                }
35              }
36              if (IDLE_TAG) {     // signal of an idle slave
37                inputPacket.recv(source, IDLE);
38                idle++;
39                WORKING2IDLE(busy,source); // mark this slave like idle
40              }
41              recv(source, flag);
42          } // while (flag)
43       } // while (idle < (groupSize-1))
44       // Send the ending message
45       for i = 1, groupSize { outputPacket.send(i, END); }
```

Algoritmo 1. Implementación del Maestro

3.2. Resolución Paralela

La versión paralela del algoritmo de Ramificación y Acotación para resolver el problema de la mochila se ha implementado mediante Paso de Mensajes [6] siguiendo un esquema *Maestro/Esclavo*.

El *Maestro* (véase el Algoritmo 2) es el responsable de la coordinación entre tareas, para ello, cuenta con la estructura de datos *busy* donde registra el estado de ocupación de cada uno de los esclavos (línea 1). Al comienzo de la computación todos los esclavos se marcan como ociosos.

El subproblema inicial (*auxSp*), el mejor valor de la función objetivo (*bestSol*) y el mejor vector solución hasta el momento se envía al primer esclavo ociosos (líneas 3-10). Mientras existan esclavos libres el *Maestro* recibe información de ellos y decide la siguiente acción a ejecutar dependiendo de si el problema está o no resuelto (línea 14), si hay una solicitud de esclavos (línea 19) o si los esclavos no tienen nada que hacer (línea 36). Si el problema está resuelto se reciben

```
1    while (1) {
2      recv(source, flag);
3      while (flag) {
4        if (END_TAG){                  // receive the finishing message
5          inputPacket.recv(MASTER, END); return;
6        }
7        if (PBM_TAG){                  // the problem to be branched
8           inputPacket.recv(source,    // source = slave or master:
9                            auxSp,     //   the initial subproblem
10                           bestSol,   //   the best solution value
11                           sol);      //   the current solution
12          auxSol = sol;
13          bqueue.insert(auxSp);       // insert in the local queue
14          while(!bqueue.empty()) {
15            auxSp = bqueue.remove();  // pop from the local queue
16   #pragma cll code numvis++;
17            high = auxSp.upper_bound(pbm,auxSol);    // upper bound
18            if ( high > bestSol ) {
19              low = auxSp.lower_bound(pbm,auxSol);   // lower bound
20              if ( low > bestSol ) {
21                bestSol = low;
22                sol = auxSol;
23                outputPacket.send(MASTER,    // send to the Master:
24                                  SOLVE_TAG,    // problem solved
25                                  bestSol,      // best solution value
26                                  sol);         // solution vector
27              }
28              if ( high != low ) {
29                rSlaves = bqueue.getNumberOfNodes();
30                op.send(MASTER,
31                        BnB_TAG,
32                        high,                 // upper bound
33                        rSlaves);             // num. of slaves req.
34                inputPacket.recv(MASTER,
35                                 nfSlaves,   // num. of slaves assig.
36                                 bestSol,    // updated best solution
37                                 rank {1,..., nfSlaves});
38                if ( nfSlaves >= 0) {
39                  auxSp.branch(pbm,bqueue);   // branch
40                  for i=0, nfSlaves{          // send problems to slaves
41                    auxSp = bqueue.remove();
42   #pragma cll code numvis++;
43                      outputPacket.send(rank,  // send to the slave:
44                                        PBM_TAG, //   tag
45                                        auxSp,   //   problem
46                                        bestSol, //   best solution value
47                                        sol);    //   the solution vector
48                  }
49                } // if nfSlaves == DONE the problem is bounded (cut)
50          } } }
51          outputPacket.send(MASTER, IDLE_TAG);  /idle slave
52        }
53        recv(source, flag);
54      } // while (flag)
55   } // while(1)
```

Algoritmo 3. Implementación del Esclavo anotada

el mejor valor de la función objetivo y el vector solución y se almacenan (líneas 15-18). Cuando el *Maestro* recibe una solicitud de "*nSlaves*" esclavos libres, viene acompañada del valor de la cota superior (*high*). Si el valor de la cota superior es mayor que el valor actual de la mejor solución (*bestSol*) la respuesta al esclavo incluye el número de esclavos libres que pueden ayudar a resolver el problema (*total*) - líneas 26-30. En otro caso, la respuesta indica que no es necesario trabajar en ese subárbol de búsqueda (línea 33). Cuando el número de esclavos libres es igual al valor inicial, el proceso de búsqueda finaliza y el Maestro

notifica a todos los esclavos que dejen de trabajar (línea 45).

Cada *esclavo* (Algoritmo 3) trabaja acotando los problemas que recibe (líneas 8-12). Se generan nuevos subproblemas mediante llamadas a la función de ramificación *branch()* (línea 39). El *esclavo* pide información sobre esclavos libres al *Maestro* (líneas 30-33). Si no hay otros esclavos libres que le ayuden en su tarea, el *esclavo* continúa trabajando localmente. En otro caso, elimina subproblemas de su cola local y los envía directamente a los otros esclavos que le hayan asignado (líneas 39-47).

Al igual que en el caso secuencial, queremos estudiar el número de nodos visitados (*numvis*). Ahora este valor está distribuido entre los *esclavos*, porque el *Maestro* no realiza labores de acotación. Por lo tanto, anotaremos el código paralelo exactamente igual que el secuencial, cambiando sólo el lugar donde se incrementa el número de nodos visitados. Las directivas CALL que lo hacen se han añadido en los puntos en los que se extraen problemas de la cola local de cada esclavo ya sea para estudiarlo (línea 16) o para enviárselo a otro esclavo para que lo resuelva (línea 42).

3.3. Estudio de los resultados

La herramienta LLAC al ser una extensión de R, proporciona la posibilidad de hacer un análisis de las muestras que se obtienen al ejecutar los experimentos generados con CALL. Con las mismas muestras se pueden realizar distintos tipos de representaciones para estudiar cuanto se ajustan la fórmula con la que habíamos anotado el programa y los resultados obtenidos.

Las ejecuciones secuenciales del problema de la mochila se ejecutaron sobre un procesador AMD-Duron a 800 MHz y 256 Mb de memoria. El experimento consistió en generar aleatoriamente diez problemas de la mochila con capacidad en el rango [500, 5000]. La Figura 2, generada con LLAC, muestra los resultados obtenidos. Cada uno de los diez puntos redondos asociado a cada tamaño (500-5000) representa el número de nodos que se visitó para resolver ese problema. Las "x" unidas con una línea punteada representan la media de nodos visitados. La línea continua representa a un polinomio de segundo grado, lo que nos indica que el número medio de nodos visitados se aproxima a una parábola. De la gráfica también se concluye que a mayor número de objetos mayor dispersión en el número de nodos visitados. Este comportamiento creemos que está ligado al generador de problemas aleatorio que se está utilizando.

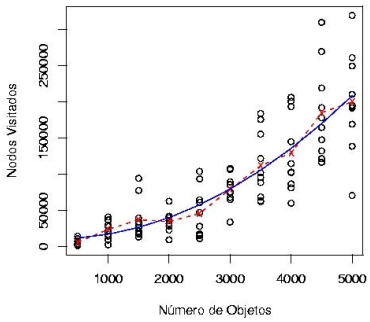


Figura 2. Resultados del Secuencial

Un parámetro muy interesante de estudiar en las implementaciones paralelas de los algoritmos de ramificación y acotación es el "equilibrado de la carga de trabajo" entre los distintos procesadores que intervienen en la ejecución del algoritmo.

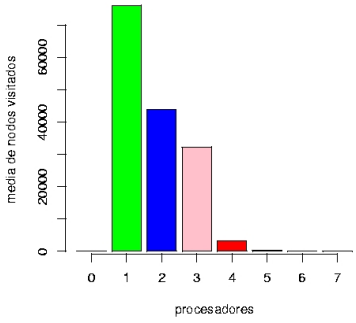


Figura 3. Resultados ejecución paralela

La Figura 3 muestra la distribución media de nodos visitados entre ocho procesadores de cinco ejecuciones de un problema de la mochila generado aletoriamente de tamaño 4000. Se trata de un conjunto de máquinas heterogéneo. Las dos primeras son AMD-Duron a 800 MHz y el resto a 500 MHz, todas con 256 Mb de memoria. Nótese que el procesador cero no visita ningún nodo puesto que se trata del *Maestro*. Creemos que el primero de los esclavos explora la mayor parte del espacio de búsqueda, porque se trata de un procesador más rápido y debido al tamaño del problema, no queda mucho trabajo para los últimos. También hemos notado una gran diferencia entre el número de nodos visitados de una ejecución a otra. La Figura 4 muestra los resultados de los cinco experimentos que dan lugar a la media de la Figura 3. Creemos que el incremento de nodos en la segunda ejecución se debe a que existía algún otro proceso en la máquina uno que consumía muchos recursos. Esto implicaría que la recepción de las cotas que permitirían podar se realiza más tarde, por lo que se explora un espacio de búsqueda mayor.

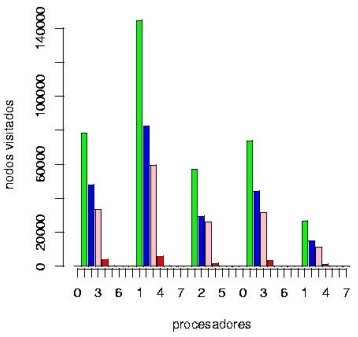


Figura 4. Cinco Ejecuciones Paralelas para N=4000

4. Conclusiones

Se ha presentado a través de un ejemplo el modo de uso de las herramientas CALL y LLAC para el análisis de complejidad de algoritmos.

El ejemplo utilizado forma parte de las prácticas de laboratorio de las asignaturas *Programación en Paralelo I y II* que se imparten en la Ingeniería Superior en Informática en la Universidad de La Laguna. Sin embargo, consideramos que estas herramientas serían de gran utilidad en la docencia de asignaturas que incluyan en sus contenidos la descripción de técnicas algorítmicas. Las herramientas proporcionan la posibilidad no sólo de estudiar implementaciones secuenciales de algoritmos, sino también paralelas.

Además, el algoritmo paralelo propuesto para resolver el problema de la Mochila 0/1, se puede generalizar para resolver mediante Ramificación y Acotación otro tipo de problemas.

Agradecimientos

Este trabajo ha sido financiado parcialmente por los proyectos del ministerio de Ciencia y Tecnología: TIC2002-04498-C05-05 (TRACER) y TIC2002-04400-C03-03 (PELICAN).

Referencias

[1] Dorta I., León C., Rodríguez C., Rojas A. *Parallel Skeletons for Divide-and-Conquer and Branch-and-Bound techniques*. In Proceeding of 11th Euromicro Conference on Parallel, Distributed and Network based Processing, 2003.

[2] Ihaka R., Gentleman R. *R, A language for Data Analysis and Graphics.* Journal of Computational and Graphical Statistics, 5 (3), pp. 299-314, 1996.

[3] Martello S., Toth P. *Knapsack Problems: Algorithms and Computer Implementations*. Jonh Wiley & Sons Ltd, 1990.

[4] Planes de Estudio del CSI. http://www.csi.ull.es

[5] Rodríguez, G. *CALL: a complexity Analysis Tool*. Proyecto Fin de Carrera, Centro Superior de Informática, Universidad de La Laguna, Junio 2002. http://nereida.deioc.ull.es/~call.

[6] Snir M., Otto S., Huss S., Walker D. and Dongarra J. *MPI-the Complete Reference*. 2nd Edition, MIT Press, 1998.

Sistemas operativos

La programación concurrente y el interbloqueo en la asignatura de Sistemas Operativos

Miguel Riesco Albizu, Marián Díaz Fondón
Depto. de Informática
Universidad de Oviedo
e-mail: albizu@lsi.uniovi.es
fondon@correo.uniovi.es

Resumen

La programación concurrente es un tópico dentro de la docencia de los Sistemas Operativos. En este artículo se revisa su papel dentro de la asignatura y su relación, excesivamente estrecha en muchos casos, con otro tema de la misma asignatura como es el del interbloqueo.

1. Introducción

Dentro de los temas típicos que se imparten dentro de la materia de sistemas operativos está el de la programación concurrente. Este tema aparece incluido en los modelos curriculares más utilizados [9, 10], dentro del cuerpo de conocimiento de *Sistemas Operativos* (en concreto en el OS2) en el primer caso y en el módulo 4.3 (*Sistemas operativos y Arquitectura de Computadores I)* en el segundo. Los libros de texto que existen sobre la materia también recogen este tema, con mayor o menor profusión.

Lo mismo ocurre, como no podía ser menos, con la inmensa mayoría de los programas de las asignaturas de Sistemas Operativos que se imparten en las distintas Universidades.

Ante semejante unanimidad, parece lógico considerar que la inclusión de esta materia en el programa de una asignatura de este estilo está fuera de toda duda o discusión. En este artículo comenzaremos discrepando de esta opinión, por muy generalizada que esté, intentando aportar las razones que nos llevan a disentir.

Por otra parte, otro tema que también se incluye en todos los libros y temarios de Sistemas Operativos es el del interbloqueo. Sin discutir en este caso su adecuación con la materia, sí discrepamos de su posición temporal en el desarrollo de la docencia de la asignatura, generalmente muy ligado al tema de programación concurrente, tanto que muchas veces aparece incluso incluido en él.

2. Programación concurrente y Sistemas Operativos

El tema de programación concurrente, como su propio nombre indica, es un tema eminentemente de programación. Trata de explicar las peculiaridades que presenta la realización de programas concurrentes, los problemas más habituales que aparecen y los medios más utilizados para resolverlos.

Ante este contenido puede surgir la pregunta de por qué se incluye este tema dentro de la asignatura de Sistemas Operativos, dado que, aparentemente, no tiene ninguna relación directa con los contenidos esperados de esta asignatura. A esa pregunta se suelen dar varias respuestas:

1. Con la inclusión de la multiprogramación, el sistema operativo debe manejar procesos concurrentes, con lo que un estudiante de sistemas operativos debe conocer los problemas que pueden aparecen en estos casos, así como la manera de solucionarlos.
2. El sistema operativo suele incluir mecanismos de sincronización y de comunicación entre procesos, por lo que se debe conocer para qué y cómo se van a utilizar estos mecanismos.
3. El propio sistema operativo se implementa muchas veces como un conjunto de funciones concurrentes, con lo que se debe ser capaz de entender, y en su caso desarrollar, programas concurrentes correctos.

Todas estas razones hacen referencia al carácter instrumental de esta materia. Es decir, es necesa-

rio conocer programación concurrente para estudiar y comprender la materia específica de Sistemas Operativos, que no es otra que estudiar la estructura interna y el funcionamiento de un sistema operativo. Podemos resumir, por tanto, esas razones para la inclusión de programación concurrente en una sola: "Para la docencia de Sistemas Operativos nos hace falta la programación concurrente; como no se imparte en otro sitio, hay que hacerlo aquí".

Esta razón podría considerarse académicamente válida si este tema fuera única y exclusivamente de utilidad dentro de la asignatura de Sistemas Operativos. Sin embargo, hay otras asignaturas, donde también son necesarios en mayor o menor grado los conceptos básicos de programación concurrente. Centrándonos en el primer modelo de currículo de los antes citados [9] son necesarios conceptos de programación concurrente en los cuerpos de conocimiento de *Programming Fundamentals* (PF7, en concreto), *Programming Languages* (PL11), *Information Management* (IM5 e IM6) y *Net-Centric Computing* (NC7), además de en el *Operating Systems* (OS2) ya citado.

Por otro lado, además de en las asignaturas que desarrollan estos cuerpos de conocimiento, en muchos Centros se considera la materia como digna de figurar como asignatura independiente en sus planes de estudio, tal y como recomienda el modelo curricular UNESCO-IFIP[10].

Volviendo al razonamiento de "inclusión por necesidad", si lo aplicamos a otras partes de la asignatura podíamos incluir también un tema sobre programación, dado que es necesaria para entender los algoritmos que se explican, otro sobre estructuras de datos, para poder comprender las estructuras de datos, muchas veces complejas, que usan los sistemas operativos. O incluso podríamos, como se hace en [5] introducir un tema sobre modelado analítico y teoría de colas, que puede ser muy interesante para estudiar y comparar el comportamiento de las políticas que se emplean en distintas partes de un sistema operativo. Como obviamente esto no parece muy adecuado, podemos concluir que el tema de "programación concurrente" no debería estar incluido en la asignatura de Sistemas Operativos.

3. ¿Dónde debería tratarse la programación concurrente?

Centrándonos en las directrices del Consejo de Universidades para las Ingenierías Técnicas en Informática [3, 4] (y podemos extenderlo a las Ingenierías Superiores [2]), podremos ver que el tema de "programación concurrente" no aparece en el descriptor de ninguna asignatura, con lo que podrá impartirse legalmente en cualquiera donde tenga sentido incluirse.

Lo que se pretende con este artículo es iniciar un debate para decidir dónde podría ser más adecuado impartir una introducción a los conceptos básicos de programación concurrente.

Nosotros proponemos que sea en alguna de las asignaturas en la que se estructure la materia de "Metodología y Tecnología de la Programación", por tres motivos:

1. Es la materia donde más conceptos relacionados con programación concurrente podemos incluir de los expuestos en [9].
2. Tiene casi el triple de carga docente asignada en las directrices del Consejo de Universidades [2, 3, 4] que la de Sistemas Operativos.
3. Independientemente de currículos y de directrices, el tema de programación concurrente es un tema (como su propio nombre indica) de programación.

4. Otro problema: el interbloqueo

Otro tema en principio relacionado con la programación concurrente es el del interbloqueo. En este apartado estudiaremos esta relación, así como el lugar más adecuado para la manera de abordar su docencia en la asignatura de sistemas operativos.

El interbloqueo es una anomalía que puede ocurrir durante la ejecución de procesos concurrentes debido a la competencia por los recursos. Si bien es cierto que prácticamente ningún sistema operativo real incorpora mecanismos de tratamiento de interbloqueo, esto es debido a una cuestión de la pérdida de rendimiento que conlleva su tratamiento para la baja probabilidad que hay de que ocurra. En un sistema operativo ideal, sin embargo, sí deberían incluirse mecanismos para su tratamiento, dado que existen y son bien conocidos.

Podemos considerar, sin temor a equivocarnos, que el tema del interbloqueo, además de ser un clásico en todos los libros, modelos de currículos, etc., sobre sistemas operativos es un tema que por sus características merece estar incluido en el temario de esta asignatura.

Si bien no se discute, por tanto, su adecuación a la asignatura, sí nos vamos a permitir cuestionar su posición en el desarrollo de la materia. La práctica totalidad de los libros de texto [5, 6, 7, 8], los distintos modelos de currículos [9, 10] y los programas de las distintas asignaturas que sobre el tema se imparten colocan este tema inmediatamente después del correspondiente a programación concurrente. Más aún, en algunas ocasiones [9] se llega a incluir el interbloqueo dentro del tema de programación concurrente.

Este hecho nos parece un importante error, por los motivos que expondremos a continuación, que hace por un lado que no se comprenda del todo bien su objetivo, y que, por otra parte, se complique innecesariamente un tema tan árido para los alumnos como es la programación concurrente.

5. La programación concurrente y el interbloqueo

Uno de los problemas con los que se puede encontrar el alumno cuando empieza a realizar sus primeros programas concurrentes es el del interbloqueo. En este contexto el interbloqueo que se produce es debido a una mala programación. Por ejemplo, en los clásicos intentos de solución al problema de la exclusión mutua se muestra este problema debido a un mal planteamiento del problema.

La única solución que hay para solventar los interbloqueos de este tipo es eliminar el error en la lógica del programa para que los procesos implicados no estén esperando el uno por el otro.

La utilización de mecanismos de sincronización de alto nivel facilita la realización de programas concurrentes libres de errores, de la misma manera que el desarrollo de un programa en un lenguaje de alto nivel es más fácil e introduce menos errores que si lo realizamos en ensamblador. Pero tanto en un caso como en otro sólo una buena programación puede asegurarnos tener éxito en nuestro propósito de construir programas correctos.

Lo que está claro es que, en ninguna circunstancia, podremos utilizar ninguna técnica que nos asegure la realización de un programa concurrente libre de interbloqueo ante un problema dado. Igualmente, tampoco podremos desde el sistema operativo evitar que ese interbloqueo se produzca si los procesos se empeñan en caer en él.

6. Interbloqueo y gestión de recursos

Otro de los campos típicos donde se presenta el problema del interbloqueo es en la gestión de recursos. En este caso varios de los procesos que se ejecutan concurrentemente en el sistema (procesos probablemente independientes entre sí) compiten por el uso de recursos no compartibles, de tal manera que hasta que el proceso que tiene asignado el recurso no lo libere ningún otro podrá utilizarlo. Si otro proceso necesita utilizarlo para continuar su trabajo, lo típico que ocurre es que este proceso suspende su ejecución hasta que el que ocupa el recurso lo libera.

En este contexto el interbloqueo se produce cuando los procesos suspendidos retienen recursos que son necesarios para la continuación de otros procesos, que a su vez retienen los recursos que necesitan los primeros. En definitiva, el problema se debe a que se han asignado los recursos que han ido necesitando los procesos sin ningún criterio, sumado al propio comportamiento de los procesos.

Este tipo de interbloqueo no se da únicamente en los sistemas operativos. Es muy típico también en los sistemas de gestión de bases de datos, por ejemplo, donde los recursos son registros o tablas de una base de datos que los procesos bloquean mientras los manejan si precisan de un acceso exclusivo.

Esta clase de interbloqueo está perfectamente estudiado y puede tratarse sin demasiada dificultad utilizando distintas técnicas, clasificadas normalmente en tres categorías: *prevención*, *evitación* y *detección y recuperación*. El problema que tienen estas técnicas es que sacrifican algo a cambio de solucionar el interbloqueo (la asignación de recursos será más lenta, al tener que comprobar si se puede permitir; se exige seguir una serie de normas a los procesos a la hora de solicitar recursos; etc.)

Este es el motivo de que no se suelan apenas utilizar en sistemas reales. Pero en cualquier caso

es un problema muy interesante de gestión de recursos que debe explicarse dentro del tema de Sistemas Operativos[1].

7. Relación entre el interbloqueo, la programación concurrente y los Sistemas Operativos

Como ya hemos indicado anteriormente, en la mayoría de libros, currículos y temarios de asignaturas de Sistemas Operativos el tema del interbloqueo está muy ligado al de programación concurrente. Tanto es así que habitualmente el tema de interbloqueo sigue inmediatamente al de programación concurrente, o incluso en algunos casos [9] aparece incluido en él como un apartado más. A nuestro juicio esto es un grave error, dado que hace parecer que son temas íntimamente relacionados cuando en realidad no es así. El interbloqueo que aparece cuando tratamos con programación concurrente es radicalmente distinto en su origen del que se presenta debido a la gestión de recursos:

- Los procesos implicados en un interbloqueo del primer tipo son procesos cooperantes, que comparten información y se sincronizan para lograr un objetivo común. En el segundo caso los procesos son independientes, y sólo comparten la necesidad de utilizar recursos comunes.
- El primer tipo de interbloqueo es causado por una mala sincronización, debido a una mala programación, al no considerar la situación que lleva al interbloqueo. En el segundo caso el problema aparece por una falta de control en la asignación de recursos que solicitan los procesos por parte del sistema operativo.
- El segundo tipo de interbloqueo sí es un problema que debe resolver el sistema operativo, mientras que el primer tipo es un problema meramente de programación.

De la misma manera, la solución al problema también es completamente distinta:

- El primer tipo de interbloqueo se soluciona mediante la corrección de los programas implicados en los mismos.
- El segundo tipo de interbloqueo se soluciona realizando la asignación de recursos a los procesos según alguna de las técnicas diseñadas al efecto (previniendo o evitando el interbloqueo, fundamentalmente).

En resumen, el segundo tipo de interbloqueo sí entraría dentro de la materia de sistemas operativos, mientras que el primero debería resolverse dentro del campo de la programación concurrente.

El modelo curricular de la UNESCO [10] sí parece recoger esta sensación, dado que incluye el interbloqueo en dos temas distintos: en el tema de procesos (donde se incluye también el de programación concurrente) en el módulo de *Sistemas Operativos y Arquitectura de Computadores I*, así como en el tema de entrada/salida del módulo de *Sistemas Operativos y Arquitectura de Computadores II*.

8. ¿Y cuál es el problema?

El contenido del tema del interbloqueo que se imparte en la asignatura de Sistemas Operativos se centra única y exclusivamente en el segundo tipo de interbloqueo, como no podía ser de otro modo.

Parece, por tanto, no demasiado adecuada la situación del tema de tratamiento del interbloqueo en el temario de la asignatura, dado que al situarlo a continuación del tema de programación concurrente puede dar la impresión, errónea, de que en ese apartado vamos a ver métodos para resolver los problemas de interbloqueo que nos han ido surgiendo en nuestros programas concurrentes, como en la "Cena de los filósofos" o en el "Problema de la sección crítica". Esto hace que el alumno no comprenda bien el objetivo del tema y su utilidad dentro de un sistema operativo, creyendo en un principio que los métodos de tratamiento que se exponen van a ser la panacea a los problemas que se le presentan en su incipiente vida de desarrollador de programas concurrentes.

El porqué se sitúa en este lugar el tema del interbloqueo no puede explicarse más que por inercia (porque a alguien se le ocurrió en su momento a hacerlo así y tuvo éxito), dado que, como hemos visto, a pesar de haber "interbloqueos" en programación concurrente éstos nada tienen que ver con los que se tratan en este tema.

[1] No hay que olvidar que una definición clásica de "sistema operativo" lo caracteriza como "gestor de recursos"

9. ¿Dónde debería situarse el tema del interbloqueo?

Dentro de los sistemas operativos los posibles problemas de interbloqueo se suelen dar por la utilización de recursos periféricos. Los distintos modelos curriculares suelen incluir un tema de "Gestión de dispositivos" (o "Gestión de entrada/salida"), donde se explica la forma que tiene el sistema operativo de manejar los distintos dispositivos periféricos.

Nuestra propuesta es, por tanto, situar el tema del interbloqueo tras el tema de "Gestión de dispositivos", para entender cómo se gestionan los dispositivos periféricos antes de ver cómo se trata el problema del interbloqueo debido a su compartición.

Con esta medida conseguimos, asimismo, alejarlo del tema de programación concurrente, lo cual nos aporta dos ventajas adicionales:

- Aumentamos el tiempo para que los alumnos asimilen los conceptos de programación concurrente, completamente nuevos para ellos, con lo que podrán comprender mejor las diferencias entre los dos tipos de interbloqueo.
- Evitamos confusiones innecesarias al enfatizar la diferencia entre los dos tipos de interbloqueo.

El modelo curricular UNESCO/IFIP [10] apoya, además, esta decisión.

10. Conclusiones

A pesar de que en buena parte de la bibliografía consultada se considera el tema de programación concurrente como una parte importante de la docencia de Sistemas Operativos, consideramos que no debe ser así. Sólo el hecho de que no se quiera incluir en ninguna otra asignatura, donde probablemente sería más adecuado hacerlo, puede justificar que se imparta en ésta, ante la necesidad que tiene de conocer esa materia para comprender los conceptos que le son propios.

En cualquier caso, si se incluye la programación concurrente dentro del temario de la asignatura de Sistemas Operativos consideramos que debe separarse claramente del tema de "tratamiento del interbloqueo", dado que, por una parte, no tiene prácticamente ninguna relación con él, y que, por otro lado, induce a los alumnos a confundir las dos clases de interbloqueo que se ven en la asignatura, complicando innecesariamente, además, la comprensión del tema de programación concurrente, bastante complejo de por sí.

Referencias

[1] Carretero Pérez, J. y otros. *Sistemas Operativos. Una visión aplicada.* McGraw Hill, 2001

[2] Consejo de Universidades. *Directrices generales propias de los planes de estudios conducentes a la obtención del título oficial de Ingeniero en Informática*. Boletín Oficial del Estado, número 278, pag. 34401 a 34403

[3] Consejo de Universidades. *Directrices generales propias de los planes de estudios conducentes a la obtención del título oficial de Ingeniero Técnico en Informática de Gestión*. Boletín Oficial del Estado, número 278, pag. 34403 y 34404

[4] Consejo de Universidades. *Directrices generales propias de los planes de estudios conducentes a la obtención del título oficial de Ingeniero Técnico en Informática de Sistemas*. Boletín Oficial del Estado, número 278, pag. 34404 y 34405

[5] Deitel, H. M. *Sistemas Operativos. Segunda Edición.* Addison-Wesley Iberoamericana, 1993

[6] Sinlverschatz, A. y Galván, P.B. *Sistemas Operativos*. Quinta Edición. Addison Wesley Longman, 1999.

[7] Stallings, W. *Sistemas Operativos.* Cuarta Edición. Prentice Hall, 2001.

[8] Tanenbaum, A. y Woodhull A. *Operating Systems Design and Implementation*, 2ª edición. Prentice-Hall, 1997

[9] The Joint Task Force on Computing Curricula, *Computing Curricula 2001* IEEE Computer Society, Association for Computing Machinery, 2000

[10] UNESCO/IFIP, *A Modular Curriculum in Computer Science*. UNESCO, 1994.

Simulador didáctico de gestión de memoria con interfaz de autoaprendizaje basado en WWW

Félix Buendía, Julio Sahuquillo, Juan-Carlos Cano
Departamento de Informática de Sistemas y Computadores
Escuela Técnica Superior de Informática Aplicada
Universidad Politécnica de Valencia
e-mail: {fbuendia, jsahuqui, jucano}@disca.upv.es

Resumen

La enseñanza de las materias relacionadas con los Sistemas Operativos cubre un conjunto de contenidos esenciales en cualquier currículo universitario de informática. Las diferentes asignaturas impartidas, incluyen temas tales como la gestión de procesos y gestión de memoria, con una fuerte componente teórica. Aunque dichos conceptos son ampliamente tratados en numerosos libros de texto, no resulta sencillo encontrar entornos adecuados al nivel de conocimientos del alumno, que permitan realizar actividades donde poner en práctica los conceptos introducidos de forma teórica. En este trabajo, se describe una herramienta de simulación realizada de forma "*ad hoc*" para mostrar el funcionamiento de los principales aspectos relacionados con la gestión de memoria en un sistema operativo. Las características de dicha herramienta permiten que el alumno pueda interactuar con el simulador, introduciendo ejemplos de carga y configurando parámetros del sistema tales como el tamaño de la memoria y el algoritmo de gestión de la misma. La herramienta esta desarrollada con tecnología de desarrollo basada en WWW lo que facilita su amplia difusión y utilización.

1. Introducción

Las asignaturas relacionadas con los Sistemas Operativos son un elemento fundamental en cualquier currículo universitario de informática. Desde las propuestas de ACM/IEEE de 1991 [1] hasta las más recientes del 2001 [2], existe un consenso sobre la importancia de esta temática. Las directrices del MEC establecen una serie de descriptores que hacen referencia a aspectos como la gestión de procesos y ficheros así como los recursos de memoria y entrada/salida. Se trata de aspectos con una fuerte componente teórica pero que deben encontrarse apoyados con actividades prácticas. Sin embargo, no resulta sencillo encontrar entornos donde realizar este tipo de actividades, que permitan al alumno fijar con la práctica los conceptos tratados de forma teórica. Tradicionalmente, se han realizado actividades prácticas directamente sobre los sistemas operativos reales, donde el acceso a los mecanismos de gestión de recursos es complejo.

En algunos casos, se dispone de sistemas operativos reducidos como Nachos [4] que aunque presentan una amplia funcionalidad, para asignaturas troncales básicas tienen como principal inconveniente su orientación hacia aspectos de diseño e implementación.

Otras herramientas como RCOS [3] simulan el funcionamiento de un sistema operativo mediante animaciones. Sin embargo, la propuesta de RCOS se centra demasiado en los aspectos visuales y no profundiza en la aplicación de aspectos teóricos. En [5], Magee y Kramer presentan una herramienta que, de manera similar a la que se presenta en esta ponencia, se centra en la demostración visual de aspectos concernientes a aspectos relacionados con los conceptos de concurrencia.

En este trabajo, se presenta un simulador utilizado para mostrar el funcionamiento de la gestión de memoria de un sistema operativo. El simulador incorpora un entorno gráfico que visualiza la actividad del gestor de memoria. Este entorno junto con un menú detallado de ayuda facilita, sin duda, el autoaprendizaje de la materia al alumno.

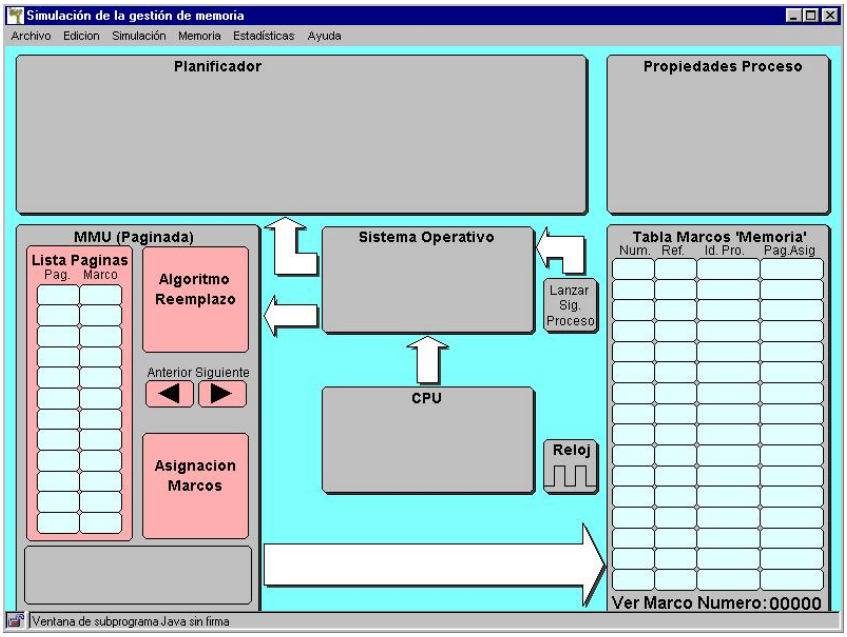


Figura 1. Esquema del simulador de gestión de memoria.

De hecho, son muchos los que durante el presente curso académico lo han utilizado para constatar los resultados de ejercicios propuestos en clase. Para facilitar esta tarea de aprendizaje, el simulador se encuentra disponible en [8].

El resto del trabajo se organiza como sigue. La Sección 2 describe la herramienta de simulación, mostrando su funcionamiento general. La Sección 3 detalla algunos ejemplos de ejercicios basados en la utilización de la herramienta. Finalmente, la Sección 4 presenta las conclusiones del trabajo.

2. Simulador de gestión de memoria

La herramienta de simulación ha sido desarrollada en lenguaje Java. Una primera versión de la misma dio lugar a un proyecto final de carrera [6], dirigido por el primer autor del artículo y enmarcado dentro de la Escuela Técnica Superior de Informática Aplicada de la Universidad Politécnica de Valencia. Nuevas versiones con interfaz mejorada se han ido incorporando como parte del trabajo realizado en el ámbito de un proyecto europeo de educación a distancia [7]. La Figura 1 muestra la pantalla principal de la última versión de la herramienta, que es la que se utiliza para realizar prácticas en las asignaturas de Sistemas Operativos.

2.1. Introducción al simulador

El Simulador de gestión de memoria tiene por objeto mostrar el funcionamiento de algunos aspectos de la gestión de memoria que intervienen en la ejecución de uno o más procesos en un entorno de multiprogramación. Concretamente, la herramienta describe la ejecución de un conjunto

de procesos mediante el algoritmo de planificación *round-robin* y cómo estos procesos generan durante su ejecución accesos a direcciones lógicas qué serán traducidas a direcciones en memoria física. El esquema de traducción de direcciones se basa en una técnica de paginación combinada con algoritmos de reemplazo, lo que se conoce globalmente como memoria virtual mediante paginación bajo demanda.

2.2. Inicio de una sesión

El primer paso en la ejecución de una simulación de gestión de memoria consiste en cargar el fichero de simulación (menú Archivo). Para describir un determinado programa, se ofrece un fichero denominado "Simulacion.inf" que describe las características de los diferentes procesos. La Figura 2 muestra un ejemplo.

```
Proceso1 50000 0
5000
0
100
0
456
4000
1050
end
0
Proceso2 50000 0
2048
7999
2248
3100
230
end
0
Proceso3 60000 0
2148
50000
3477
6000
2340
end EOF
```

Figura 2. Ejemplo de carga.

Este fichero contiene una secuencia de direcciones lógicas agrupadas por cada proceso que se incluye en el sistema. Para ello se utiliza una cabecera con tres campos: el primero hace referencia al identificador del proceso, el segundo el tamaño asignado a dicho proceso y el tercero a la prioridad, que identificará la cola a la que pertenece el proceso. Por ejemplo, la primera línea del fichero contiene la siguiente declaración:

```
Proceso1 50000 0
```

que define el Proceso1, con un tamaño de 50000 bytes y una prioridad 0. A partir de esta línea aparecen valores numéricos separados en diferentes líneas, cada uno de los cuales representa una dirección lógica. Las direcciones lógicas contenidas en un mismo proceso finalizan cuando aparece la etiqueta "end". A continuación, y antes de empezar la especificación de un nuevo proceso, aparece un valor numérico que especifica el desplazamiento temporal en la activación del siguiente proceso respecto al anterior. En este caso, se utiliza un valor 0 que indica que el Proceso1 y el siguiente (Proceso2) se activan al mismo tiempo. Un valor de 2 significaría que deben transcurrir dos instantes de tiempo antes que se active el Proceso2. La aparición de una etiqueta "EOF" a continuación de la última marca "end" indica la finalización del fichero de simulación.

Dentro de la opción "Ver Ventana de Direcciones Lógicas" (menú Edición) se puede editar el contenido del fichero de simulación, para modificar, por ejemplo, alguna de las direcciones lógicas o el tamaño de un proceso.

2.3. Configuración del sistema

Una vez que se ha cargado el fichero de simulación, se puede proceder a la configuración del sistema mediante una opción con dicho nombre del menú Archivo. La Figura 3 muestra un ejemplo. Se puede observar, como esta opción permite asignar diferentes atributos del sistema tales como el tamaño de la memoria, el tamaño de la página, los algoritmos de asignación y reemplazo del sistema.

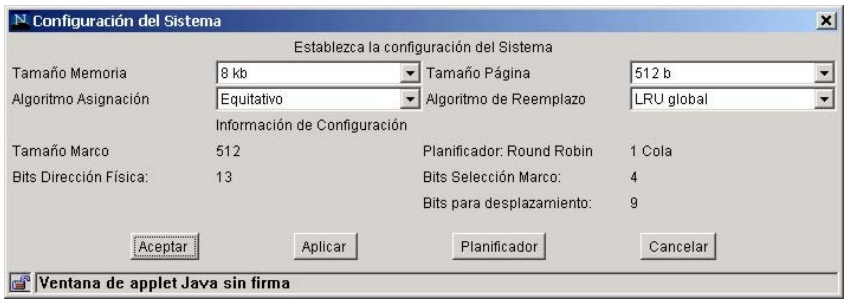


Figura 3. Ventana de configuración del sistema.

- Tamaño de memoria: indica el número de bytes disponibles en la memoria principal. El rango permitido oscila desde 8KBytes hasta 1Mbyte.
- Tamaño de página: indica la capacidad en bytes de cada una de las páginas en que se divide un proceso. Puede seleccionarse entre un rango de valores que van desde 512 bytes a 4 KBytes.
- Algoritmo de reemplazo: determina el método que se utiliza para seleccionar la página víctima de un proceso cuando se produce un fallo de página y no existe un marco de página disponible en memoria. Los posibles métodos a elegir son: LRU: local y global y FIFO: local y global.

En caso de utilizar un algoritmo de reemplazo local, habrá que elegir un determinado algoritmo de asignación, el cual determina el método utilizado para repartir los marcos de página disponibles en memoria entre los procesos del sistema. Existen tres métodos posibles:

- Equitativo.
- Proporcional.
- Prioritario.

La selección de atributos del sistema permite determinar aspectos tales como el tamaño del marco de página, el número de bits de la dirección física y cómo estos se distribuyen (los bits de mayor peso se destinan a la selección del marco y los de menor peso al desplazamiento dentro del marco).

Por otra parte, tal y como se muestra en la Figura 3, mediante la opción "Planificador" también pueden configurarse ciertos parámetros referidos a la Planificación del Sistema. Los parámetros que se pueden configurar son los siguientes:

- Algoritmo de planificación: actualmente, sólo se encuentra implementado un algoritmo de planificación del tipo "round-robin" (turno rotatorio). Cabe matizar, que existen 2 proyectos fin de carrera en fase de escritura destinados a implementar otros algoritmos.
- Número de colas: indica el número de estructuras de datos para almacenar procesos activos. Se pueden elegir de 1 a 5 colas mediante los identificadores cola 0 a cola 4, donde la cola 0 es la más prioritaria.
- Quantum: permite definir para cada cola el "quantum" o cantidad de tiempo que se asocia a un proceso en su turno de ejecución. En este caso, dicho tiempo se evalúa en base al número de direcciones lógicas. Por defecto su valor es 1, es decir, se accede a una dirección lógica en cada instante de tiempo.

2.4. Comienzo de la simulación

Una vez cargado el fichero de simulación y realizada la configuración del sistema, para simular el comportamiento se elige la opción "Comenzar Simulación" dentro del menú de Archivo.

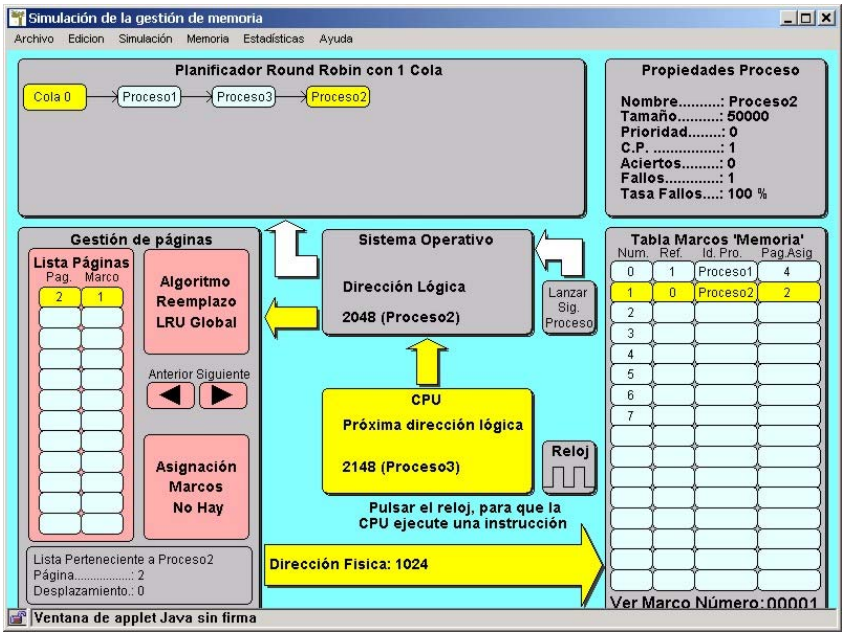


Figura 4. Ejemplo de ejecución del simulador.

Una vez iniciada la simulación, se puede observar la evolución de la misma a través de diferentes zonas de la pantalla mostradas en la Figura 4. Existen dos formas de controlar la simulación, las cuales se pueden seleccionar desde el menú de Simulación:

- Modo paso a paso: permite visualizar los resultados de cada paso (unidad temporal) en que se realiza cada acceso a memoria o la activación de un nuevo proceso. Para ello, el usuario debe pulsar el icono de Reloj que aparece en la parte central de la pantalla.
- Modo automático: en este caso la ejecución de los procesos y por tanto, el acceso a direcciones de memoria se realiza de forma automática sin intervención del usuario.

La parte superior izquierda de la pantalla muestra una representación gráfica de los procesos activos en cada cola. En el ejemplo se utiliza una única cola (Cola 0) que contiene los tres procesos definidos. Además, cada vez que se accede a la dirección lógica de un proceso, éste se pondrá al inicio de la cola y cambiará el color del símbolo que lo representa, para indicar que es el proceso en ejecución.

La parte superior derecha de la pantalla informa del estado del proceso en ejecución. Para ello, se incluye el contador de direcciones lógicas (C.P.) que define el orden relativo de la dirección lógica accedida por el proceso en ese momento y que se actualiza con cada nuevo acceso. Además, se indica el número de accesos realizados hasta el momento actual, diferenciando entre fallos de página (la página que contiene la dirección actual no está en memoria principal y tiene que ser transferida desde disco) y aciertos (la página accedida está ya en memoria). También se incluye el valor de la tasa de fallos.

La parte inferior izquierda representa la gestión de páginas e incluye información como:

- El algoritmo de reemplazo seleccionado.
- El tipo de asignación de marcos si lo hubiera.
- La lista de páginas utilizada por el correspondiente algoritmo de reemplazo para elegir la página víctima y decidir el marco que va a ser liberado. Esta estructura incluye unos iconos ("Anterior" y "Posterior") para poder manejar los elementos de la lista que excedan el tamaño dispuesto en la pantalla (12 páginas). También se incluye información de la dirección lógica actual, como la página y el desplazamiento dentro de ella. Esta lista no representa la tabla de páginas ya que sólo almacena parte de su información (en este caso el número de marco asociado a una página).

La parte inferior derecha de la pantalla representa la Tabla de Marcos de la memoria física. En ella se muestra la siguiente información:

- El número de marco: su valor va desde 0 hasta N, siendo N+1 el número de marcos disponible. En caso de sobrepasar el número máximo de elementos de la tabla en pantalla (16), se podrán utilizar una serie de controles que se activan al colocar el cursor en alguno de los números de la opción "Ver Marco Número". Cuando se activan estos controles aparece el símbolo de una mano y al pulsar el botón izquierdo del ratón el número se incrementa, mientras que al pulsar el botón derecho, se decrementa. Ello permite el acceso a la totalidad de la estructura de la tabla de marcos de página.
- El campo referencia define la antigüedad de las páginas que se almacenan en memoria y sirve para decidir el marco ocupado por la página a reemplazar, en caso de fallo. Por tanto, depende del tipo de algoritmo de reemplazo elegido.
- El identificador de proceso: indica el proceso que accede a la página.
- Página asignada: se trata de la página almacenada en el marco referido.

Por último, la parte central de la pantalla define la relación entre las estructuras descritas previamente. Indica en cada momento, que proceso ha sido elegido por parte del planificador para su ejecución y la dirección lógica implicada en este paso. Dicha dirección puede ser seleccionada de forma automática (modo "Automático") o interactiva a través del icono Reloj (modo "Paso a paso"). Asimismo, se puede activar un nuevo proceso mediante el control "Lanzar Siguiente Proceso", sin que se tenga que esperar al plazo de tiempo definido en el fichero de carga.

También se encuentran en esta parte de la pantalla tres etiquetas que contienen la siguiente información:

- Sistema operativo: indica la dirección lógica que actualmente está siendo traducida.
- CPU: representa la próxima dirección a traducir.
- Dirección física: contiene la dirección física resultado de la traducción actual.

2.5. Resultados de la simulación

Mediante la opción "Ver Ventana de Direcciones Físicas" del menú Edición, se obtiene una traza de la simulación. Esta traza muestra un listado de direcciones físicas como consecuencia del proceso de traducción. La Figura 5 muestra un ejemplo de dicha traza.

```
****************************************
Proceso1  Dir.Lógica: 5000      Dir. Física: 904
Proceso2  Dir. Lógica: 2048      Dir. Física: 1024
Proceso3  Dir. Lógica: 2148      Dir. Física: 2148
Proceso1  Dir. Lógica: 0      Dir. Física: 3072
Proceso2  Dir. Lógica: 7999      Dir. Física: 4927
Proceso3  Dir. Lógica: 50000      Dir. Física: 5968
Proceso1  Dir. Lógica: 100      Dir. Física: 3172
Proceso2  Dir. Lógica: 2248      Dir. Física: 1224
Proceso3  Dir. Lógica: 1024      Dir. Física: 6144
Proceso1  Dir. Lógica: 0      Dir. Física: 3072
Proceso2  Dir. Lógica: 4096      Dir. Física: 7168
Proceso3  Dir. Lógica: 1024      Dir. Física: 6144
Proceso1  Dir. Lógica: 456      Dir. Física: 3528
Proceso2  Dir. Lógica: 2134      Dir. Física: 1110
Proceso3  Dir. Lógica: 3477      Dir. Física: 405
Proceso1  Dir. Lógica: 4000      Dir. Física: 2976
Proceso2  Dir. Lógica: 3100      Dir. Física: 4124
Proceso3  Dir. Lógica: 6000      Dir. Física: 6000
Proceso1  Dir. Lógica: 1050      Dir. Física: 7194
**********************************
```

Figura 5. Ventana de direcciones físicas

Tras finalizar la simulación y descargar los procesos de la memoria, se pueden obtener estadísticas de la ejecución de la simulación. La Figura 6 muestra un ejemplo de dichos resultados, los cuáles se pueden obtener utilizando las opciones "Tasa de Fallos" y "Rendimiento" dentro del menú "Estadísticas".

```
**********************************************
Tasa de Fallos
**********************************************
El Proceso Proceso3 ha efectuado 7 accesos a memoria
obteniendo 1 aciertos y 6 fallos de página
Con una tasa de fallos del  85  %

El Proceso Proceso2 ha efectuado 7 accesos a memoria
obteniendo 2 aciertos y 5 fallos de página
Con una tasa de fallos del  71  %

El Proceso Proceso1 ha efectuado 7 accesos a memoria
obteniendo 3 aciertos y 4 fallos de página
Con una tasa de fallos del  57  %
**********************************************
Rendimiento del sistema
**********************************************
Se han producido un total de 21 accesos a memoria
contabilizando un total de 6 aciertos de página y
15 fallos de página. Con una tasa de fallos del  71  %
```

Figura 6. Estadísticas de la simulación.

3. Aplicación del simulador a las prácticas de Sistemas Operativos

En la sesión práctica relacionada con la gestión de memoria, se sugiere al alumno que siga los pasos mencionados en el apartado anterior. A partir de un fichero de simulación, el alumno debe configurar el sistema de varias formas interpretando los resultados obtenidos. El alumno debe realizar las siguientes tareas.

- Cargar el fichero de simulación.
- Configurar el sistema. Se sugiere el uso de un tamaño de memoria de 8 KB, un tamaño de página de 1 KB y alguno de los algoritmos de reemplazo global (LRU o FIFO). Respecto el planificador, se puede considerar al principio, una única cola (cola 0).
- Comenzar la simulación. Para un mejor control, se puede utilizar un modo "paso a paso" y la visualización simultánea de los resultados en la "Ventana de Direcciones Físicas". Se sugiere al usuario que antes de acceder a la siguiente dirección lógica, analice la información disponible y determine el proceso y si provocará un fallo de página, comprobando posteriormente qué página ha sido reemplazada y la dirección física generada.
- Evaluar los resultados de la simulación. Una vez finalizada la simulación se trata de comprobar los resultados obtenidos que reflejan la tasa de fallos de cada proceso y el rendimiento global del sistema.

Con la información obtenida se plantean ejercicios relacionados con la práctica realizada, y cuyo objetivo es analizar el nivel de adquisición de conocimientos del alumno. A continuación se muestra un posible ejercicio:

- Determinar el estado final de la memoria según una configuración de tamaño de página de 1 KB, algoritmo LRU de reemplazo GLOBAL y tamaño de memoria de 8 KB. Para especificar la solución, se le ofrece al alumno la tabla adjunta.

Marco	Pid	Pag.
0		
1		
2		
3		
4		
5		
6		
7		

Tabla 1. Ejemplo de ejercicios basados en tablas.

Aunque dicho ejercicio, es relativamente "mecánico" de utilización del simulador, también se sugiere al alumno que previamente se realice de forma completamente manual, de forma que el alumno pueda contrastar su conocimiento de la materia, utilizando el simulador para solucionar posibles dudas. También pueden plantearse actividades de tipo reflexivo que permitan validad el nivel de conocimientos y el trabajo del alumno.

A continuación se muestran algunos ejemplos de estos ejercicios propuestos al alumno.

- Indique si existe alguna diferencia en la aplicación de ambos algoritmos (LRU y FIFO) y en qué consiste. Justifique la existencia de una misma tasa de fallos de páginas en ambos ejemplos.
- Encuentre un ejemplo de carga donde la aplicación de un algoritmo LRU devuelva un menor número de fallos de página, respecto un algoritmo FIFO.

4. Conclusiones

En este trabajo, se ha presentado una herramienta dirigida a la simulación de algoritmos de gestión de memoria. Dicha herramienta ha sido realizada en lenguaje Java, lo que facilita su utilización en entornos Web y su portabilidad a diferentes sistemas operativos. Asimismo, proporciona una interfaz visual que favorece su uso por parte de los alumnos. Tras tres años de experiencia los alumnos han aceptado de buen grado y manifestado la utilidad de dicho simulador como herramienta de autoaprendizaje.

La herramienta, además de reproducir información en un formato atractivo, invita al alumno a interactuar con la aplicación y permite fácilmente la modificación de algunos de los parámetros de entrada como el tamaño de la página o de la memoria principal. La herramienta es de gran utilidad para el alumno principalmente para la comprobación de ejercicios planteados y resueltos en clase. La experiencia, en líneas generales, ha resultado positiva a pesar del elevado coste de desarrollar una herramienta *ad hoc* de uso específico.

Como trabajo en vías de realización, se está considerando la integración de:

- Algoritmos de planificación basados en prioridades.
- Aspectos teóricos en el funcionamiento del simulador
- Algoritmos de reemplazo de páginas alternativos.

Referencias

[1] A.B. Tucker et al., ACM/IEEE-CS Joint Curriculum Task Force. Computing Curricula 1991, ACM Press; IEEE Comp. Soc. Press. , 1991.

[2] ACM/IEEE Task Force on the Year 2001 Model Curricula for Computing , Computing Curricula 2001(CC-2001), obtenido el 10/11/1999 de la página web: http://www.computer.org/education/cc2001/.

[3] David Jones, Andrew Newman, A constructivist-based tools for operating systems education, Proceedings of EdMedia'2002, Denver, Colorado, June 2002.

[4] W A Christopher et a. (1993), The Nachos Instructional Operating System. Proceedings of the Winter 1993 Usenix Technical Conference, pp 481-489.

[5] Jeff Magee and Jeff Kramer. Concurrency: State Models & Java Programs.

[6] Llopis Mengual, J.Espinosa Rodilla. Simulador de gestión de memoria. Proyecto final de carrera (PFC1), 1999.

[7] Innovations for Education in Information Technology through Multimedia and Communication Networks. Proyecto Socrates de Red Temática. http://www.eui.upv.es/ineit-mucon/.

[8] Programa de Simulación de Gestión de memoria. http://www.eui.upv.es/ineit-mucon/ Applets /mem_simulator/Applet.htm.

Tecnologías de la información en la gestión empresarial

Gestión de Clientes en el marco de los Portales Corporativos. Ensayo de enseñanza interdepartamental en la EUEEZ. Integración de las visiones empresarial y tecnológica.

Javier Gutiérrez
E.U..Estudios Empresariales
Dept. de Informática e Ingeniería de Sistemas
Universidad de Zaragoza
e-mail: adsogu@posta.unizar.es

María Jesús Lapeña
E.U..Estudios Empresariales
Dept. de Informática e Ingeniería de Sistemas
Universidad de Zaragoza
e-mail: mlape@posta.unizar.es

Pilar Urquizu
E.U..Estudios Empresariales
Dpto. de Econocmía y dirección de Empresa
Universidad de Zaragoza
e-mail: purquizu@posta.unizar.es

Resumen

En la Escuela Universitaria de Estudios Empresariales de Zaragoza estamos llevando a cabo una experiencia interdepartamental para la didáctica integrada de aspectos empresariales y tecnológicos en relación a la gestión de clientes en el marco de los portales corporativos. Esta experiencia, se basa en un caso común, de índole práctica, planteado en cada asignatura implicada desde su punto de vista específico. Estos puntos de vista van desde el análisis y estrategias de segmentación de mercado, a la implementación de dichas estrategias en forma de bases de datos relacionales y consultas SQL, que finalmente se integran en un portal de empresa basado en tecnologías de Internet.

1. Gestión de clientes en el marco de los Portales corporativos

Los Portales Corporativos [3] o Portales de Empresa están adquiriendo una importancia fundamental como vertebradores del sistema informático de las empresas [1 y 4], al unificar el acceso a información, aplicaciones, trabajo cooperativo y todo tipo de recursos y servicios. Esta visión es la consecuencia final de la evolución de intranets y extranets, redes privadas virtuales, sedes web y sistemas de información tradicionales [3]. El entorno más adecuado para el desarrollo de estos Portales lo proporciona sin duda la tecnología procedente de Internet.

Como entorno unificado, en estos portales se integran diversos aspectos de gestión empresarial y toma de decisiones. No sólo proporcionan el marco de trabajo unificado, sino además la posibilidad de trabajar de forma integrada con información procedente de diversos departamentos o secciones de la empresa.

El caso de la gestión de clientes no es una excepción. Desde el emergente campo del llamado CRM (Customer Relational Management) surgen propuestas y puntos de vista de mayor alcance de los hasta ahora considerados en el mundo del marketing. La tecnología disponible hoy en día es responsable, en buena parte, de esta evolución, al permitir una gestión mucho más depurada de la que se venía haciendo hasta este momento y el acceso unificado a información de clientes procedente de distintos departamentos.

Debido a este estrecho maridaje entre marketing y tecnología, creemos fundamental que el alumno de empresariales conozca no sólo los aspectos de análisis de mercados sino también posibles tecnologías subyacentes. En este artículo presentamos una experiencia didáctica basada en el seguimiento de un caso supuesto. Este caso es extremadamente sencillo, tanto desde el punto de vista del marketing como desde el punto de vista informático. La pretensión con esta experiencia es que el alumno pueda acceder a distintos puntos de vista sobre un mismo problema: el punto de vista del marketing relacional, el de las bases de datos y el de la programación de portales corporativos.

Este artículo se estructura como sigue. En primer lugar se plantea el problema de la gestión

de clientes en el marco de los portales corporativos.

En el segundo apartado presentamos la didáctica de esta gestión en nuestra escuela, didáctica que desarrollamos mediante un caso de segmentación de mercado, de forma integrada desde dos departamentos y cuatro asignaturas. En este capítulo se destaca además la importancia de integrar conocimientos conceptuales del dominio de la empresa con conocimientos tecnológicos de Sistemas de Información e Internet. Los apartados tercero, cuarto y quinto presentan las diferentes asignaturas implicadas en esta experiencia, así como la relevancia del problema en cada una de las asignatura y particularmente cómo se enmarca en ellas el caso de ejemplo. En el apartado sexto hacemos una valoración de nuestra aproximación didáctica y presentamos nuestras ideas sobre futuras actuaciones en didáctica integrada.

2. Presentación del caso de ejemplo

En nuestra didáctica integrada suponemos la existencia de una empresa mediana dedicada a la venta de material para deportes de aventura y de naturaleza, y a la organización de actividades guiadas en ese ámbito.

Esta empresa tiene un portal corporativo donde se integra un módulo de toma de decisiones y en particular un modulo de gestión de clientes y segmentación de mercado. Nuestro interés en este momento se centra en agrupar a los clientes de nuestro negocio en dos segmentos, con el objetivo de difundir selectivamente información sobre actividades guiadas. Nos referiremos a estos dos segmentos como "clientes de turismo activo" y "clientes de deporte de riesgo".

Para realizar esta segmentación tendremos en cuenta diversas características de nuestros clientes, tales como las actividades en las que hayan participado previamente, o el tipo de material que hayan adquirido en nuestra tienda. Para ello utilizamos información procedente de diversos departamentos de la empresa: tienda, actividades y gestión de clientes (esta integración de información procedente de actividades diversas de la empresa es una de la características fundamentales, si no la más importante, de los portales corporativos.)

El alumno deberá establecer rangos de valores para las variable determinantes de los segmentos de mercado, tras haber comprendido por qué se han escogido esas variables y no otras. Esto lo realizará en el ámbito de las asignaturas del Departamento de Economía y Dirección de Empresa. En las asignaturas del Departamento de Informática e Ingeniería de Sistemas realizará una implementación de estas reglas. En la asignatura Informática II el alumno creará una base de datos adecuada para recoger la información de clientes, productos y actividades, así como las sentencias SQL y formularios necesarios para implementar las reglas y la asignación de valores a las variables. Finalmente, en la asignatura de Lenguajes de Programación integrará esta base de datos como parte del Portal Corporativo, permitiendo que cualquier usuario autorizado del Portal pueda ejecutar las consultas correspondientes a las reglas y en su caso modificar las propias reglas o sus variables.

3. Gestión de clientes (visión empresarial)

En la docencia impartida desde el Departamento de Economía y Dirección de Empresas tienen una relevancia particular los aspectos relacionados con la aplicación de las nuevas tecnologías al marketing, fundamentalmente porque permiten profundizar en el conocimiento y tratamiento de la información sobre los clientes para realizar una gestión más depurada.

Estos conocimientos se desarrollan fundamentalmente en dos asignaturas: *Procesos Básicos de Producción* y *Análisis de Mercados*. En este marco se concede especial importancia a la Gestión Relacionál de Clientes (CRM, Customer Relational Management) o Marketing Relacional. En este sentido resulta especialmente importante conocer cómo realizar una adecuada segmentación de mercado. El propósito de esta segmentación es realizar un marketing selectivo que fidelice a los clientes y permita aumentar la cuota de mercado.

Hay que destacar la importancia de definir correctamente el problema y comprender la necesidad de la tecnología para implementar segmentaciones adecuadas que permitan personalizar las estrategias de Marketing y crear relaciones a largo plazo con los clientes.

El alumno deberá comprender, guiado por el profesor, cuáles son las variables a tener en cuenta para realizar dicha segmentación, y determinar qué intervalos de valores son los que permitirán realizar dicha segmentación de forma adecuada.

A continuación mostraremos un ejemplo de uno de los posibles resultados de dicha segmentación. A este resultado se ha llegado después de un análisis en el que se han determinado cuáles son las variables que es preciso tener en cuenta (propuesto por el profesor) y posteriormente qué rango de valores va a ser el que nos proporcione la segmentación deseada. Análisis más completos y realistas darán resultados más complejos. Hay que tener en cuenta que se trata de un ejemplo muy sencillo, debido sobre todo a que se pretende que el alumno pueda encontrar el hilo de continuidad de su análisis con su confección informática como una base de datos en un portal corporativo.

Determinación del segmento de "clientes de turismo activo"

- Clientes que han contratado cualquier actividad comprendida en las modalidades de : *"Excursión ecuestre", "Itinerario botánico", "Senderismo"*
- Clientes que han adquirido productos de alguna de estas secciones: *"Senderismo"*

Determinación del segmento de "clientes de deporte de riesgo"

- Clientes que han contratado cualquier actividad comprendida en las modalidades de: *"Escalada", "Esquí de montaña", "Expedición"*
- Clientes que han adquirido productos de alguna de estas secciones: *"Escalada", "Esquí"*

4. Bases de datos (nivel tecnológico)

En la asignatura Informática II abordaremos todo lo concerniente al diseño y desarrollo de bases de datos referente al caso que nos ocupa. De hecho, esta asignatura está centrada en el estudio de las bases de datos, ya que es obvia su importancia en el mundo de la empresa actual.

La asignatura tiene una parte teórica, en la que se explican conceptos y nociones fundamentales y se tratan todos los aspectos teóricos relacionados con el análisis y diseño de bases de datos. Tiene también una parte práctica, que se desarrolla en paralelo a la parte teórica, que tiene como objetivo concreto la implementación de bases de datos en Access. Aunque de forma simplificada y siempre en un nivel básico acorde a nuestras necesidades y posibilidades, se abordará cada una de las fases del ciclo de vida de las bases de datos: análisis, diseño, desarrollo, pruebas y mantenimiento. Se insistirá siempre en la importancia de seguir un método, y en la necesidad de hacer un control de calidad y generar una correcta documentación.

El enfoque que damos al estudio de las bases de datos viene dado por el entorno en el que se encuadra la asignatura: la diplomatura en empresariales; por ello, los ejemplos y casos que se desarrollan corresponden al entorno empresarial, tratando de buscar el equilibrio entre problemas clásicos de gestión y problemas emergentes como CRM.

En el caso del ejemplo (Gestión de Clientes en el marco de los Portales Corporativos) los alumnos construirán una base de datos y almacenarán los datos correspondientes a clientes, productos y actividades en sendas tablas, con la siguiente estructura (a modo de ejemplo):

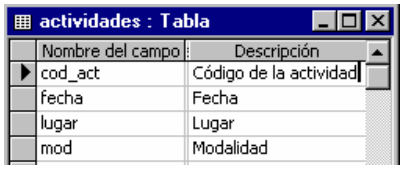
clientes : Tabla

Nombre del campo	Descripción
cod_cli	Código del cliente
apel1	Primer apellido
apel2	Segundo apellido
nom	Nombre
correo_e	Correo electrónico
tel	Teléfono
dir	Dirección
edad	edad

productos : Tabla

Nombre del campo	Descripción
cod_prod	Código del producto
denom	Denominación
marca	Marca
seccion	Sección

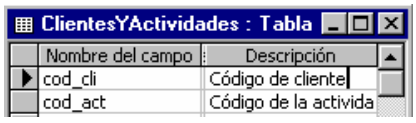
Los posibles valores de la sección son:, "Senderismo", "Escalada" y "Esquí".

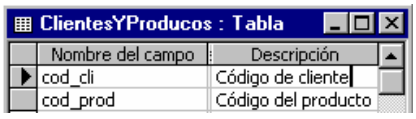


Los posibles valores de la modalidad son: "Excursión ecuestre", "Itinerario botánico", "Senderismo", "Escalada", "Esquí de montaña" y "Expedición".

El diseño de estas tablas y los datos a considerar será el primer problema a resolver por los alumnos, una vez hayan decidido las variables a tener en cuenta para llevar a cabo la segmentación de mercado.

Tienen que construir, además, otras dos tablas que representan las relaciones de los clientes con las actividades y con los productos. El alumno creará las restricciones de clave foránea necesarias.





Asimismo, los alumnos deberán diseñar e implementar consultas y formularios sobre dichas tablas, construyendo soluciones integradas para la gestión de clientes. Como tema colateral, a lo largo de todo el desarrollo, se insistirá particularmente en considerar con especial interés el tema de la interfaz de usuario, aplicando normas de estilo que nos lleven a diseños de calidad.

A modo de ejemplo, la implementación como SQL de la primera de las cuatro reglas del apartado anterior sería así:

```
SELECT clientes.*
FROM    clientes    INNER    JOIN
(actividades      INNER      JOIN
ClientesYActividades            ON
actividades.cod_act             =
ClientesYActividades.cod_act)   ON
clientes.cod_cli                =
ClientesYActividades.cod_cli
WHERE
(((actividades.[mod])="Excursión
ecuestre"                       Or
(actividades.[mod])="Itinerario
botánico"                       Or
(actividades.[mod])="Senderismo")
);
```

Una vez diseñadas las consultas, se construirán formularios como éste que nos permitan seleccionar los datos de los clientes requeridos:

Hay que decir además que el desarrollo de esta base de datos en el entorno de la asignatura Informática II va más allá de lo estrictamente necesario para la experiencia interdepartamental del caso ejemplo. Así, todas las consultas y formularios se integrarán en una completa aplicación de gestión de clientes, para cuya construcción los alumnos han de poner en práctica todos los conocimientos que han ido adquiriendo en la asignatura.

5. Programación de Portales (nivel tecnológico)

En la asignatura de Lenguajes de Programación se estudia el desarrollo de portales como herramienta integradora. Se trata de una asignatura predominantemente práctica, en la que el desarrollo de un portal justifica la exposición teórica de los temas propuestos para la asignatura.

En una primera unidad didáctica se enmarca el problema de la programación y la creación de portales en el ámbito de la empresa, justificando las distintas tecnologías estudiadas desde su uso para marketing, gestión de decisiones, venta on-line, recopilación de información de clientes y trabajo cooperativo. En sucesivas unidades se hace una introducción tecnológica a Internet, Buscadores Web, codificación HTML y uso de editores web. Como complemento a estos conceptos se introducen nociones de multimedia y estilo de diseño. Otro capítulo importante del programa consiste en la presentación de una Metodología de Desarrollo de Sedes Web, que resulta de la simplificación de las principales Metodologías de Sistemas de Información utilizadas hoy en día.

La parte más directamente implicada en el problema de la programación en entornos de red trata sobre arquitecturas cliente servidor y la programación en ambos lados, ofreciéndose conocimientos básicos de JavaScript para la programación del lado cliente. En cuanto a la programación de lado servidor se centra la atención en el problema de las Bases de Datos, orientándose hacia la tienda virtual y las bases de datos de clientes y de control de productos. Para ello se estudian formularios web y nociones muy básicas de servidores web y ASP.

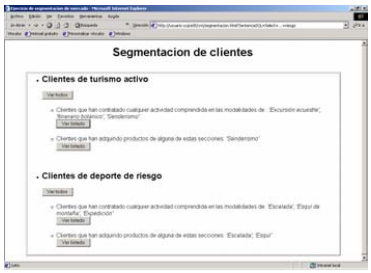
La práctica del curso consiste en el desarrollo guiado mediante especificaciones detalladas, de una Sede Web en sentido muy amplio, que puede ser concebida como un pequeño portal. Esta Sede está inspirada en un caso real de negocio virtual de gran resonancia nacional, *barrabes.com*. En esta Sede o portal se desarrollan varios módulos, funcionando a un tiempo como Intranet, Extranet y Sede Web. Se concede gran importancia al problema del acceso a bases de datos a través de Internet.

En este contexto el caso práctico que presentamos en este artículo aparece como una práctica más de las realizadas durante el curso, consistiendo en el trabajo con los elementos necesarios parta realizar el acceso: creación de formularios web, comprensión de un pequeño script ASP que permite la consulta a la base de datos y finalmente la colocación de la base de datos Access en el servidor web (IIS). A partir de este momento el alumno puede efectuar las consultas preparadas, a través de una página web integrada en el portal.

En el desarrollo de esta práctica el alumno desarrollará una página HTML con formularios, uno para cada consulta deseada. Cada uno de estos formularios tiene un campo oculto, con una sentencia SQL. A modo de ejemplo presentamos el formulario correspondiente a la sentencia SQL mostrada en el capítulo anterior

```
<FORM      action="http://usuario-
yujce0t/crm/consulta.asp"
method="post">
 Clientes que han contratado
 cualquier actividad comprendida
 en las modalidades de:
 <I>'Excursión ecuestre',
 'Itinerario
  botánico',Senderismo' </I> <BR>
 <input type="hidden"
 name="SentenciaSQL"
 value="SELECT  clientes.* FROM
 (actividades INNER JOIN
 ClientesYActividades ON
 actividades.cod_act =
 ClientesYActividades.cod_act)
 INNER JOIN clientes ON
 ClientesYActividades.cod_cli =
 clientes.cod_cli WHERE
 (((actividades.mod) = 'Excursión
 ecuestre')) Or
 (((actividades.mod) =
 'Itinerario botánico')) Or
 (((actividades.mod) =
 'Senderismo'))">
 <input type="submit" value="Ver
        listado">
</FORM>
```

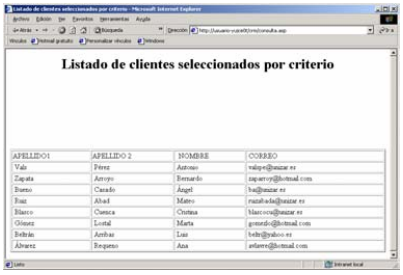
Las sentencias SQL utilizadas para construir los formularios de esta página serán las que el alumno ha desarrollado en la asignatura Informática II. El aspecto de la página Web que debe desarrollar es el siguiente:



Los formularios de esta página llaman a un script ASP que establece una conexión con la base de datos Access. El script recoge la sentencia SQL que le ha enviado el formulario Web y la envía a la base de datos. Con el resultado de la consulta, el script genera una página Web que es devuelta al cliente. La base de datos consultada es la que el alumno ha creado en la asignatura Informática II. No se le pide al alumno que programe este script, sino sencillamente que lo comprenda y sepa realizar alguna pequeña modificación en él:

```
<HTML><HEAD><TITLE>Listado de
clientes seleccionados por
criterio</TITLE></HEAD><BODY>
<%
Dim laSentencia

laSentencia =
Request.Form("SentenciaSQL")

Set conexion =
CreateObject("ADODB.Connection")
conexion.Open
"PROVIDER=MICROSOFT.JET.OLEDB.4.0
; DATA SOURCE=
C:\WebShare\Wwwroot\crm \crm.mdb"
Set misDatos =
conexion.Execute(laSentencia)
Response.Write
("<CENTER><H1>Listado de clientes
seleccionados por
criterio</h1><table BORDER=2
WIDTH='100%'> </CENTER>")
Response.Write("<tr><td>" & "
APELLIDO1  " & "</td><td>" & "
APELLIDO 2 "  & "<td> " & "
NOMBRE " & "</td>" & "<td> "
& "CORREO" & "</td></tr>")

while (not misDatos.Eof)

Response.Write("<tr><td> " &
misDatos("apel1") &
"</td><td> " &
misDatos("apel2") & "<td> "
& misDatos("nom") & " </td>"
& "<td> " &
misDatos("e_mail") &
" </td></tr><br>")
      misDatos.MoveNext
wend

Response.Write ("</TABLE>")

conexion.Close
Set conexion = Nothing
%>
</BODY></HTML>
```

De esta forma el alumno ha utilizado las sentencias SQL que ya había realizado, insertándolas en formularios Web para consultar, gracias al uso de un script sencillo, a la base de datos también creada previamente. El resultado final para cada una de las consultas es una página Web, generada por el propio script. Esta página contendrá los datos de los clientes seleccionados, como podemos ver en la imagen:



6. Conclusiones

Pensamos que la experiencia didáctica que presentamos en este artículo supone una efectiva

innovación didáctica, basada fundamentalmente en las siguientes ideas:

- Se trata de una **experiencia interdepartamental e interasignatura**, que permite mostrar al alumnos la cohesión e interdependencia de diversas asignaturas de su currículo.
- Se trata de una **experiencia internivel**, que permite al alumno comprender la continuidad entre el nivel teórico de los análisis de mercado y el nivel de implementación informática. Se transmite así al alumno la idea de que los conocimientos tecnológicos que adquiere en la diplomatura están cohesionados con la formación teórica a la que dan soporte; en el caso que nos ocupa, mostrando la necesidad de las tecnologías de la información para desarrollar las estrategias de Marketing.
- Asimismo nuestra orientación sigue **la línea de las metodologías docentes del caso** [5], al presentar al alumno un problema equivalente a un problema real típico que en distintas prácticas debe ir resolviendo bajo el asesoramiento y orientación del profesor. Creemos que se trata de una perspectiva adecuada, ya que, como se indica en [5], "para enseñar el uso de una ciencia es conveniente comenzar por casos en que al educando le resulte relativamente fácil estructurar el problema en términos de las variables de esa ciencia. lo que significa utilizar casos más bien próximos a problemas teóricos"
- Finalmente, se trata de una experiencia inspirada en un caso real, el de una empresa prototípica en nuestra comunidad, barrabes.com, que ha evolucionado desde un pequeño negocio familiar en un pueblo de montaña ("donde se acaba la carretera") hasta llegar a ser una empresa líder en comercio electrónico a nivel nacional, e incluso a ser parte constituyente de la moderna plataforma de e-business Walqa [2].
- Creemos que esto supone un poderoso incentivo didáctico para el alumno de empresariales de nuestra comunidad

No obstante, esta experiencia se encuentra todavía en pleno desarrollo en este curso 2002 / 2003, lo que nos impide efectuar una evaluación y un análisis restrospectivo de su efectividad. Se trata en todos los casos de asignaturas anuales. La respuesta, tanto a nivel de profesorado como de alumnos está siendo muy positiva; los alumnos agradecen la presentación de casos que les acerquen a su futura realidad profesional y que requieren la puesta en práctica de conocimientos correspondientes a materias de distinta índole. Hemos previsto efectuar una encuesta a final de curso, que nos permita medir el nivel de satisfacción de alumnos y profesores.

Creemos que se trata de una verdadera innovación docente en los Estudios Empresariales, que, en general, siempre han adolecido de formación en Informática; desconocemos la existencia de experiencias en esta línea, que presenta grandes perspectivas y posibilidades docentes y que requiere, por tanto, un impulso para su desarrollo.

En cuanto a líneas futuras de actuación integrada en la Escuela de Empresariales de Zaragoza, hemos de destacar que en el nuevo Plan de Estudios pendiente de aprobación, está prevista una línea de especialización que, bajo el título "Marketing y Nuevas Tecnologías", pretende englobar contenidos de Marketing y de Informática con objeto de ofrecer al alumno una formación que le capacite para la realización de análisis de clientes, fidelización de clientes y comercialización de productos a través de la red. Concretamente, en dicha especialidad se integran las asignaturas de Informática "Bases de Datos y Sistemas de Información" y "Nuevas Tecnologías de la Información".

Referencias

[1] Aguila, Ana Rosa del, *Comercio electrónico y estrategia empresarial*, Madrid, Rama, 2000

[2] Arnal, José Carlos, *Sueños Electrónicos*, Zaragoza, Institución Fernando el Católico, 2002

[3] Joyanes Aguilar, Luis, *Portales de 3ª generación*, en DATA.TI, Diciembre 2002, p. 54 58

[4] O'Brien, James A., *Sistemas de información gerencial*, McGraw Hill, 1999

[5] Orti González, Ana María, Metodologías para la Formación de Emprendedores: *El Método del Estudio del Caso, el Método de las Situaciones y el Estudio de Incidentes Críticos*,http://www.unsam.edu.ar/unsam/secyt/DVE2001/index.htm, 2001

Telemática

Experiencia en la aplicación de un entorno virtual como apoyo a la docencia de laboratorios presenciales

Javier Macías, José Javier Martínez, José María Gutiérrez, Roberto Barchino
Dpto. Ciencias de la Computación
Universidad de Alcalá
28871 Alcalá de Henares
e-mail: {javier.macias, josej.martinez, josem.gutierrez, roberto.barchino}@uah.es

Resumen

Se presenta una experiencia de aplicación de un entorno virtual en un laboratorio presencial, indicando las razones de tal selección, la configuración del entorno, su explotación y los resultados obtenidos.

1. Introducción

Existe en la actualidad un número significativo de Universidades españolas que utilizan entornos virtuales, como por ejemplo la UNED [2], la UOC [4], la Universidad de las Islas Baleares [11] o el consorcio ADA-Madrid [10] (en el cual participan seis universidades madrileñas).

Sin embargo, la mayoría de las veces estos entornos sólo se explotan en asignaturas impartidas a distancia o de una forma semipresencial, ya sea por tratarse de una universidad a distancia, como es el caso de la UNED o la UOC, por la especial casuística de los alumnos a los que van destinadas, como ocurre en la Universidad de las Islas Baleares debido a los problemas de desplazamiento que provoca su insularidad, o por la propia naturaleza de las asignaturas, como es el caso de las asignaturas ofertadas por ADA-Madrid, que son de libre elección y su objetivo es ampliar el horizonte académico de los alumnos.

Además, debido a la importancia estratégica que supone el uso de las nuevas tecnologías de la información y la comunicación [6], la puesta en marcha de los entornos virtuales se ha hecho en ciertos casos más como inversión pensando en el futuro que con vistas a una explotación inmediata eficiente.

Por otro lado, los nuevos entornos producen desconfianza en muchos profesores, al tratarse de una herramienta para la enseñanza que nunca experimentaron como alumnos [9].

A todo ello debemos añadirle el hecho de la juventud en que todavía se encuentran las plataformas de soporte de estos entornos virtuales y el proceso de desarrollo en que todavía se encuentran los estándares que permitan su interconexión y, por tanto, respalden las inversiones realizadas en la producción de material educativo. En cualquier caso, es de destacar la cada vez mayor convergencia de funcionalidades de estos entornos y la tendencia a un agrupamiento de plataformas [13].

2. Objetivos

Aprovechado la disponibilidad en el Departamento de algunos de los entornos virtuales más utilizados y los conocimientos y experiencia de los autores en dichos entornos, se decidió la puesta en marcha en el curso 2001/02 de uno de ellos con los siguientes objetivos:

- El entorno se emplearía en una asignatura de carácter troncal u obligatorio.
- La asignatura debería tener unas características tales que el uso del entorno produjera unas ganancias significativas en uno o más aspectos docentes.
- El entorno debería ser integrador, en el sentido de ser capaz de reunir la mayor cantidad de herramientas de apoyo a la docencia basadas en las tecnologías de la información y la comunicación, incluidas aquellas ya utilizadas en cursos anteriores si procediera

(como el correo electrónico o la transferencia de ficheros).

- El esfuerzo requerido tanto al profesor como al alumno para el aprendizaje y uso del entorno tendría que ser razonable.
- Aproximadamente a mitad de curso, se realizaría una evaluación del entorno virtual, tanto para poder realizar una valoración del funcionamiento del mismo como para tomar las medidas correctoras oportunas en caso de que se detectaran problemas que hubieran pasado desapercibidos.

3. Elección de la asignatura y la plataforma

Una vez establecidos los objetivos, las primeras acciones a desarrollar consistían en la elección de la asignatura y la plataforma.

Como asignatura se seleccionó "Laboratorio de Metodología de la Programación", impartida en el segundo cuatrimestre a los alumnos de primer curso de Ingeniería Técnica Informática de Sistemas e Ingeniería Técnica Informática de Gestión de la Universidad de Alcalá. Dicha asignatura es de carácter obligatorio y tiene una carga lectiva de tres créditos.

Las razones por las cuales se eligió una asignatura de laboratorio de programación fueron las siguientes [5, 12]:

- Distribución de enunciados. Los enunciados suelen variar total o parcialmente cada año, entre otras razones para evitar que el alumno copie las prácticas entregadas en cursos anteriores. Frecuentemente los enunciados van acompañados de código que el alumno debe modificar o utilizar y/o de ficheros con datos de prueba. Asimismo, es reseñable que en el caso de enunciados de nueva creación, existe mayor posibilidad de que, durante el desarrollo de la práctica por parte de los alumnos, se descubran errores de diverso tipo que pasaron desapercibidos al profesor. Todo ello hace difícil la edición de libros de prácticas y justifica el disponer de una plataforma que permita una comunicación ágil entre profesores y alumnos.
- Entrega y corrección de prácticas. Usualmente, los alumnos deben entregar una o más prácticas en las fechas que se establezcan. El profesor, tras recoger las prácticas, deberá realizar acciones como la corrección manual o automática, el uso de aplicaciones detectoras de plagio y la comunicación a los alumnos del resultado. Todo ello genera una cantidad importante de trabajo, parte del cual se puede calificar como trabajo administrativo.
- Problemática de horarios. Los laboratorios suelen ser semanales, de unas dos horas de duración (como ocurre con la asignatura seleccionada). En ciertos casos, los horarios de laboratorios no son adyacentes a los de las clases teóricas, lo que puede dificultar la asistencia de los alumnos, en particular de aquellos que también trabajan. Esto ocasiona un número de faltas mayor al esperable.
- Deficiencias en infraestructura. Uno de los objetivos generales de todo laboratorio suele ser el trabajo en equipo. Sin embargo, la mayoría de las veces los laboratorios sólo disponen de un ordenador por equipo de alumnos, lo cual supone que dos o tres personas estarán trabajando sobre el mismo ordenador en el momento de codificar. Este hecho, además de ser contrario a la propia naturaleza de la programación modular, ocasiona un aprovechamiento deficiente del tiempo de clase, por lo que no suele ser suficiente para terminar las prácticas. Como consecuencia, frecuentemente los alumnos deben continuar trabajando en casa, dificultándose la comunicación entre los componentes del grupo y con el propio profesor.
- Demanda de tutorías. Aunque, al igual que para las tutorías de clases teóricas, existe el problema común de posible coincidencia de éstas con el horario de otras clases, en el caso de los laboratorios el problema se acentúa, ya que las tutorías para laboratorios suelen ser más frecuentadas que las tutorías para clases teóricas. Las razones más obvias para esta mayor utilización vienen dadas por un menor contacto con el profesor de prácticas en horario de clase (una vez por semana) y por la entrega periódica de prácticas, que marcan frecuentemente los ritmos y picos de

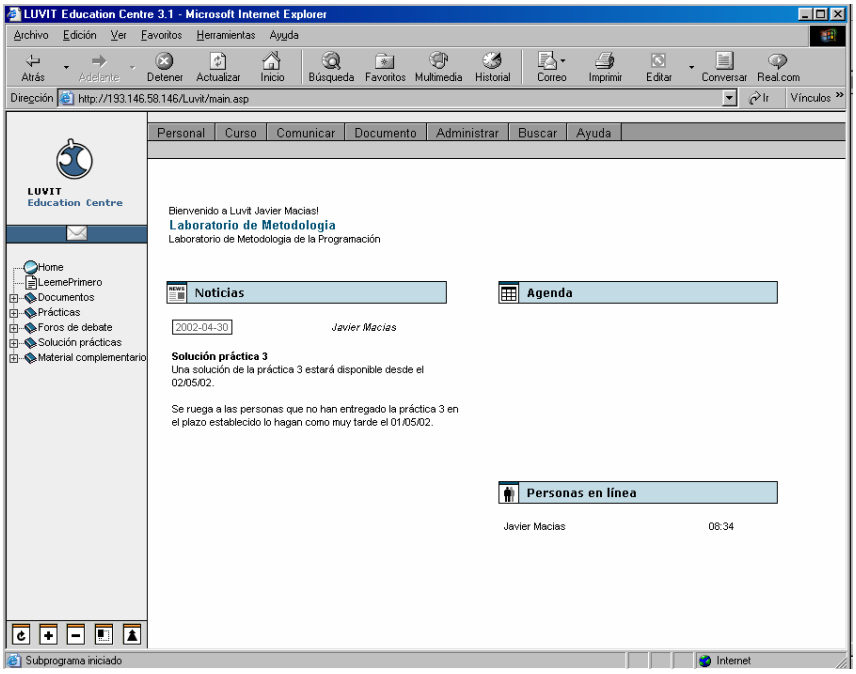


Figura 1. Página inicial del curso Luvit de Laboratorio de Metodología

utilización de dichas tutorías. En particular, los alumnos demandan una rápida respuesta a sus cuestiones sobre programación, ya que una sola duda o problema puede ocasionar que una práctica completa no funcione, o, incluso, que no puedan ni llegar a la codificación de una sola línea por dificultades relacionadas con el análisis o el diseño.

Los alumnos de la asignatura seleccionada estaban divididos en ocho grupos por titulación de 30 a 35 alumnos por grupo, dado el tamaño físico y número de máquinas de los Laboratorios. Se decidió incluir en el entorno a sólo dos grupos (denominados A2 y A5), por lo que el total de alumnos incluidos en la experiencia era de aproximadamente 65. Para simplificar la gestión del entorno y mantener una mayor animación en los foros, se acordó incluir a ambos grupos en el mismo entorno virtual. Además, bastó un solo profesor para atender el curso durante su explotación.

Respecto a la elección de la plataforma se buscó, en consonancia con los objetivos perseguidos, una de fácil uso y con disponibilidad inmediata. Ello nos llevó a descartar tanto cualquier diseño propio a medida (aun usando componentes de alto nivel) como soluciones externas que pudieran requerir una instalación compleja, necesitaran adaptación o no dispusieran del soporte técnico adecuado. Por tanto, nos decidimos por utilizar una de las plataformas disponibles en el mercado de calidad contrastable.

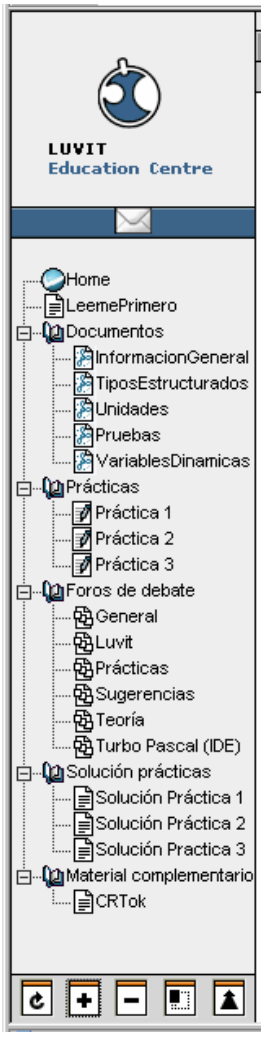


Figura 2. Árbol de navegación de Luvit

En concreto, elegimos Luvit® [7], debido a su disponibilidad en el Departamento, la experiencia que ya poseíamos con ella y a su claridad (Figura 1) y facilidad de uso, fundamentada en el árbol de navegación (Figura 2) y similar al árbol del Explorador de Windows®. Una comparativa de Luvit con otras plataformas se encuentra en [3].

4. Fases

A continuación se detallan las fases desarrolladas en la experiencia aquí presentada.

4.1. Preparación del curso

La preparación del curso exige cierto trabajo, aunque parte de él puede ser aprovechado en los siguientes años.

En primer lugar, y suponiendo que la plataforma que dará soporte a los cursos está ya instalada, hay que dar de alta (crear) el curso, indicando entre otros datos el nombre del curso, el del profesor y las fechas de inicio y fin del mismo.

Seguidamente se procede a dar la estructura inical del curso. Se inserta un mensaje de bienvenida y un documento de introducción al curso (denominado generalmente como "Leeme Primero" o de alguna otra forma similar) y se crean las secciones principales, incluyendo los distintos foros y un mensaje inicial en cada uno de ellos indicativo del cometido del mismo. Aunque la plataforma lo permitía, se decidió no crear una sección específica para charlas planificadas (*chat*), ya que al tratarse de una herramienta síncrona, podría generar una cantidad de trabajo no asumible, al menos no el primer año.

Posteriormente, se cargan de alumnos en el curso. En Luvit los alumnos se identifican mediante una cuenta de correo electrónico (todos tienen al menos una proporcionada por la Universidad), siendo obligatorio también el nombre del alumno. Para obtener estos datos, los alumnos escribieron durante una clase su correo y nombre en un mismo fichero texto uno a uno (bastaron unos pocos minutos para que todos los alumnos pasaran por el ordenador donde debían introducir sus datos). Este fichero fue cargado fácilmente en Luvit tras un pequeño preproceso.

4.2. Presentación del curso y puesta en marcha

En una de las clases de Laboratorio, se realiza una breve presentación del curso a los alumnos, ya que debido a que se trata de alumnos de informática y que la asignatura pertenece al segundo cuatrimestre, se presupone que tienen suficiente destreza para manejar el entorno. Se les explica la forma de acceso, cómo recibirán la contraseña de entrada (en nuestro caso se decidió enviarla por correo electrónico, cosa que puede hacer automáticamente Luvit) y se les instó a que consultaran en primer lugar el documento "LeemePrimero" del curso donde se les daban las instrucciones necesarias para su manejo. Además, se les insistió en que podían recurrir a las tutorías presenciales para explicarles el manejo del entorno a aquellos que tuvieran más problemas.

Se sugirió a los alumnos que formaran parejas en las que al menos uno de los componentes tenga acceso a Internet desde su casa, aunque también podían acceder al entorno desde las aulas de informática de la Universidad.

Tras la clase, se mandaron las contraseñas a los alumnos para que quien así quisiera pudiera utilizar el entorno desde el primer día.

4.3. Explotación

Una vez enviadas las contraseñas, se procede a la explotación del curso. Regularmente se accede al entorno para contestar las preguntas del foro y del correo, añadir nuevo material (como documentos, prácticas y soluciones), corregir las prácticas y comunicar noticias de interés para los alumnos (fechas de disponibilidad de documentos, enunciados y soluciones, fechas de entrega, fechas de exámenes y revisiones, etc.). También se mantuvo alguna charla interactiva con ellos cuando se coincidió en línea (en concreto, el número de charlas en las que participó el profesor fueron cuatro).

Es fundamental que el tiempo de respuesta a las cuestiones planteadas por los alumnos sea mínimo, idealmente inferior a 48 horas. Esta recomendación fue seguida fielmente durante el desarrollo del curso. Para lograr estos tiempos de respuesta, además de atender el curso durante el tiempo de tutoría presencial en los momentos en que no asistían alumnos, también se aprovechaban otros instantes en que el profesor se conectaba a Internet por alguna otra razón, ya fuera dentro o fuera de la Universidad. La carga de trabajo que ello supuso no fue excesiva en ningún momento, debido a que los mensajes de los alumnos se encontraban bastante repartidos a lo largo del cuatrimestre.

Como anécdota resaltar que un problema eléctrico acaecido en mitad de las vacaciones de Semana Santa hizo que se perdiera el acceso al curso. La caída se detectó y solucionó rápidamente (en menos de 48 horas) y se mandó un correo electrónico a los alumnos para indicarles por qué no habían podido acceder e informándoles del restablecimiento del curso.

4.4. Evaluación

Cuando el curso llevaba funcionando aproximadamente dos meses, se realizó una evaluación tal y como estaba prevista. Para ello se recopilaron datos objetivos obtenidos directamente del entorno virtual (número de mensajes en el foro, correos enviados, etc.). Asimismo, se pasó una encuesta anónima en papel a los alumnos durante una de las clases [8].

5. Secciones

Conviene realizar un repaso, siquiera somero, de las distintas secciones que se crearon en el curso y de su cometido. Para quién desee ver más a fondo el curso, se ha habilitado un usuario en el mismo [1].

- *LeemePrimero*. Documento en HTML que contiene las instrucciones básicas para el curso.
- Documentos. En esta sección se fueron añadiendo los distintos documentos con la teoría complementaria necesaria para poder realizar las prácticas. Asimismo se incluyeron las soluciones a los exámenes de junio y septiembre justo después de terminar éstos. Todos los documentos se proporcionaban en formato PDF.
- Prácticas. En esta sección el alumno accedía a los enunciados de las prácticas, junto con los archivos de código o prueba cuando

procedía. También desde aquí entregaba las prácticas y recibía las correcciones.

- Foros. Los foros son una herramienta muy importante, ya que constituyen la forma básica de realización de las tutorías remotas y de comunicación y ayuda entre los propios alumnos, con la ventaja de que una respuesta a una cuestión planteada por un alumno puede ser aprovechada por el resto de compañeros. Se crearon 6 foros, denominados *General* (para cuestiones generales), *Luvit* (para cuestiones relativas a la plataforma), *Prácticas* (para cuestiones relativas a las prácticas que había que entregar), *Sugerencias* (para plantear opiniones, críticas, sugerencias, etc.), *Teoría* (para cuestiones teóricas) y *Turbo Pascal (IDE)* (para cuestiones relativas al entorno de desarrollo utilizado en la asignatura).
- Solución prácticas. En esta sección se publicaban las soluciones propuestas para las prácticas, una vez terminado el plazo de entrega, para que el alumno pudiera compararla con su solución.
- Calificaciones. Esta sección se añadió tras el examen de junio para publicar las notas de los alumnos.
- Material complementario. Esta sección fue necesaria crearla durante el desarrollo del curso para incluir material que no tenía cabida en ninguna de las secciones anteriores. Por ejemplo, en esta sección se colocó un fichero que solucionaba ciertos problemas que surgían al ejecutar en ordenadores rápidos los programas compilados mediante el IDE utilizado en la asignatura.

6. Evaluación y datos finales

En el momento de la evaluación del curso se estimó que el número de alumnos que seguían regularmente la asignatura, ateniéndonos al criterio de haber tenido tres o menos faltas de asistencia, era de 21 en el grupo A2 y 24 alumnos en el grupo A5. Es de resaltar que en el recuento del grupo A5 se ha incluido a un alumno que, aunque no pudo asistir ningún día a clase debido a una operación que le imposibilitaba el andar, siguió el curso a través del entorno virtual y entregó sus prácticas regularmente.

Las medidas objetivas obtenidas en ese momento del entorno virtual, centradas principalmente en el uso de los foros, se muestran en la Tabla 1. Es de destacar que muchas de las personas que habían enviado mensajes (en concreto, cerca de un 60%), lo habían hecho más de una vez, por lo que pudimos deducir que su satisfacción por la respuesta obtenida fue alta, ya que de otra forma no hubieran vuelto a usarlos. Como dato particular, el alumno que no podía asistir había enviado tres mensajes hasta ese momento.

Descripción	Valor
Mensajes de alumnos a los foros	26
Alumnos que enviaron mensajes a los foros	14
Alumnos que enviaron más de un mensaje a foros	8
Máximo número de mensajes enviados a los foros por un mismo alumno	4
Correos recibidos por el profesor	3

Tabla 1. Medidas objetivas del curso en el momento de la evaluación

Respecto a la encuesta realizada, fue contestada por 31 alumnos, 18 de los cuales pertenecen al grupo A2 y 13 al grupo A5. Se debe subrayar que de los 31 alumnos, tan sólo uno indicó no haber utilizado la herramienta Luvit por problemas de conexión (problemas que posteriormente solucionó), quedando así excluido como individuo estadístico a analizar. Conviene decir que dicho alumno formaba pareja para las prácticas con otro alumno que sí se conectaba, por lo que en todo momento pudo obtener los documentos y enviar las prácticas.

Aunque el número de alumnos que ha contestado la encuesta no permite realizar un análisis estadístico determinante, sí que permite apreciar tendencias. En particular, cabe destacar los siguientes resultados:

- A la mayoría de los encuestados les ha parecido que la utilidad de Luvit en la asignatura "Laboratorio de Medodología de la Programación" era alta o muy alta. Sólo un alumno contestó que la utilidad le parecía baja, si bien se debe notar que este alumno

reseñó que tenía frecuentemente problemas al conectar con Luvit. Además, a la mayoría también les pareció útil la idea de extender el uso de la herramienta a otras asignaturas de laboratorio, aunque el porcentaje descendió hasta la mitad cuando se les preguntó por su extensión a asignaturas de teoría.

- A la mayoría les resultó sencillo aprender su manejo, aunque a dos alumnos les resultó muy difícil o difícil. Más de la mitad consideraron que no era necesario o que era poco necesario el haber utilizado una clase para enseñar su manejo. Además, todos consideran que, tras haber aprendido, no tienen dificultades para utilizarlo.
- Valoran positivamente la asincronicidad que la herramienta les proporciona a la hora de realizar consultas, descargar documentación y para la entrega de prácticas. Es también de destacar que más de la mitad de los encuestados califica de positiva la iniciativa de establecer una hora en la semana en la cual poder mantener charlas con los compañeros y el profesor.
- Dos terceras partes de los encuestados habían hecho uso de los foros, mientras que sólo un quinto de ellos habían utilizado las tutorías presenciales. Dos de las razones aportadas por los alumnos del porqué no utilizaban las tutorías presenciales fueron los horarios de las mismas y que les era más cómodo resolver dudas específicas utilizando los foros. Es importante señalar que casi todos los que hicieron uso de los foros habían encontrado útil alguna respuesta a preguntas planteadas por otros compañeros.
- Como era esperable, los alumnos que más favorablemente valoran la herramienta son en general los que disponen de conexión a Internet desde su casa, en particular los tres que disponen de ADSL. Seis de los alumnos no disponen de conexión desde su casa, aunque afirmaron que no tenían problemas para acceder ya que usaban las aulas de la Escuela Politécnica, aunque se quejan de su saturación. También expresaron la dificultad de recordar el nombre de la página de entrada a Luvit por lo que, para solucionar el problema, se creó un enlace directo a Luvit desde la página web del Departamento.

A final de curso se realizó una medida final resumen sobre algunos de los aspectos principales del curso (Tabla 2). Cabe remarcar, teniendo en cuenta el bajo tiempo de respuesta por parte del profesor, que en tres ocasiones los alumnos contestaran a preguntas de sus compañeros.

Descripción	Valor
Alumnos que entraron al menos una vez	45
Documentos publicados por el profesor	9
Enunciados	5
Soluciones propuestas por el profesor	5
Material complementario	3
Otros documentos	3
Noticias	20
Correos recibidos por el profesor	9
Charlas en las que participó el profesor	4
Mensajes enviados al foro por alumnos	35
Mensajes enviados al foro por el profesor	35
Mensajes de alumnos en los foros en respuesta a mensajes de otros compañeros	3

Tabla 2. Medidas finales del curso

Dada la especial importancia de los foros, se realizó un recuento detallado de los mensajes publicados en los mismos (Tabla 3). Como era de suponer, el mayor número de mensajes se concentró en el foro de prácticas. Es de destacar que el foro de teoría, aunque sólo tuvo tres mensajes de alumnos, éstos fueron de gran calidad, y, posiblemente, no hubieran sido planteados de no disponerse de este curso.

Foro	Mensajes		
	De alumnos	Del profesor	TOTAL
General	2	2	4
Luvit	4	2	6
Prácticas	24	21	45
Sugerencias	1	2	3
Teoría	3	5	8
Turbo Pascal	1	3	4
TOTAL	35	35	70

Tabla 3. Medidas detalladas de los foros

7. Conclusión

La experiencia aquí presentada muestra que la utilización de un entorno virtual para apoyar la docencia de laboratorios presenciales es bien recibida por los alumnos y ampliamente utilizada. Esta gran utilización es achacable en parte a que las funcionalidades proporcionadas por este tipo de herramientas se adaptan especialmente bien a los requisitos exigidos por las asignaturas de laboratorio.

El alumno valora positivamente la accesibilidad que esta herramienta le proporciona, tanto en lo que respecta al material como al profesor, incrementándose el número de dudas planteadas y resueltas. En concreto, prefiere usar los foros del entorno virtual para dudas puntuales y recurrir a las tutorías presenciales sólo cuando se trata de dudas generales.

Por otro lado, el trabajo que el uso del entorno virtual supone al profesor no es excesivo, en particular si no se hace uso extensivo de las herramientas síncronas, de las cuales la charla es su mayor exponente. Es de destacar que parte del trabajo ocasionado por el curso, en concreto, el mantenimiento de los foros, revierte debido a que una misma respuesta puede solucionar una duda común a un número significativo de alumnos. Además, es posible reutilizar la estructura del curso y posiblemente gran parte del material para años siguientes.

8. Futuras líneas de trabajo

Tras el éxito inicial obtenido, el siguiente paso consiste en ampliar la experiencia a todos los grupos de laboratorio de la asignatura. Ello implicará, entre otras cuestiones, la participación de más profesores, que deberán ser formados para que puedan obtener un rendimiento satisfactorio de la plataforma.

Posteriormente, y tras una valoración de los resultados y conocimientos obtenidos de esta experiencia a mayor escala, se podría ampliar la utilización de los cursos virtuales al resto de laboratorios relacionados con la programación u otros laboratorios o asignaturas cuyas características maximicen las ganancias del uso de este tipo de herramientas.

Referencias

[1] *Curso "Laboratorio de Metodología de la Programación" (plataforma Luvit)*. Dpto. Ciencias de la Computación, Universidad de Alcalá. http://193.146.58.146/Luvit/entrance /entrance.asp?cid=17 (usuario: invitado@jenui2003.es, contraseña: invitado).

[2] *Cursos virtuales de la UNED*. UNED. http://virtual0.uned.es/

[3] *Evaluation of Learning Management Systems*. University of Fribourg. http://www.edutech.ch/edutech/tools/ev2.php

[4] *La Universidad Virtual. Campus Virtual*. UOC. http://www.uoc.edu/web/esp /launiversidad /comoseestudia/campus.htm

[5] Labra, J., Morales, H. y Turrado, R. *Plataforma de enseñanza de lenguajes de programación a través de Internet: Proyecto IDEFIX*. Actas de las VIII Jornadas de Enseñanza Universitaria de la Informática (JENUI, 2002).

[6] *Ley Orgánica de Universidades*. Boletín Oficial de las Cortes Generales. Número 45-13, 26 de diciembre de 2001.

[7] *Luvit (página de inicio de la web de Luvit en inglés)*. Luvit. http://www.luvit.com /P00.m4n?language=en

[8] Macías, J. *Apéndice del curso Luvit*. Dpto. Ciencias de la Computación, Universidad de Alcalá, 2002. http://www.cc.uah.es/jmacias /jenui2003/apendice.pdf

[9] McKeachie, W. *Teaching tips*. Houghton Mifflin, 1999.

[10] *¿Qué es ADA-Madrid?*. Comunidad de Madrid. http://adamadrid.uc3m.es /campusvirtual/Idioma1 /proyectoadamadrid.htm

[11] *Presentación de Campus Extens*. Universidad de las Islas Baleares. http://campusextens.uib.es:2200/portal/pages /cast/ques.htm

[12] Rodríguez, J. *Gestión Automática de entrega de Prácticas vía web*. Actas de las VIII Jornadas de Enseñanza Universitaria de la Informática (JENUI, 2002).

[13] Ruipérez, G. *Las plataformas de gestión de aprendizaje (PGA)*. UNED, 2002. http://www.uned.es/euva/euvaflash/contenidos /articuloruiperez.htm

Desarrollo de actividades en grupos coordinados sobre el modelado y simulación del proceso de transmisión de datos

José Oliver, Alberto Bonastre, José L. Poza
Dpto. de Informática de Sistemas y Computadoras
Universidad Politécnica de Valencia 46022 Valencia
e-mail: {joliver, abonastre, jopolu}@disca.upv.es

Resumen

En este artículo se presenta la realización de actividades en grupo como metodología docente, y los resultados obtenidos dentro del marco de una asignatura sobre sistemas de transmisión de datos.

Durante el desarrollo de estas actividades se simula el proceso de transmisión de datos punto a punto a nivel físico. Cada grupo se centra en realizar el modelado de un elemento del sistema, que puede estar a nivel de codificación de datos, de modulación o de medio físico. Para realizar las actividades, los grupos deben coordinarse entre ellos, tanto para resolver problemas comunes (comunicación horizontal), como para definir claramente el interfaz para el paso de información que circula entre niveles contiguos (comunicación vertical). Por último, se plantea una serie de itinerarios en los que varios grupos de alumnos se agrupan formando grupos mayores y realizan una prueba global de todo el sistema.

1. Introducción

La visión clásica del proceso enseñanza-aprendizaje se basa en clases magistrales apoyadas por el enunciado y resolución de ejercicios en clase. Este planteamiento ofrece poca interacción profesor-alumno e invita escasamente a la participación del alumnado, que suele adoptar una posición de comodidad y paulatina pasividad, lo que resulta en un menor índice de éxito en el aprendizaje.

En 1988, la Universidad Politécnica de Valencia puso en marcha el Plan de Innovación Educativa (PIE). En aquel momento representaba una propuesta atrevida y avanzada encaminada a incentivar las mejoras del sistema enseñanza-aprendizaje en la docencia. Este nuevo plan y la reforma de los planes de estudio de Informática en 1996 aumentaron considerablemente el número de sesiones de docencia en laboratorio, donde el alumno resuelve problemas más reales y maneja instrumentación relacionada con la materia.

Doce años después, la Universidad Politécnica de Valencia propone un nuevo proyecto que recoge el testigo del antiguo PIE, adaptándose a las nuevas necesidades de la Universidad moderna. Este plan se denomina Proyecto EUROPA (Una Enseñanza ORientada al APrendizaje) [4] [5] y toma del PIE parte de sus objetivos, ampliándolos en algunos casos y complementándolos con nuevas propuestas en otros. En concreto se propone la inclusión de técnicas de aprendizaje innovadoras, como es el uso de seminarios en clase, con debates, mesas redondas y foros, que facilitan el aprendizaje mediante la participación interactiva del alumnado, y otras técnicas como las actividades, en las que los alumnos forman grupos de trabajo para realizar diversos ejercicios.

En este artículo presentamos las actividades propuestas para la asignatura optativa Sistemas de Transmisión de Datos, y el resultado obtenido por parte de los alumnos. Siguiendo el espíritu del proyecto EUROPA, nuestro objetivo no ha sido únicamente el desarrollo de habilidades en el marco de la temática de la asignatura, sino que otras capacidades no curriculares, como el trabajo en grupo y la capacidad de coordinación y liderazgo, también han sido potenciadas.

2. Sistemas de Transmisión de Datos

La asignatura Sistemas de Transmisión de Datos (STD) [1] se centra en los primeros niveles del modelo ISO/OSI (nivel físico y enlace de datos),

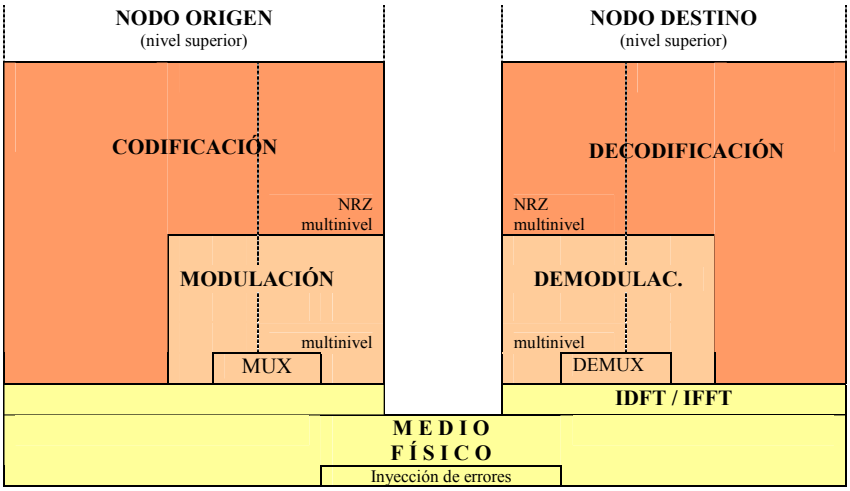


Figura 1. Proceso de comunicación punto a punto a modelar

profundizando en las técnicas de transmisión existentes.

Es una asignatura optativa, correspondiente a los planes de estudios de 1.996, para la obtención de los títulos de Ingeniero en Informática (5º curso, con 6 créditos) e Ingeniero Técnico en Informática de Sistemas /de Gestión (3º curso, con 4.5 créditos).

La asignatura se desarrolla en dos bloques temáticos diferenciados, cuyos temas son los expuestos a continuación.

Bloque I. Fundamentos de Transmisión de Datos.
1. Sistemas Teleinformáticos
2. Medios Físicos de Transmisión
3. Señal. Fundamentos teóricos
4. Modems
5. Transmisión serie

Bloque II. Sistemas de transmisión
6. ADSL-Modem cable
7. Tarjetas de Red. Ethernet
8. Transmisiones inalámbricas

La división en dos bloques se realiza para separar los fundamentos teóricos de las principales aplicaciones reales. Esta división proporciona dos visiones de los sistemas de transmisión de datos, una teórica donde se adquiere la base para comprender la segunda visión, la práctica y real.

3. Actividades propuestas

Las actividades consisten en la realización obligatoria por parte de grupos de alumnos de trabajos teórico/prácticos relativos a la asignatura. Dichos trabajos son tutorizados por el profesor de prácticas, y expuestos durante las últimas sesiones de teoría.

Los contenidos de dichos trabajos tienen una parte teórica (investigación bibliográfica) y otra práctica (implementación, simulación y evaluación de distintos componentes del nivel físico), y es necesaria la interacción entre los grupos tanto en su desarrollo como para la comprobación del funcionamiento adecuado de todo el sistema.

3.1. Coordinación de los grupos

Cada grupo de actividades está formado por tres componentes. En un grupo, cada uno de los

integrantes debe asumir una de las siguientes responsabilidades de coordinación:

- Responsable de coordinación interna del grupo. Se encarga de planificar los objetivos según etapas, y de que estos plazos se cumplan en la medida de lo posible.
- Responsable de coordinación externa. Es la persona responsable de comunicación con los otros grupos, tanto para resolver dudas y necesidades comunes, como para realizar las pruebas del sistema final.
- Responsable de coordinación con el profesor. Su labor es resolver las dudas que tienen, tanto los componentes del grupo hacia el profesor, como el profesor hacia los alumnos, manteniendo en todo momento una realimentación para que el profesor conozca la marcha de las actividades y los principales problemas observados.

Las responsabilidades que se plantean no son más que *roles* o papeles que se invita al alumno a interpretar. El objetivo es acercar el funcionamiento de las actividades al trabajo en grupo y la coordinación entre grupos que se exige en el mundo laboral.

3.2. Temática de los trabajos

Las actividades se centran en el área del temario situada dentro del Bloque I de la asignatura, por lo que una vez estudiados los primeros temas, el alumno ya se encuentra preparado para desarrollar las actividades. La segunda parte de la asignatura se cubrirá por medio de seminarios y estudios de caso. La distribución de créditos de las nuevas metodologías introducidas en este artículo, comparadas con las clásicas sesiones de teoría y prácticas, puede consultarse en [4]. En concreto se pretende realizar un entorno de simulación de los sistemas de transmisión punto a punto a nivel físico. Los trabajos que se realizan son parte de un entorno global, aunque se centran en un aspecto concreto del sistema.

La parte teórico de las actividades consiste en que el alumno se documente sobre el componente a implementar, sepa en qué consiste, su posible construcción hardware, cómo se puede simular su comportamiento, etc. La parte práctica del trabajo consiste en la implementación de código que, a partir de la entrada de un archivo de datos, los transforme de manera adecuada, de forma que según el nivel al que se esté trabajando, los codifique, module o simule su transmisión por un medio físico, volcando el resultado en otro archivo de datos.

El esquema general de la comunicación punto a punto que se pretende modelar es el indicado en la figura 1, en el que los datos se transmiten de un nodo origen a otro destino. Estos datos se inyectan en la red codificados, y posiblemente modulados. Otro aspecto que se pretende que el sistema pueda manejar es la multiplexación de datos. Para esto se deberá tener en cuenta que varias fuentes de datos, modulados a distintas frecuencias, se puedan transmitir un mismo medio físico mediante multiplexación en frecuencia.

Posteriormente, los datos procesados por el nodo origen, una vez codificados y modulados, se inyectan físicamente en la red. Para caracterizar el comportamiento del medio físico es necesario tener una representación espectral de los datos, para lo que se utiliza la transformada discreta de Fourier. A parte de los típicos efectos introducidos por el medio de transmisión (atenuación, distorsión por retardo de grupo, etc) se pueden inyectar errores en la red; algunos afectarán a su comportamiento en frecuencia (por ejemplo, un típico ruido introducido por la red eléctrica funcionando a 50hz) y otros serán puntuales en el tiempo (ruidos impulsivos, posibles ruidos blancos, etc.). Por último, la información tiene que ser procesada correctamente para que se recupere en el nodo destino, por lo que los alumnos también deberán implementar los procesos inversos de demultiplexación, demodulación y decodificación.

En la figura 2 se puede observar el conjunto de trabajos que se proponen, situados cada uno en su nivel correspondiente. Esta figura no es más que un desglose de la figura 1, detallando los posibles trabajos que se puede implementar en cada nivel.

3.3. Implementación de las actividades

A la hora de poder coordinar unas actividades de estas características resulta fundamental especificar el formato de los datos que las aplicaciones generan y reciben. En nuestras actividades utilizamos un formato básico muy

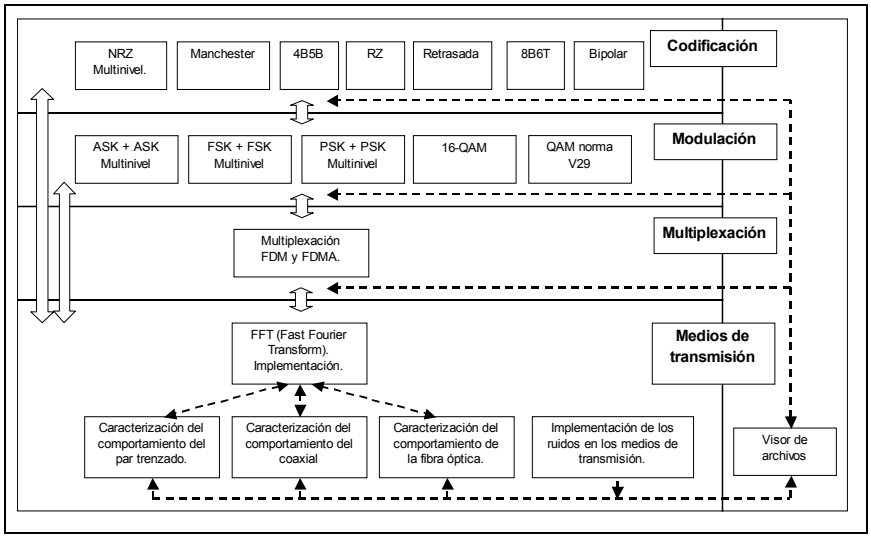


Figura 2. Conjunto de actividades

simple. Representamos una señal como un conjunto de niveles de tensión resultantes de muestrear esa señal en cada instante de muestreo. El resto de parámetros específicos de cada nivel se introducen por teclado cuando se ejecuta dicho módulo.

Un fichero que representa a una señal tiene como primera componente la frecuencia de muestreo usada, después se indica el número de muestras obtenidas y por último se encuentra el vector de muestras, representada cada una de ellas por un valor en coma flotante en simple precisión. Con este sencillo formato se puede representar, a partir de los valores de tensión en una serie de instantes discretos o muestras de la señal, el estado aproximado de una señal en todo momento. A su vez, el hecho de tratar la señal de manera discreta (o discretizada) nos permite un procesamiento digital que simplifica su análisis y modelado.

Para la realización de las actividades los alumnos no necesitan partir de cero, ya que se les ofrece una serie de esqueletos de programa para cada nivel, que implementan tareas básicas como la lectura/escritura de ficheros de muestras, la petición de datos al usuario y la definición básica de cabeceras e interfaces.

Además, para facilitar la comprensión por parte del alumno de todo el sistema que se pretende modelar, y permitir encuadrar bien el trabajo que se está desarrollando, se les facilita un ejemplo compilado de un módulo por cada nivel, de forma que ellos pueden hacer pruebas con un sistema que se encuentra ya resuelto. Por otra parte también se dispone de un visor de datos que permite leer varios ficheros de muestras y mostrarlos en pantalla de manera simultánea, con lo que se pueden observar las transformaciones que va sufriendo la señal en cada uno de los módulos.

4. Comunicación entre grupos

Cuando el alumno finalice su formación deberá haber desarrollado ciertas habilidades de trabajo en grupo, comunicación y coordinación, así como la capacidad de asumir su *rol* dentro del grupo de trabajo de la empresa. Por ello, se ha planteado dar un paso más en las actividades por medio de la

coordinación dentro del propio grupo e intergrupal.

La novedad de incluir una acción formativa de trabajo en grupo no es única ni innovadora, sin embargo sí que se ha considerado como novedad el hecho de que estos grupos deban comunicarse entre ellos, tanto a nivel vertical como a nivel horizontal [3]:

- *Comunicación vertical.* En el proyecto de actividades expuesto anteriormente, los grupos escogen un área en la que trabajar: codificación, modulación o medios de transmisión. Entre estos niveles o capas de las actividades deberá existir una comunicación para consensuar el formato de los archivos y paso de datos.
- *Comunicación horizontal* Gran parte del código que deba desarrollar cada grupo, así como de la documentación, será común a todos los grupos que trabajen dentro de la misma capa. Por ello, entre los grupos deberá existir una comunicación para no repetir trabajo y repartirse labores comunes.

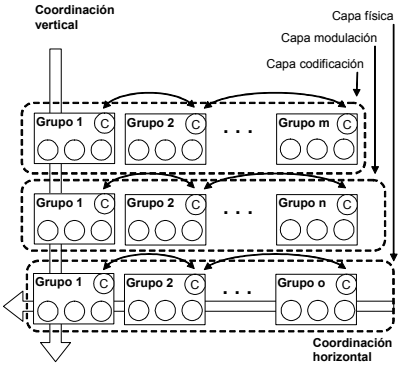


Figura 3. Relaciones de comunicación entre los diversos grupos de las actividades.

Con este sistema se pretende estudiar la interactuación entre grupos cuando el trabajo de unos depende de otros [2]. De esta forma la comunicación entre coordinadores, tal y como se ve en la figura 3, será muy importante. Dicha comunicación se llevará a cabo durante reuniones periódicas.

5. Resultados

La experiencia propuesta se ha llevado a cabo durante dos cursos diferentes. En un primer año tan sólo se desarrollaron los trabajos propuestos, sin llegar a hacer una prueba global de todo el sistema. De esta manera, sólo hubo ocasión de realizar coordinación horizontal, mediante el apoyo entre grupos que trabajaban en proyectos similares pertenecientes al mismo nivel. En un segundo año se pudo poner en marcha unas sesiones adicionales para realizar labores de coordinación vertical, intentando integrar todos los trabajos de manera que el objetivo final de colaboración entre proyectos a distintos niveles quedara totalmente cubierto.

5.1. Comunicación horizontal: resultados por niveles

Al iniciar el desarrollo del trabajo, cada grupo se centra en documentarse e implementar una actividad concreta. En esta primera fase, la comunicación predominante será la horizontal, ya que distintos grupos que trabajan al mismo nivel se encuentran con problemas similares, que podrán resolver de manera coordinada. Aun así la comunicación vertical seguirá presente, ya que hay ciertos parámetros del interfaz de la aplicación que dependerán de cómo estén implementando su actividad los grupos de otros niveles.

Los grupos que estén trabajando en la capa de codificación son los que inician el proceso de la simulación. Deben tomar los datos a partir de una secuencia de bits y realizar su representación según la codificación a aplicar. Para esto se hace un muestreo de la señal resultante a partir de cada uno de los valores binarios de la fuente de datos. Sabemos que la duración de un bit dependerá directamente de la velocidad de transmisión. Por otra parte, el número de muestras por bits, una vez establecida la duración del bit, dependerá del periodo de muestreo, determinado por la frecuencia de muestreo. Por tanto, el problema de obtener una secuencia de muestras a partir de un flujo de bits es común para todos los grupos que trabajen a nivel de codificación, y podrá resolverse de manera coordinada. Posteriormente, bastará con realizar una asignación distinta de los

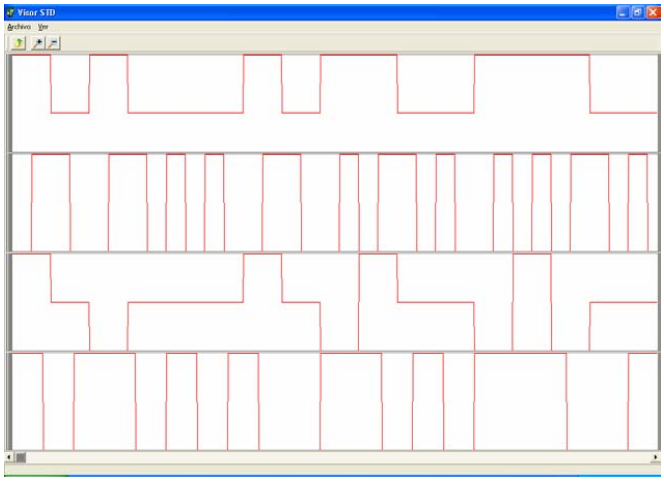


Figura 4. Resultado de distintas actividades en la capa de codificación.

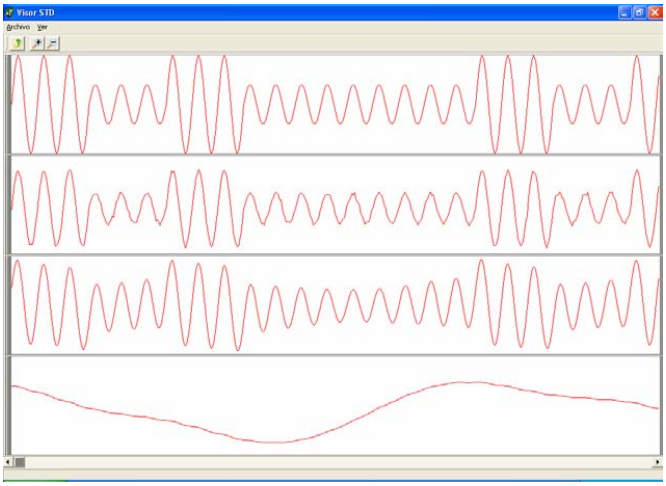


Figura 5. Resultado de las actividades en la capa de medio físico de transmisión.

niveles de tensión en función del valor del bit y el tipo de codificación implementada.

En la figura 4 tenemos el resultado de codificar el mismo flujo de bits utilizando distintas codificaciones implementadas por los grupos de actividades. La primera de todas, situada en la parte superior de la pantalla del visor, se corresponde con una sencilla NRZ, de

manera que el grupo encargado de implementar este tipo de codificación simplemente debía asignar un nivel alto de tensión cuando el bit a codificar fuera un uno, y un nivel bajo en caso contrario. La siguiente codificación es una Manchester. En este caso, para codificar un bit a uno se debe especificar un flanco de subida, codificando la primera mitad del intervalo del bit como nivel bajo y la segunda como nivel alto. La tercera codificación se corresponde con una bipolar en la que un cero binario se implementa con un nivel de tensión de cero voltios, y los bits a uno iteran sucesivamente de un nivel de tensión determinado positivo +V a otro –V, balanceando de esta forma el nivel de continua. Por último se encuentra una codificación del tipo 4B5B, en la que la asignación de valores a las muestras ya no se realiza bit a bit de manera directa, sino que existe una asignación previa de 5 bits por cada medio byte de la fuente original. Una vez realizada esta conversión, el tipo de codificación será similar a la NRZ, pero utilizando los valores indicados en la tabla de conversión 4B5B.

Una vez resuelto el problema común de asignar valores muestras en función de los bits de entrada, implementar una u otra codificación resulta sencilla, de manera que los alumnos han aprendido como trabajar en grupo para resolver un problema les resulta más productivo que enfrentarse al mismo problema por separado.

5.2. Comunicación vertical: pruebas de itinerarios

Tal y como está planteada hasta ahora la experiencia resulta enriquecedora, ya que promueve el trabajo en equipo y entre equipos, pero sin embargo aun puede completarse un poco más. Con este objetivo, en un segundo año, se incluyó la necesidad de realizar pruebas conjuntas entre equipos que trabajan a distintos niveles, de manera que se potenciase aun más la coordinación entre grupos de distintas capas.

De esta forma, se plantean una serie de itinerarios que agrupan una actividad de cada nivel, de manera que, una vez terminado, depurado y entregado el trabajo a realizar por un grupo, se llevan a cabo una serie de pruebas conjuntas en las que cada grupo aporta su módulo. Así se observa cómo la señal se transforma en función del tipo de medio utilizado y otros factores, como la longitud del cable, su categoría, o el nivel de ruido térmico o impulsivo introducido en el medio. La figura 5 muestra un ejemplo de cómo la señal modulada (arriba) varía al sufrir un ligero ruido térmico (segunda señal), se atenúa ligeramente al utilizar un cable de categoría cinco de 500 metros (tercera señal), o casi por completo al utilizar un cable de inferior categoría y una distancia de 5000 metros. Hay que resaltar que la última señal ha tenido que ser ampliada y reescalada, ya que la mayor parte de frecuencias se encuentran atenuadas.

En este tipo de pruebas conjuntas resulta muy interesante una actividad concreta como es la multiplexación en frecuencia. En la figura 6 encontramos un ejemplo de itinerario en el que se ha aplicado multiplexación de dos fuentes de datos distintas, que han sido codificadas con NRZ y moduladas aplicando ASK OOK. Estas dos señales, que se encuentran representadas en la parte superior del visor, se multiplexan en un mismo medio siendo el espectro resultante el que se muestra en la tercera posición de esta figura. Una vez transmitida la señal por el medio se filtra en el nodo destino utilizando dos filtros paso-banda, de manera que se puede volver a recuperar ambas fuentes por separado. Algunas componentes frecuenciales se mezclan de forma inevitable, y por tanto las señales variarán ligeramente respecto a su representación en el nodo original, como se observa en las señales situadas en la parte inferior del visor.

Por otra parte, se pueden realizar pruebas de itinerarios más reales, como por ejemplo simular la transmisión de datos por un canal telefónico utilizando la norma v.21. Para realizar esta prueba disponemos de todos los ingredientes necesarios. A nivel de codificación y modulación, usamos un NRZ a 300 bps con modulación FSK, utilizando portadoras de 1080 hz con desviaciones de 100 hz a ambos extremos para el canal inferior, y otra portadora de 1750 hz con la misma desviación para el superior. Una vez obtenidas ambas señales de subida y bajada de datos, se utiliza el módulo de multiplexación para combinar ambas fuentes que, una vez multiplexadas, se pasan al módulo de medio físico de transmisión que modela el canal telefónico. Este ejemplo es mucho más próximo al mundo real y por esto el alumno se encuentra más motivado. Además, se les plantea ejemplos de empresas que, para realizar el diseño de sus

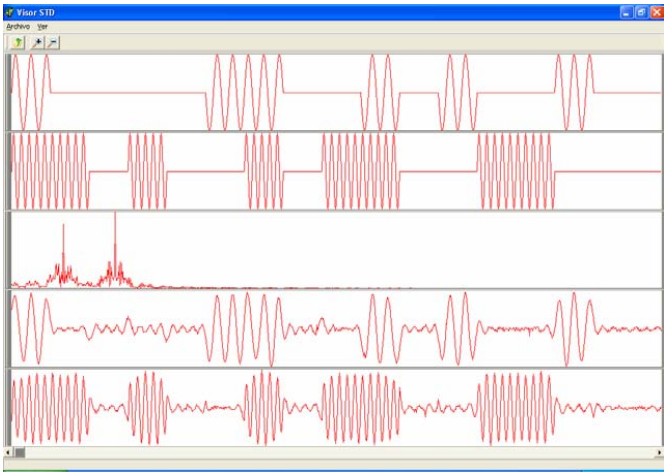


Figura 6. Resultado de un itinerario con multiplexación en frecuencia.

tarjetas de red, hacen estudios de comportamiento del medio similares a los que ellos llevan a cabo, aunque con un nivel de complejidad mayor.

Para realizar las pruebas de itinerarios se dedicaron dos sesiones prácticas de dos horas. En ellas los alumnos ya no sólo cooperaron entre los miembros de un mismo grupo, sino que realizaron las pruebas con personas que estaban trabajando con otros grupos de otros niveles, con lo que adquirieron una visión más global, adoptando distintos puntos de vista, y se favoreció la percepción de la importancia de la coordinación vertical para lograr el correcto funcionamiento del trabajo que se estaba realizando.

6. Conclusiones y trabajo futuro

En este artículo se ha presentado la aplicación de actividades como una nueva metodología docente en una asignatura sobre sistemas de transmisión de datos. Los resultados obtenidos por parte de los alumnos han sido satisfactorios, tanto en el desarrollo de los trabajos en sí, como en la mejora de habilidades para el trabajo en grupo.

Además, mediante una encuesta que se ha pasado a los alumnos, se refleja como el índice de satisfacción por parte de los alumnos es bastante alto, obteniendo una calificación final de 8,1, medio punto por encima del resultado obtenido al preguntarles por las metodologías docentes más clásicas, como son las sesiones de teoría o las prácticas en el laboratorio.

Como trabajo futuro, y a partir de las experiencias obtenidas durante este año, pretendemos potenciar desde el principio la coordinación entre grupos del mismo itinerario.

Referencias

[1] Alberto Bonastre Pina, Félix Buendía García, Manuel Pérez Malumbres. *Equipos y Sistemas de Transmisión de Datos*. SPUPV. 1994

[2] Contreras J.M. *Cómo trabajar en grupo*. Ed. San Pablo. 1997.

[3] Musitu, G. *Psicología de la comunicación*. Nau Llibres. 1987.

[4] José Luis Poza, Alberto Bonastre, Jose Oliver. *Aplicación de las directivas EUROPA en la asignatura Sistemas de Transmisión de Datos*. VIII Jornadas de Enseñanza Universitaria de la Informática, 2001

[5] Vicerrectorado de coordinación académica y alumnado. *Una enseñanza orientada al aprendizaje: proyecto EUROPA*. UPV. 2001

Demostraciones

Desarrollo de un simulador de programación del microprocesador Intel 8085

Manuel Rodríguez Álvarez, Ángel Manuel Gómez García,
Pedro Mesas García, José Ignacio Ruiz Núñez, Alberto Prieto.
Departamento de Arquitectura y Tecnología de Computadores.
Escuela Técnica Superior de Ingeniería Informática. Universidad de Granada
C/ Daniel Saucedo Aranda s/n. 18071 Granada
e-mail: mrodriguez@atc.ugr.es ; manolo@goliat.ugr.es

Resumen

El objeto de este trabajo es la concepción, desarrollo y realización de un simulador de microprocesador 8085 diseñado por Intel, al objeto de ser utilizado con fines docentes en la Universidad de Granada en diferentes asignaturas de las titulaciones de Ingeniero en Informática o Ingeniero Técnico en Informática de Gestión y Sistemas en la Escuela Técnica Superior de Ingeniería Informática o bien de Ingeniería Electrónica, Ingeniería Industrial e Ingeniería de Telecomunicaciones. La idea principal del mismo es disponer de un entorno amigable de simulación que permita al estudiante adquirir un profundo conocimiento de la programación de dicho microprocesador. Obviamente el uso de dicho simulador es libre y gratuito al objeto de facilitar su uso y difusión.

En este trabajo, después de una breve introducción pasaremos a describir la funcionalidad del simulador y su aspecto.

1. Introducción

Un simulador de un microprocesador es un programa que se comporta de igual forma que el microprocesador simulado.

El simulador, como cualquier otro programa, se apoya en el microprocesador de la maquina en la que se ejecuta, el cual no tiene porque parecerse en nada al microprocesador simulado y siendo en el caso que nos ocupa, el microprocesador de la máquina donde se ejecuta mucho más avanzado que el simulado. En nuestro caso, el simulador se comporta como un traductor que comprende un programa diseñado para 8085 y que genera las instrucciones necesarias para que el procesador sobre el que se ejecuta se comporte de forma similar a un 8085. La simulación trata de proveer la misma apariencia externa que el elemento que se simula. En cualquier simulación existen una serie de aspectos sobre los que nos centramos y otra serie de aspectos que obviamos. La simulación puede ser más amplia o menos según en la cantidad de aspectos en los que centre, y más o menos precisa según cuanto profundice en ellos. Obviamente una simulación más amplia y detallada, implicará una mayor complejidad del simulador.

Podemos centrar la simulación del 8085 a muchos niveles. Desde un nivel operativo, en el que el microprocesador realiza una determinada tarea, hasta un nivel electrónico, en el que se simula la evolución de todos los componentes electrónicos del procesador.

Dado que el objetivo de este trabajo es obtener un simulador didáctico para aprender a programar un 8085, sólo nos interesarán ciertos aspectos de la simulación, obviando el resto y centrando el simulador en la programación del 8085.

2. Funciones del simulador

El simulador del 8085 que se ha desarrollado, ha sido a nivel de programación. Por tanto, las unidades que se han simulado son las instrucciones, las cuales afectan a los registros, los bits de estado, la memoria y los puertos. También se ha incluido la opción de que el simulador disponga de dispositivos externos de salida (pantalla de texto y gráfica, panel de leds y visualizadores de 7 y 15 segmentos) y de entrada (panel de interruptores y un teclado externo). Esto lleva a que también se ha simulado cómo las instrucciones afectan a los dispositivos externos.

El desarrollo del trabajo comienza con el estudio de las diferentes posibilidades en cuanto al simulador, el cual se ha desarrollado para que se

pueda trabajar bajo un entorno de Windows 98 o Windows 2000 en un computador Personal Compatible.

Para llevar a cabo la interfaz con el usuario se ha empleado un entorno de programación que ha permitido el desarrollo de una aplicación en un entorno gráfico de ventanas con la posibilidad de utilizar menús, botones, barras deslizantes, etc.. Para tal fin se ha utilizado un lenguaje de tipo visual el cual permite diseñar un entorno agradable de interfaz con el usuario. Dicha interfaz presenta, una serie de ventanas donde se muestra, entre otras funciones, el contenido de la zona de memoria de datos, memoria de programa, memoria de pila, los registros internos del microprocesador y los bits de estado.

Asimismo, se permite trabajar en tiempo real con el simulador, introduciendo datos y programas directamente desde el teclado, o bien leyendo los mismos desde un fichero previamente editado. En este último caso, el simulador permite la utilización de un editor de textos, que ha sido diseñado específicamente para el mismo, el cual contempla además la posibilidad de edición de programas en un sencillo lenguaje ensamblador para 8085. Asimismo, se pueden grabar los resultados de una simulación en un fichero.

Por último, se desarrolló en hipertexto un manual de ayuda en línea, integrado en el propio simulador, utilizable dentro de él o bien independientemente.

3. Aspecto del simulador

En la Figura 1 se muestra el aspecto que presenta la interfaz con el usuario del simulador de 8085. En ella pueden distinguirse las siguientes zonas:

3.1. Registros de la CPU

El 8085 cuenta con varios registros internos, (situados en la parte superior derecha del simulador) B, C, D, E, H y L, el A (acumulador) y el F (bits de estado), de 8 bits cada uno, y los registros SP (puntero de pila) y PC (contador de programa), de 16 bits. Los registros AF, BC, DE y HL, se presentan por parejas, pues muchas de las instrucciones del 8085 usan estas agrupaciones de registros. Situando el puntero del ratón sobre los bits de cada registro es posible cambiar su valor.

3.2. Bits de estado

Existen para el 8085 cinco bits de estado, signo (S), cero (Z), acarreo auxiliar (Ac), paridad (P) y acarreo principal (C). Aparecen en forma de pequeños leds en la parte central del simulador. Si un led está encendido indica que el bit está activo y si está apagado, el bit estará inactivo. Situando el puntero del ratón sobre los leds es posible cambiar su estado.

3.3. Puertos de Entrada y de Salida

El 8085 cuenta con 256 puertos de entrada y 256 puertos de salida de 8 bits cada uno que se sitúan en la parte inferior central del simulador. Todos los puertos aparecen en una misma lista, en la que en la columna izquierda se indica el número de puerto correspondiente.

3.4. Memoria de instrucciones

El 8085 dispone de una memoria de 65536 bytes. En esta memoria se cargan las instrucciones de los programas y los datos. Las instrucciones ocupan un byte, dos bytes y tres bytes. En la zona de memoria de instrucciones (parte izquierda del simulador) aparecen tres columnas, que contienen la siguiente información:

- *Dirección:* indica la posición de memoria en hexadecimal y puede ir entre 0000_h y $FFFF_h$.
- *Nemotécnico:* indica el nemotécnico de la instrucción ubicada en esa dirección de memoria.
- *Código:* contiene el código de la instrucción en hexadecimal y puede ir entre $00)_h$ y $FF)_h$.

También se puede apreciar que dos de las filas están iluminadas con distinto color que las demás:

- En color rojo se señala la posición del contador de programa (PC).
- En color azul se indica una posición seleccionada para ser editada por el usuario.

Pulsando con el botón derecho del ratón sobre la cabecera de la ventana de instrucciones aparecen tres opciones:

- *Ir a dirección de PC:* se salta a la posición en la que se encuentra situado el contador de programa.

- *Ir a dirección de comienzo de programa:* se salta a la dirección de comienzo del programa.
- *Ir a dirección...:* Aparece una ventana en la que puede introducir una dirección de la memoria de instrucciones.

3.5. Memoria de datos y memoria de pila

Estas dos listas que aparecen en la parte inferior derecha del simulador tienen una estrecha relación con la memoria de instrucciones. En realidad todas ellas contienen los mismos datos. La casilla en color azul corresponde a la selección de la posición actual. Para cambiar cualquier posición de las memorias o situarse en una posición determinada, basta con picar sobre ella con el cursor del ratón. En la memoria de pila siempre aparece una posición resaltada con el color verde indicando la posición del puntero de pila. Si se pulsa con el botón derecho del ratón sobre la cabecera de la memoria de datos o pila, aparece un menú emergente con 2 opciones:

- *Ir a dirección SP:* Esta opción no es igual para la memoria de pila y para la memoria de datos.
  - En la memoria de pila, muestra la posición a la que apunta el puntero de pila.
  - En la memoria de datos, esta opción permite ir a la primera dirección donde se encuentran los datos del programa.
- *Ir a dirección...:* Tanto para el caso de la memoria de datos como para la memoria de pila, se puede introducir una posición de memoria.

3.6. Control de ejecución

Situado en la parte central del simulador, contiene 4 botones desde los que es posible controlar la ejecución de sus programas. Cada botón realiza una tarea distinta:

- *Botón Step:* Ejecuta la siguiente instrucción paso a paso y se detiene.
- *Botón Over:* Tiene la misma función que el botón anterior salvo que, cuando se llega a una instrucción de llamada a subrutina, el simulador ejecuta todas las instrucciones pertenecientes a la subrutina, dejando el contador de programa en la siguiente instrucción a la instrucción de llamada.
- *Botón Run:* Ejecuta en modo continuo.
- *Botón Stop:* Para la ejecución en modo continuo.

3.7. Panel de interrupciones

El 8085 permite detener la ejecución normal de un programa mediante una serie de interrupciones. Los parámetros que controlan las interrupciones se han representado por casillas (situadas en la parte inferior izquierda del simulador) que pueden tomar dos posibles valores, ☑ (activa) o ☐ (inactiva). Existen 4 tipos de interrupciones, TRAP, RST 7.5, RST 6.5 y RST 5.5. Además existe un elemento más, llamado INTR, el cual le indica si las interrupciones están permitidas o no. Cada interrupción tiene asociadas dos casillas que tienen el siguiente significado:

- La casilla de la izquierda indica si una interrupción está permitida o no. La interrupción TRAP siempre está permitida.
- La casilla de la derecha permite realizar las peticiones de interrupción. En el momento se que quiera realizar una petición de interrupción basta con pulsar con el ratón en la casilla correspondiente.

3.8. Entrada / Salida serie

En la esquina inferior izquierda aparece un pequeño panel con dos casillas, que le permitirá controlar la entrada serie (SID) y salida serie (SOD).

- Casilla SID: Si está activa, cuando se ejecute la instrucción RIM, se cargará en el bit 7 del acumulador el valor 1. En caso contrario, este bit contendrá el valor 0.
- Casilla SOD: Cuando se ejecute la instrucción SIM, si los bits 6 y 7 del acumulador están puestos al valor 1, la casilla se activará. En caso contrario la casilla permanecerá inactiva.

4. Conclusión

Los objetivos más relevantes que se han conseguido con el simulador de programación de microprocesador Intel 8085 han sido:

- Acercar al estudiante al 8085.
- Ayudar al profesorado a dar a conocer el 8085.
- Facilitar el aprendizaje del 8085.
- El programa trabaja bajo un entorno de Windows 9x o Windows 2000 en un computador Personal Compatible lo que facilita enormemente su difusión.

- La interfaz con el usuario se ha llevado a cabo en entorno de programación que permita el desarrollo de una aplicación en un entorno gráfico amigable entre el usuario y el simulador.
- La interfaz presenta una serie de ventanas donde se muestra el contenido de la zona de memoria de datos, memoria de programa, memoria de pila, los registros internos del microprocesador y los bits de estado.
- El simulador incorpora un fichero de ayuda en línea, así como un editor de textos integrado para editar, compilar y hacer funcionar los programas de simulación.

Referencias

[1] Alecop. *Monografía nº 20. Microprocesador 8085*. 1987.

[2] José María Angulo. *Microprocesadores y Microcontroladores 8085, MCS-51 y ST-6*. Paraninfo, 1996.

[3] Timothy Budd. *Introducción a la programación orientada a objetos*. Addison-Wesley Iberoamericana. 1994.

[4] Antonio Cañas. *Apuntes de la asignatura Estructura de los Computadores I y II*. Escuela Técnica Superior de Ingeniería Informática. Universidad de Granada, 1999.

[5] Francisco Charte. *C++ Builder 4*. Anaya Multimedia, 1999.

[6] C. H. Pappas, W. H. Murria. *Manual de Borland C++*. McGraw-Hill, 1993.

[7] Alberto Prieto, Antonio Lloris y Juan Carlos Torres. *Introducción a la Informática*. McGraw-Hill, 2001.

[8] Alberto Prieto, Julio Ortega, Antonio F. Díaz y Antonio Cañas. *Estructura y funcionamiento de microprocesadores*. 3ª Ed. Copistería La Gioconda. Granada, 1999.

[9] Manuel Torres. *Microprocesadores y Microcontroladores aplicados a la industria*. Paraninfo, 1991.

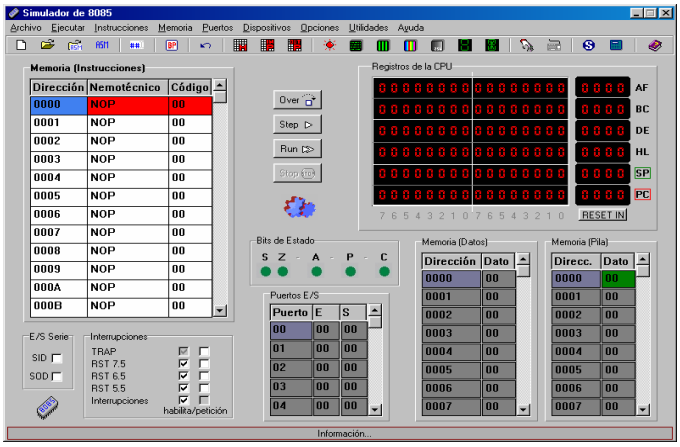


Figura 1. Aspecto inicial del Simulador de 8085.

Jlógica, un entorno de diseño de sistemas digitales

José A.Alvarez, V.G.Ruíz, J.F.Sanjuan, Javier Roca P.
Depto. Arquitectura de Computadores y Electrónica.
Universidad de Almería.
04120 Cañada de San Urbano.
jaberme@ual.es

Carlos A. Bermejo
Área de desarrollo – S. Informática.
Universidad de Almería.
04120 Cañada de San Urbano.
cbermejo@ual.es

Resumen

Jlógica es el resultado de una visión docente del lenguaje de programación Java. Con Jlógica se dispone de una herramienta de diseño y simulación de circuitos homogénea, intuitiva y sin restricciones, capaz de facilitar el aprendizaje en asignaturas tales como Estructura de Computadores, Tecnología de Computadores y Laboratorio de Arquitectura.

1. Introducción

Jlógica es una herramienta Java pensada para homogeneizar el entorno de trabajo en las prácticas de varias asignaturas, de esta manera módulos realizados para una determinada asignatura podrían ser reutilizables a medida que el alumno vaya adquiriendo conocimientos de mayor nivel con la consiguiente ventaja del conocimiento y experiencia práctica obtenida con el entorno de trabajo. Una de las principales ventajas de la herramienta es la independencia que otorga Java con respecto a la plataforma, así como la capacidad de integración con la red. La primera es importante ya que no restringe el sistema operativo usado en los equipos de los laboratorios. La segunda ventaja es que se hace extremadamente sencilla su utilización ya que una vez instalado en un servidor Web, el acceso puede realizarse tanto desde el laboratorio como desde cualquier punto conectado a la red.

Por otra parte, Jlógica dispone de un módulo generador de código Java a partir del diseño gráfico del circuito, el cual facilita el análisis del comportamiento del mismo mediante la modificación de parámetros básicos. Las asignaturas para las que se está desarrollando esta aplicación son Tecnología de Computadores y Estructura de Computadores.

En este artículo se detallan las características de utilización de esta herramienta. En el apartado 2 se describe la interfaz y en el apartado 3 su utilización como entorno de simulación de circuitos. En el apartado 4 se detalla la utilización de esta herramienta como generador de código a partir del diseño gráfico de un circuito.

Además de haber optado por Java para la implementación por las razones antes aducidas, portabilidad e integración en la red, se pueden argumentar razones secundarias tales como que Java dispone de librerías de clases (Swing [4] y Awt [2]) que facilitan el desarrollo de la interfaz además de sobrados componentes visuales libres[1] y reutilizables [1].

2. Descripción de la interfaz.

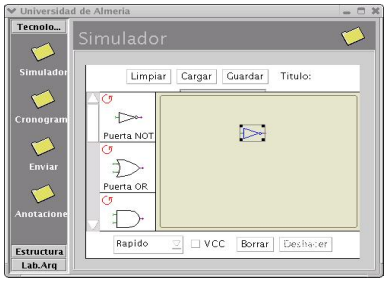


Figura 1. Aspecto de la interfaz Jlógica.

[1] Libre ha de entenderse que el propietario del software cede el componente para su uso de manera desinteresada.

Jlógica cuenta como punto de entrada con una interfaz muy similar a la interfaz ofrecida por Outlook para relacionar la aplicación con las asignaturas que harán uso de ella y separar adecuadamente la funcionalidad de la aplicación. El panel de la izquierda dispone las áreas:

- Tecnología Computadores.
- Estructura de Computadores.
- Laboratorio de Arquitectura.

Dentro de cada una de estas áreas se encuentra la funcionalidad requerida para las asignaturas, sea el caso mostrado en Fig.1 en el que para la asignatura de Tecnología se dispone de opciones como *Simulador* que inicia la aplicación, *Enviar* para mandar al profesor resultados, etcétera. Pero la parte importante de la aplicación no es la interfaz que la relaciona con las asignaturas sino el propio simulador (Fig.2).

Figura 2. Aspecto del simulador.

El simulador dispone de un juego de puertas lógicas básicas (AND, OR, NOT) y de una serie de piezas de utilidad como conectores de entrada/ salida y de empalme (Fig. 3), las cuales están dispuestas en la barra de puertas a la izquierda del área de trabajo Jlógica. Con estas puertas y piezas de utilidad el alumno puede comenzar a montar circuitos fácilmente. Además Jlógica cuenta con dos barras de herramientas (superior e inferior) que suman funcionalidad a la herramienta. La barra superior cuenta con funciones aplicables a todas las puertas del circuito, estas son:

- Limpiar: Elimina el diseño del circuito por completo.
- Cargar/ Guardar[2]: Exportación / importación del circuito a disco.
- Título: Nombra un diseño de circuito para usos posteriores.

En la barra inferior contamos con funciones aplicables a un subconjunto de puertas del circuito tales como:

- Borrar: Elimina las puertas seleccionadas así como sus conexiones con otras puertas.
- Deshacer: Restaura el diseño del circuito anterior a una modificación.
- Iconizar: Construye una nueva puerta con el diseño del circuito que esté en ese momento activo en el panel. Esta nueva puerta se añade con el nombre que se haya usado en la opción "Título" para que pueda ser usada como un componente en otros diseños.

Además de estas opciones aparecen otras dos que tienen que ver con la simulación en sí, estas son el desplegable en el que se puede elegir la velocidad con la que se verán la transición entre estados de una tabla de verdad y el checkbox[3] *VCC* que es el que inicia la simulación haciendo las veces de fuente de alimentación.

3. Áreas de aplicación

Se entiende por áreas de aplicación a las asignaturas que pueden, en principio, hacer uso del simulador para adaptar parte de sus lecciones prácticas. Entre ellas se encuentran las que a continuación se enumeran.

3.1. Tecnología de computadores.

En esta asignatura Jlógica le permite al alumno comprobar de forma fácil cómo los resultados obtenidos tras una minimización correcta de funciones coinciden con las versiones canónicas de los mismos, bien expresada en su forma de tabla lógica o circuito combinacional.

[2] El circuito se copia en un fichero binario al que se le añade un identificador para que el simulador sólo abra los ficheros que él ha generado y no cualquier otro tipo de fichero.
[3] Opción que se activa o desactiva marcando con un tick en el lugar habilitado.

La herramienta simula al panel-entrenador que con carácter general es utilizado por los alumnos en este tipo de prácticas, disponiendo de los elementos fundamentales integrados en el panel (interruptores, leds, conectores, etc.).

La disponibilidad de puertas lógicas, flip-flops u otro tipo de circuito lógico se obtiene a partir de la posible combinación de los elementos básicos (inversor, puerta AND y puerta OR) que se hayan integrados (Fig.2) con el desarrollo de la herramienta. De esta forma la tarea del alumno, ante el testeo de cualquier circuito propuesto, consiste en el ensamblaje apropiado de estos tres elementos hasta conseguir la función necesitada. Toda función diseñada anteriormente puede encapsularse como elemento de circuito y reutilizarse posteriormente en diseños más complejos, confeccionando de esta forma una librería tan extensa como se necesite. A continuación se describe un ejemplo.

La Fig.3 muestra cómo construir un circuito para implementar una determinada función.

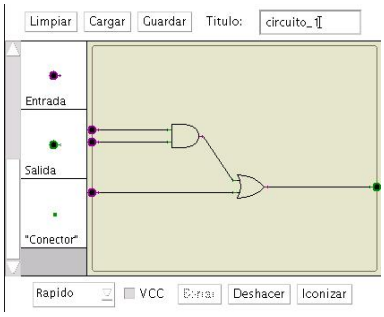


Figura 3. Diseño de circuito.

En Fig.4 se ve un ejemplo de cómo se encapsula para construir una puerta adicional llamada "circuito_1" con funcionalidad ampliada.

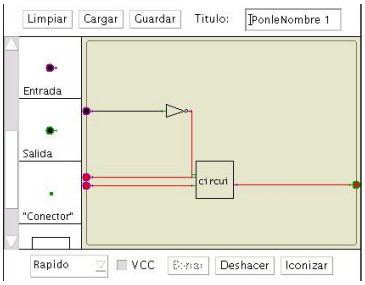


Figura 4. Encapsulación y reutilización.

En circuitos complejos donde el número de puertas sea elevado, existe la posibilidad de perder el diseño del circuito encapsulado que se está usando. La confusión es mayor si en lugar de un encapsulado se usan varios. Por tanto, JLógica permite inspeccionar el contenido de los circuitos encapsulados sin más que seleccionar la pieza encapsulada y pulsar sobre el botón "Expandir[4] ", tal y como se puede ver en la Fig. 5.

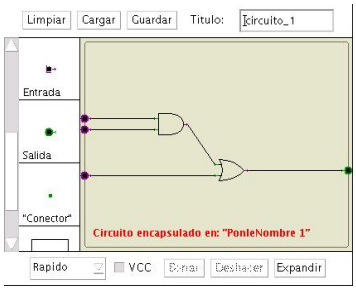


Figura 5. Contenido del encapsulado utilizado en el montaje.

3.2. Estructura de computadores.

El enfoque que se le otorga a esta asignatura en nuestro actual plan de estudios es el de iniciar al alumno en el conocimiento de los bloques fundamentales constitutivos de un computador.

[4] La opción expandir sólo se hace visible cuando existen componentes compuestos en el circuito.

El alumno al finalizar la asignatura ha debido adquirir el conocimiento de la metodología de diseño a nivel de registros así como la estructura y organización de una unidad central de proceso. Así pues, el aprendizaje realizado con módulos lógicos elementales en la asignatura Tecnología de Computadores son aplicados en esta asignatura para constituir bloques más complejos como ALU's o el diseño completo de una unidad central de proceso sencilla.

Las características de Jlógica explicadas anteriormente facilitan la consecución de estos objetivos sin más que sustituir el conjunto de piezas básicas usadas en la asignatura Tecnología de Computadores por un nuevo conjunto de piezas necesarias para esta asignatura como pueden ser una ALU, bancos de memoria, bancos de registros, multiplexores, etcétera.

4. Generación de código.

Uno de los aspectos prácticos tratados en Laboratorio de Arquitectura de Computadores es la simulación computacional de circuitos lógicos [3], mediante la que el alumno experimenta el testeo de bloques computacionales constitutivos de una arquitectura analizando el comportamiento ante distintos patrones de entrada y modificación de parámetros característicos.

Jlógica facilita el análisis anterior mediante la generación de código (Fig.6) a partir del diseño gráfico del bloque a estudiar. El modelo en esta versión se genera en código Java, si bien se pretende ampliar a otros lenguajes tal capacidad.

5. Conclusiones.

Jlógica goza de las capacidades propias del lenguaje Java como son la independencia de la plataforma lo que le confiere gran portabilidad y la hace una herramienta muy flexible y poco restrictiva, integrable en Internet. Jlógica se integra en páginas Web sin dificultad (applet). Otra ventaja de Jlógica es el carácter interdisciplinario que adquiere al poder utilizarse en diferentes asignaturas.

El desarrollo de Jlogica es el fundamento de una herramienta con la que se pretende en un futuro inmediato alcanzar niveles de complejidad de una arquitectura completa, abordando tanto el diseño del camino de datos como el de la unidad de control.

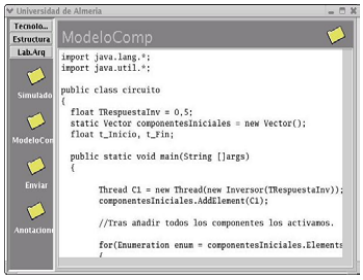


Figura 6. Ejemplo de generación de código para un inversor.

Referencias

[1] Duguay, Claude. Revista electrónica JavaPro. http://www.fawcette.com/archives/magazines/javapro

[2] Knudsen, Jonathan. Java 2D Graphics. First Edition. ISBN 1.56592-484-3. Ed. O'Reilly.

[3] Ruiz, V.G. *Diseño de Simuladores Digitales usando SDLC++*. **IV Congreso de Tecnologías Aplicadas a la Enseñanza de la Electrónica (TAEE)**. Barcelona, Spain. Septiembre. 2000.

[4] Walrath, Kathy y Campione, Mary. The JFC Swing Tutorial, a guide to construct GUIs. ISBN 0-201-43321-4. Ed. Addison-Wesley.

Estructura de computadores, simuladores e Internet

Javier García Zubía y José María Sáenz Ruiz de Velasco
Dpto. de Arquitectura de Computadores, Dpto. Ingeniería del Software
Facultad de Ingeniería
Universidad de Deusto
48007 Bilbao
e-mail: zubia@eside.deusto.es, jmsaenz@eside.deusto.es

Resumen

El trabajo presenta la aplicación Simul_Intenet que aborda la simulación de dos computadores básicos. Las ventajas de esta aplicación se centran en el grado de detalle al simular y en que está disponible en Internet. Simul_Internet es interesante para los docentes y alumnos de un curso de Estructura de Computadores.

1. Introducción

Lo tratado en los siguientes párrafos afecta en la Universidad de Deusto a las asignaturas de *Estructura de Computadores I* de Ingeniería Informática (primer semestre del segundo curso) y de *Tecnología de Computadores* de Ingeniería de Telecomunicaciones (segundo semestre del pirmer curso), ambas de seis créditos: 4,5 teóricos + 1,5 prácticos.

A nuestro modo de ver, el objetivo principal que han de tener estas asignaturas es que el alumno sea capaz de diseñar completamente un computador básico y programar en él. El primer paso es un trabajo en el aula, mientras que el segundo necesita de un simulador.

En general, los simuladores permiten cargar un programa, ejecutarlo y observar los resultados. En algunos, además, la ejecución puede ser instrucción a instrucción. Por último, los simuladores más completos son capaces de simular estado a estado, a nivel de microinstrucción. Se puede hablar por tanto de tres niveles de simulación.

La aplicación Simul_Internet contempla los tres niveles de simulación y permite la modificación del camino de datos durante la ejecución del programa. En cuanto a la implementación, Simul_Internet ha sido desarrollada en JAVA y esta accesible desde Internet, con las ventajas (y desventajas) que esto supone.

A continuación se presentan los computadores simulados, el entorno de ensamblado y el simulador propiamente dicho.

2. Computadores básicos: la Máquina Sencilla, la M+ y la M++

Antes de pasar a los simuladores es necesario describir, aunque sea brevemente, los computadores simulados: la MS y la M+.

2.1. La Máquina Sencilla

La Máquina Sencilla (MS) es ya es un clásico en la disciplina. Fue diseñado por Valero y Ayguadé en 1989 [1]. Brevemente, la MS es un computador con estructura de Memoria-Memoria, con cuatro instrucciones (ADD F, D, CMP F, D, MOV F, D y BEQ D) y direccionamiento absoluto. La memoria es de 128x16 bits y el camino de datos aparece en la figura 1.

El autómata simplificado para la unidad de control cableda cuenta con 8 estados, tres entradas y 10 salidas, las propias de la unidad de control.

La ventajas de la MS son bien claras: sencillez extrema y facilidad de diseño. Todo esto sin dejar de ser un computador. Su desventaja principal es que la MS es demasiado simple, demasiado sencilla.

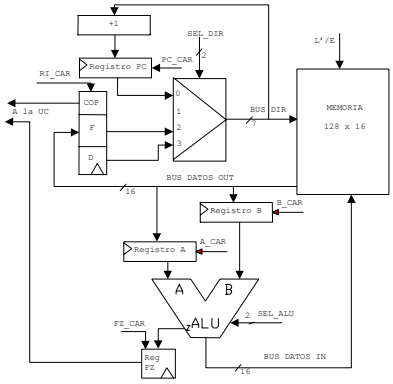


Figura 1. Camino de Datos de la Máquina Sencilla

2.2. La Máquina Plus (M+)

La M+ es un computador básico pero más evolucionado que la MS. Su primer diseño data del año 1997 [2].

La estructura de la M+ es Registro-Registro, tiene un único bus de datos bidireccional de 8 bits, 26 instrucciones (ADD, SUB, MOVE, AND, OR, XOR, NOT, CMP, INR, BEQ, BC y JMP, LDA, STA, LDAX, STAX, LFA y SFA) y un bus de direcciones de 16 bits potente y variado: inmediato, por registro, absoluto e indirecto. La memoria es de 64 Kbytes y el camino de datos puede verse en la figura 2.

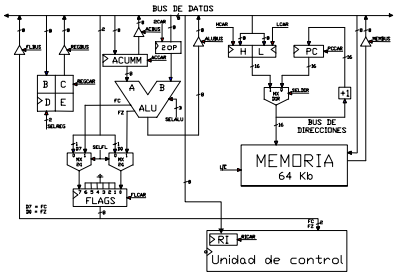


Figura 2. Camino de datos de la M+

El autómata de la unidad de control cableada de la M+ tiene 25 estados, 10 entradas (8 bits del código de operación y los flags FZ y FC) y 21 líneas de salida, las propias de la unidad de control.

2.3. El simulador de la Universidad de Deusto

El simulador desarrollado en la Universidad de Deusto, Simul_Internet, tiene como características principales:

- es accesible desde Internet,
- contempla conjuntamente a la MS y la M+,
- las unidades de control están implementadas cableadas mediante autómatas,
- niveles de ejecución: programa, instrucción y estado,
- visualiza dinámicamente el autómata de la unidad de control,
- permite la modificación interactiva de la máquina,
- carga y ensambla los programas con control del usuario y
- es potente, cómodo, visual y gratuito.
- Está accesible en: http://paginaspersonales.deusto.es/zubia

En la Figura 3 se ve el aspecto de la pantalla principal de Simul_Internet. En los siguientes párrafos describiremos brevemente la aplicación.

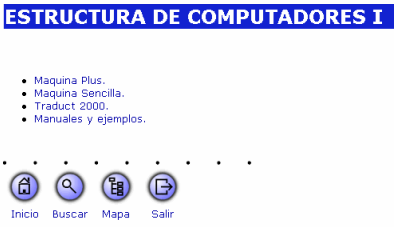


Figura 3. Pantalla principla de Simul_Internet

2.4. El ensamblador de Simul_Internet

En primer lugar se muestra el funcionamiento y aspecto del ensamblador de Simul_Internet (Traduct 2000). En la Figura 4 se puede ver como el proceso de ensamblado es una pantalla aparte. En ella el alumno escribe el programa en

ensamblador, y al ensamblarlo obtiene el código hexadecimal que deberá cargar en la memoria del computador correspondiente.

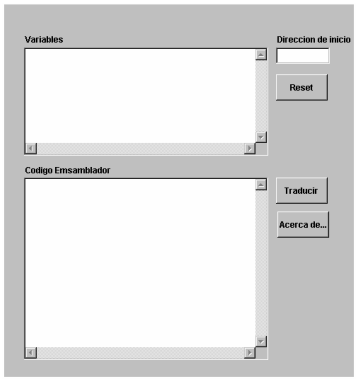


Figura 4. Pantalla principal del ensamblador

En la Figuras 5 y 6 se pueden ver, respectivamente, un programa en ensamblador de la M+ y el correspondiente código hexadecimal.

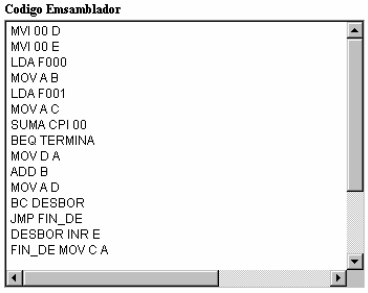


Figura 5. Programa en ensamblador de la M+

Una vez obtenido el código hexadecimal, el usuario dispone del mecanismo para cargar este programa en cualquier posición de la memoria.

2.5. La simulación de la MS

Si en la Figura 3 se hubiera pulsado la opción Máquina Sencilla, el alumno se habría encontrado con la Figura 7.

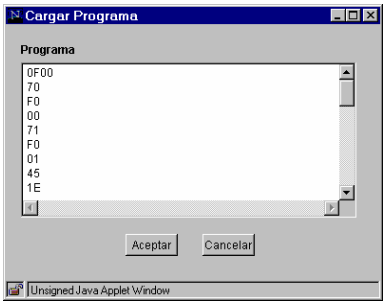


Figura 6. Programa en código hexadecimal

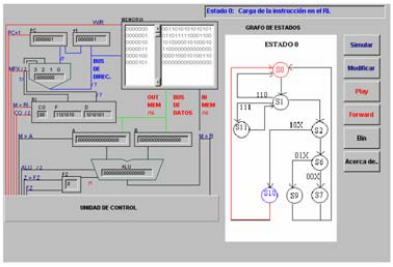


Figura 7. Pantalla del simulador de la MS

En la Figura 7 se pueden ver en la parte izquierda el camino de datos y las señales de control de la MS. Todos los registros, campos, etc. son accesibles por el alumno, que podrá modificarlos.

En la parte de la derecha se pueden ver los controles del nivel de simulación, la opción de modificación de campos y la elección binario/hexadecimal.

En la parte central se puede ver el diagrama de transición de estados del autómata de la unidad de control cableada. Cuando el alumno haya elegido la opción de simulación por estados podrá ver pintado en rojo el estado actual, y en azul el anterior. La visualización dinámica del autómata nos parece muy importante desde el punto de vista didáctico.

2.6. La simulación de la M+

Si en la Figura 3 se elige la opción Máquina Plus lo que nos encontramos es la Figura 8.

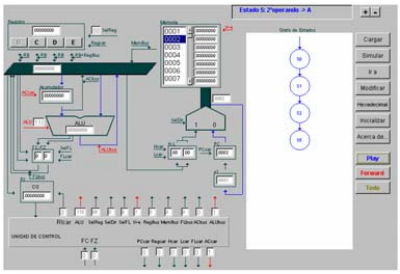


Figura 8. Pantalla del simulador de la M+

La pantalla es algo más compleja, pero igualmente manejable. Podemos ver en la parte izquierda el camino de datos y las señales de control, en la derecha, el control del nivel de simulación y en el centro, el diagrama de estados del autómata de la unidad de control cableada.

En cuanto a la simulación propiamente dicha, el alumno podrá ver cómo van cambiando el estado y contenido de la máquina con cada flanco de reloj. El Simul_Internet resalta visualmente, cambiando de color, aquellas líneas que modifican su valor.

La idea aplicada en Simul_Internet es que ha de verse todo, y todo debe ser accesible, siendo el alumno el que tiene el control de la máquina.

3. Experiencia docente

Desde el curso 1999-2000 el alumno dispone de una primera versión de Simul_Internet. Bastará una clase de 2 horas para que el alumno se haga con la aplicación.

La aplicación es utilizada por más de 300 alumnos al año. Las asignaturas implicadas son, como ya se ha dicho, *Estructura de Computadores I* en el primer semestre del segundo curso de Ingeniería Informática y *Tecnología de Computadores* del segundo semestre del primer curso de Ingeniería en Telecomunicaciones.

Aun a falta de estudios formales, la experiencia es positiva, y así nos lo hacen saber los alumnos. Ellos destacan de Simul_Internet su sencillez, potencia, adecuación al aula y disponibilidad en la red, y señalan como desventaja los problemas derivados de "la lentitud Internet".

4. El simulador e Internet

Un aspecto novedoso de Simul_Internet es que utiliza a Internet como plataforma. La aplicación se ha desarrollado en JAVA. La experiencia ha sido positiva en general, pero también ha habido problemas de lentitud. Por un lado la red es lenta y por otro, toda aplicación JAVA necesita ser interpretada por una máquina virtual de alto nivel, lo que no hace sino retardar todavía más la ejecución. Como opción a esta lentitud está que el alumno descargue la aplicación y la ejecute en su propio computador.

La ventaja de usar Internet reside en que la aplicación está disponible de continuo y en que la aplicación que se distribuye siempre está actualizada. Además poco a poco todos los centros de enseñanza están haciendo esfuerzos por trasladar parte de sus clases a la red.

Los requisitos para ejecutar la aplicación son un computador cualquiera tipo Pentium, un navegador actualizado (Internet-Explorer, Netscape, Mozilla, etc.) y el runtime para ejecutar programas en JAVA 1.3 o posterior.

5. Conclusiones

En el trabajo se ha presentado el simulador Simul_Internet. Su interés y originalidad se centran especialmente en su disponibilidad vía WEB, en el abanico de posibles niveles de simulación y en el control total que el usuario ejerce. Simul_Internet permite al profesor enseñar computadores básicos con el nivel y enfoque que el profesor desee, convirtiéndose en una herramienta ideal para enseñar en el aula.

Referencias

[1] Valero, M. y Ayguadé, E. *La Máquina Sencilla: Introducción a la estructura básica de un computador,* Facultad de Informática de la Universidad Politécnica de Cataluña, 1989.

[2] Angulo Usategui, J.M. García Zubía, J. y Angulo Martínez, I. *Fundamentos y estructura de computadores,* Ed. paraninfo, 2003.

CGRAPHIC: una herramienta gráfica para la enseñanza de los fundamentos de la programación (usando C)

Antonio J. Fernández, Jesús Millán Sánchez
Dept. de Lenguajes y Ciencias de la Computación, ETSII,
Campus de Teatinos, Universidad de Málaga
29071 Málaga
e-mail: afdez@lcc.uma.es

Resumen

Se describe una herramienta multimedia de apoyo docente para la enseñanza de los conceptos básicos de la programación. Esta herramienta consiste en un tutorial-simulador gráfico accesible en la Web que permite al alumno ahondar, gráfica y gradualmente, en los conceptos fundamentales de la programación imperativa.

1. Motivación y trabajo relacionado

En las ingenierías universitarias, la programación es una de las asignaturas básicas que más tiempo necesita para su comprensión y además, es un hecho reconocido que el lenguaje C es elegido en muchas universidades españolas como primer lenguaje de programación. Particularmente, [3] discute las adecuación de utilizar C como primer lenguaje de programación en las ingenierías. Entre sus ventajas citamos su practicidad, la generalidad de uso, motivación para el alumno, existencia de numerosas aplicaciones industriales escritas en C así como la disponibilidad gratuita de múltiples entornos para C. La principal desventaja consiste en que "es un lenguaje difícil tanto de enseñar como de aprender en un primer curso" universitario.

Un aspecto que contribuye a una mejor comprensión de la programación es la visualización dinámica de sus conceptos coordinada con la ejecución gradual de las sentencias fuente. En realidad existen muchos trabajos con fines educativos para facilitar el aprendizaje, aunque la mayoría se enfocan en el nivel físico sobre el funcionamiento interno de la máquina [2,5] y otros incluyen algún tipo de animación visual [4,6,9]. Sin embargo, se aprecia una carencia de simuladores orientados a lenguajes de alto nivel (e.g., [1] simula ejecuciones de código en PASCAL).

Para reducir en alguna medida la dificultad de enseñar y aprender los conceptos básicos de la programación usando el lenguaje C, hemos elaborado una herramienta que permite ahondar, de forma gradual y gráfica, en los fundamentos de la programación imperativa. Esta herramienta, que llamamos *CGRAPHIC*, es original (i.e., no está basada en otras) y complementa, de forma gráfica e interactiva, los conocimientos adquiridos en clase, los cuales se explican de forma intuitiva.

2. CGRAPHIC como software educativo

CGRAPHIC puede ser fácilmente catalogado como software educativo [8] puesto que cumple muchas de sus características tales como

- facilidad de uso,
- capacidad de motivar al alumno,
- tratamiento de temas relevantes,
- versatilidad y accesibilidad,
- interactividad con el alumno y
- originalidad.

Además, CGRAPHIC cumple las principales funciones del software educativo [8], tales como:

- *una función instructiva* pues orienta el aprendizaje de los estudiantes;
- *una función motivadora* ya que incluye elementos gráficos que captan la atención de los alumnos, mantienen su interés y los guía hacia los aspectos más importantes de los conceptos;
- *una función evaluadora* porque da una respuesta inmediata a las acciones de los alumnos según éstos las demanden.

3. Teoría + práctica

CGRAPHIC integra un tutorial on-line, de los conceptos básicos del lenguaje C, y un simulador visual de la ejecución de programas escritos en C, combinando pues una parte teórica con otra eminentemente práctica.

En la parte teórica, cada concepto básico de programación viene acompañado primeramente de una explicación teórica similar a la que pueda aparecer en cualquier libro básico de programación. Primero se define un concepto y luego se muestran, a nivel teórico, ejercicios explicativos del mismo. Además, se proponen una lista de conceptos afines que el alumno puede consultar de forma inmediata on-line.

En la parte práctica, asociado a cada concepto existe un conjunto de ejercicios interactivos cuya ejecución es visualizada gráficamente. En este contexto práctico, se combinan características de un intérprete de C con las de un depurador del mismo. Básicamente CGRAPHIC visualiza la ejecución de un programa diseñado mediante el lenguaje de programación C. La versión actual incluye un conjunto amplio de programas -de dificultad diversa- que se asocian a conceptos fundamentales de la programación imperativa, tales como: tipos básicos, tipos compuestos (registros-estructuras, arrays, etc), constantes, variables locales y globales, ámbitos de visibilidad, estructuras de control, bucles, direcciones de memoria, punteros, ficheros, funciones, procedimientos, paso de parámetros y tipos abstractos de datos (TADs).

4. El entorno de visualización

El entorno gráfico de ejecución de la parte práctica consta de lo siguientes elementos, los cuales son fácilmente identificables en la Figura 1: (1) *Una pantalla gráfica* para mostrar una representación gráfica de los elementos del programa C a medida que éstos son creados, modificados y/o eliminados; (2) *Una pantalla de código* que muestra el código del ejercicio actual que se está representando; además en cada uno de los estados intermedios, se resalta cuál es la siguiente línea de código que va a ser ejecutada; (3) *una pantalla de salida,* que indica tanto la salida estándar (es decir el monitor) del programa como instrucciones al usuario para un correcto funcionamiento; (4) *un control del modo de ejecución* (ver explicación posterior)*;* (5) *un cuadro de entrada,* que proporciona interactividad pues permite introducir datos como si del teclado se tratase; (6) un botón para iniciar la visualización *(ejecución) y* (7) *un botón de acción,* para pasar de un estado a otro en la ejecución.

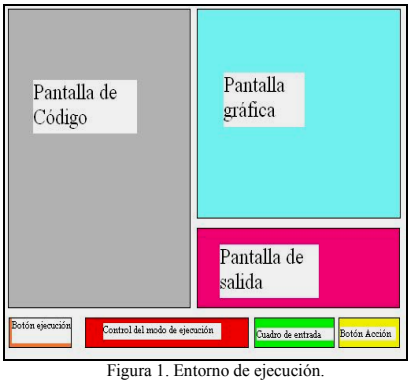


Figura 1. Entorno de ejecución.

La implementación: Debido a las limitaciones de espacio, comentaremos brevemente la implementación de CGRAPHIC. La parte teórica ha sido diseñada básicamente mediante páginas HTML; el entorno gráfico ha sido desarrollado en Java y sólo requiere un programa navegador para ser ejecutado. La Figura 2 muestra el esquema de todas las clases empleadas: se han implementado una serie de objetos genéricos que son usados para visualizar diferentes conceptos de programación. Entre esos objetos tenemos: variables, arrays, funciones, punteros, mapas de memoria, ficheros, estructuras y buffer de teclado. Cada objeto se implementa mediante una clase Java y cada ejercicio (e.g., *Ex*), implementado en otra clase Java (e.g., *Ex.java*), contiene un programa C que se carga durante la ejecución en la ventana de código del entorno (ver Figura 1). La ventana de código es definida en el fichero *Code.java*. El resto del entorno gráfico se implementa en el fichero *GraphicBox.java*, en cual es compartido por todos los ejercicios (los cuales también comparten la misma plantilla, implementada en el fichero *Template.java*). Más información puede ser encontrada en la siguiente dirección:

http://campusvirtual.uma.es/fundinfo/online/English_CGRAPHIC/default.htm.

Modos de ejecución: Se permiten dos modos posibles de ejecución a elegir por el usuario: un modo *directo*, en el cual el programa se ejecuta tal y como lo haría en un entorno clásico de programación en C, y un modo *pausado o paso a paso*, en el cual los programas son ejecutados instrucción a instrucción y la resolución (trazas) de cada instrucción es mostrada gráficamente por pantalla. En este modo, el usuario indica cuándo se quiere pasar al siguiente estado mientras se observan cómo varían los elementos del programa (variables, punteros, memoria,...) en los estados intermedios; consiste pues en una interpretación gráfica del ejercicio.

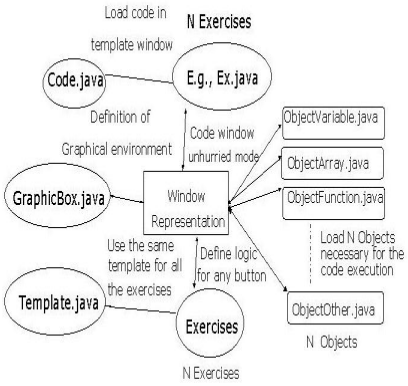


Figura 2. Esquema de Clases.

Ejemplos: La Figura 3 muestra un estado intermedio derivado de la ejecución paso a paso de un ejercicio (concretamente, el de intercambio de enteros usando una función *swap*). En esta figura se observan algunos de los *objetos* tratados gráficamente en la herramienta tales como las variables globales (en verde), las variables locales (en azul), el mapa de memoria, la línea de código a ejecutarse (resaltada en rojo), etc.

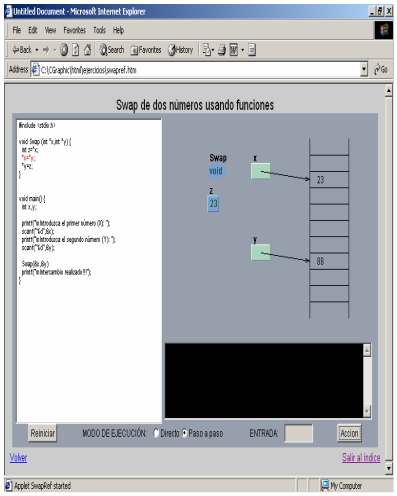


Figura 3. Estado intermedio de ejecución *paso a paso*.

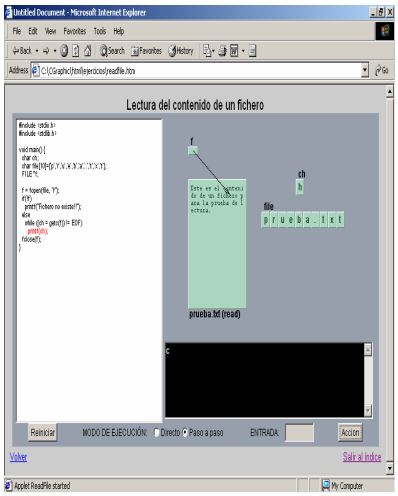


Figura 4. Otros estados intermedios. Ficheros-Arrays.

Las Figuras 4 y 5 muestran el estado intermedio de la ejecución pausada de otros

ejercicios relacionados con otros conceptos (en la Figura 4 la lectura de ficheros y las cadenas de caracteres; en la Figura 5, las direcciones de memorias, punteros, funciones y el TAD pila).

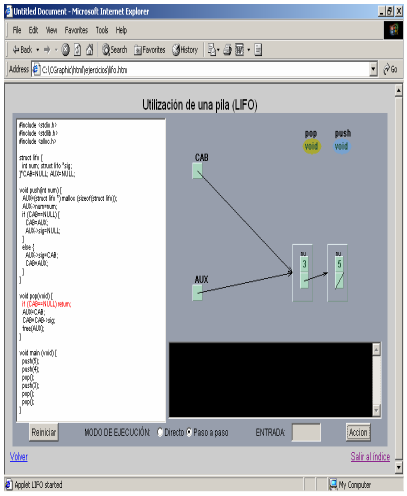


Figura 5. Otros estados intermedios. TADs-punteros

5. Conclusión

Hemos presentado CGRAPHIC una herramienta original diseñada exclusivamente para la enseñanza, usando C, de los conceptos fundamentales de la programación imperativa. Esta herramienta reduce el enorme hueco existente entre la asimilación de la teoría y su utilización práctica cuando se utiliza el C como primer lenguaje de programación. CGRAPHIC está totalmente operativa y accesible en la siguiente dirección web: *http://campusvirtual.uma.es/fundinfo/ (enlace On-line)*.

Evaluación Con respecto a la evaluación de la herramienta, todavía es pronto para analizar su uso por parte de los alumnos pues ha comenzado a ser operativa en este segundo cuatrimestre. Sin embargo diremos que, desde el punto de vista del profesor, nos está resultando muy útil a la hora de explicar conceptos difíciles tales como la "entrada de datos bufferizada". En este sentido estamos usando el típico cañón para proyectar en las clases prácticas la visualización paso a paso de ejercicios, la cual es complementada con la explicación teórica del profesor. Esto facilita la comprensión del concepto por parte del alumno.

Al final del curso, proyectamos realizar una evaluación plena, siguiendo criterios ya establecidos [7].

Referencias

[1] Brette J-F. Transparent running and contextual help to learn and to teach an imperative language. *SIGCSE Bulletin*, 27(2):7-12, 1995.

[2] Campbell R. Introducing computer concepts by simulating a simple computer. *SIGCSE Bulletin*, 28(3):9-11, 1996.

[3] Fernández A.J., Trella M., Aranda M.C. y Galindo P. *Nuevas tecnologías y metodologías para la enseñanza de una introducción a la programación de ordenadores en ingenierías.* III Congreso Chileno de Educación Superior en Computación, Chile, 2001.

[4] Garralón J., Contreras D. y Núñez P. *PERICO: Programa Educativo de pRogramación Imperativa con visualización de elementos físiCOs*. II Jornadas Andalusíes de Informática Gráfica, pp:63-72, 2000.

[5] Hassapis G. An interactive electronic book approach for teaching computer implementation of industrial control systems. *IEEE Transactions on Education*, 46(1):177-184, 2003.

[6] Henderson W. Animated models for teaching aspects of computer systems organizations. *IEEE Transactions on Education*, 37 (3):247-256, 1994.

[7] Iglesias O., Paniagua C. y Pessacq R. Evaluation of University Educational Software. *Computer Applications in Engineering Education*, 5(3):181-188,1997.

[8] Marques P. *Software educativo. Guía de uso y metodología de diseño*. Estes, 1995.

[9] Robbins S. y Robbins K. A microprogramming animation. *IEEE Transactions on Education*, 41(4):293-300, 1998.

Una Aplicación Interactiva para Visualizar las Hipótesis Generadas por Algoritmos de Aprendizaje Computacional

Santiago David Villalba Bartolomé, Juan José Rodríguez Díez*
*Área de Lenguajes y Sistemas Informáticos
Universidad de Burgos
e-mail: sdvb@wanadoo.es, jjrodriguez@ubu.es

Resumen

Se presenta una herramienta que puede ser empleada en la docencia de asignaturas de Minería de Datos, Aprendizaje Computacional, o de Inteligencia Artificial, que incluyan estas cuestiones dentro de su temario. Inspirada en diversas aplicaciones que visualizan las superficies de decisión generadas por algún tipo de clasificador, la presente herramienta es capaz de mostrar gráficamente la hipótesis lanzada por una vasta gama de clasificadores, desde árboles de decisión hasta redes neuronales, en el marco de un problema bidimensional (sencillo aunque visualizable) y multiclase.

1. Introducción

OAIDTB (Otra Aplicación Interactiva Demostrando Técnicas de Boosting) es una suite de programas para la aplicación y evaluación de técnicas de clasificación en general, y de boosting en particular, a problemas de aprendizaje computacional. Los programas que incluye la suite son los siguientes:

- Una biblioteca de clases java que implementan diversos algoritmos de boosting, extendiendo a la biblioteca WEKA [11]; a este conjunto de clases las llamaremos a partir de ahora boosters. Pueden ser utilizados desde la línea de comandos o embebidos en otro código java.
- Una aplicación gráfica para la selección, configuración, aplicación y evaluación de los métodos de aprendizaje tanto de WEKA como de OAIDTB.
- Una aplicación gráfica docente, *Bidimensional Generalization*, que permite visualizar, de una manera general, las hipótesis lanzadas por los clasificadores de WEKA y OAIDTB en el marco de un problema bidimensional y multiclase.

Existen diversas herramientas capaces de visualizar las superficies de decisión generadas por algún sistema de clasificación, como redes neuronales [13], máquinas de vectores soporte (SVM) [12], boosting [9][10], etc. La aportación principal de la herramienta que se presenta en este trabajo es que es capaz de visualizar una amplia gama de clasificadores, todos los incluidos en la biblioteca WEKA [11], así como aquellos que se implementen dentro del marco definido por esta biblioteca. En particular, si en la asignatura se exige la implementación de algún algoritmo, los alumnos son capaces de visualizar las hipótesis generadas por el clasificador que han desarrollado sin necesidad de programar la parte gráfica.

OAIDTB es software de código abierto y se distribuye bajo la licencia GNU General Public License. Se puede encontrar la última versión en http://pisuerga.inf.ubu.es/lsi/Asignaturas/MD/.

2. Un poco de teoría

A pesar de que se presupone en el lector un cierto conocimiento de los fundamentos necesarios para comprender el presente texto, se dan a continuación unas breves definiciones, más o menos precisas, de varios conceptos fundamentales.

- *Minería de datos:* Aplicación de algoritmos específicos para la extracción de patrones desde los datos.

- *Aprendizaje computacional:* En el contexto en que nos encontramos, este término se refiere al conjunto de teorías y algoritmos que forman la base técnica para la minería de datos.
- *Instancia:* La entrada para un esquema de aprendizaje computacional es un conjunto de instancias. Estas instancias son las "cosas" que deben ser clasificadas, agrupadas o asociadas. Cada instancia es un ejemplo individual e independiente del concepto que debe ser aprendido y cada instancia está caracterizada por los valores que toman un conjunto predeterminado de atributos.
- *Clasificador:* un algoritmo de aprendizaje clasificador trata de extraer información de un conjunto de instancias de entrenamiento, de manera que es capaz de predecir la clase a la que pertenecen ejemplares desconocidos (distintos de los utilizados para entrenarlo).
- *Boosting:* Los algoritmos de aprendizaje computacional basados en boosting (potenciación) buscan iterativamente una combinación lineal ponderada de clasificadores base que haga predicciones correctas en datos desconocidos.

Las clases que implementan a los algoritmos de boosting son el principal componente de la suite. Los algoritmos implementados son AdaBoostM1 [4], AdaBoostM1W [2], AdaBoostOC [6], AdaBoostECC [8], AdaBoostMH [5], RealAdaBoost [3], GentleAdaBoost [3], AdaCost [1] y CSBx [7].

3. Las aplicaciones gráficas

En el terreno de la didáctica son las aplicaciones gráficas de la suite OAIDTB las que tienen algo o mucho que decir.

La interfaz gráfica ha sido diseñada con el afán de ser cómoda e intuitiva para el usuario, proporcionando ayuda sobre qué es y para que sirve cada componente mediante "tooltips". El funcionamiento de las aplicaciones se basa, fundamentalmente, en el empleo del ratón, aunque se proporcionan teclas de acceso rápido a la mayoría de las funciones. Además, las aplicaciones no pierden en ningún momento la interactividad con el usuario, pudiendo siempre ser cancelados los procesos lanzados a pesar de que puedan, eventualmente, tener una importante carga computacional.

Sin más preámbulos, echemos un vistazo a lo que nos ofrecen estos programas.

3.1. El panel de clasificación

El panel de clasificación (Fig. 1) es una herramienta que permite seleccionar, o cargar un clasificador de OAIDTB o de WEKA, configurarlo, entrenarlo sobre un conjunto de instancias, guardarlo, evaluar su precisión proporcionando variadas estadísticas y una representación textual del clasificador y, en el caso de los boosters, mostrar detalles tales como la evolución del error, mediante gráficas cartesianas, o la precisión de los clasificadores base entrenados.

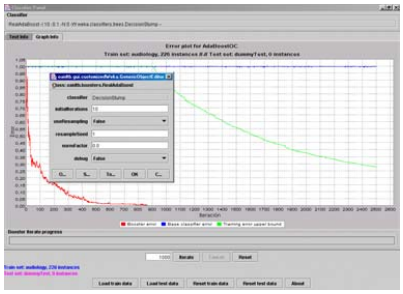


Figura 1. El panel de clasificación

La aplicación es flexible, ya que añadirle nuevos esquemas de clasificación es sencillo. Basta con implementar el algoritmo deseado en el marco proporcionado por la biblioteca WEKA y añadir el nombre de la clase a un fichero de texto que permite configurar la aplicación. La utilización de las capacidades de java para descubrir los miembros de las clases en tiempo de ejecución proporciona la posibilidad de incorporar estas clases, con sus características particulares, sin tener que recompilar el programa.

3.2. "Generalización Bidimensional"

Esta aplicación permite la visualización de las hipótesis lanzadas por diversos algoritmos de

aprendizaje computacional en el marco de un problema de datos bidimensional y multiclase.

En concreto, se trata de clasificar instancias representadas por puntos en un espacio bidimensional y cuyo atributo clase es el color de dicho punto. Con ella es posible enseñar, de una manera práctica y visual, cómo funcionan dichos algoritmos, mostrando propiedades teóricas de los mismos y conectando las hipótesis que lanzan con un problema que, aunque limitado, es fácil de entender y "entra por los ojos".

Una vez ejecutada, la ventana de la aplicación se divide en varias zonas (Fig. 2).

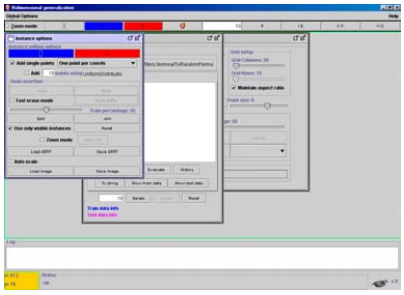


Figura 2. Pantalla principal de la aplicación

En la parte superior existe un pequeño menú con algunas opciones de configuración de la aplicación, y una barra de herramientas con botones para las acciones más frecuentes. En el centro se sitúa el área de dibujo, que es la región de la pantalla en que se dibujan las instancias (puntos de un determinado color) que introduce el usuario mediante pulsaciones del ratón. En la parte inferior existe un área de mensajes, en la que se le muestran al usuario las informaciones pertinentes, desde tiempos de ejecución hasta los posibles errores o excepciones producidas.

Flotando sobre todos estos componentes hay tres ventanas cuyo tamaño, para comodidad del usuario, puede ser alterado.

La primera ventana se corresponde con el panel de clasificación, ya comentado en la sección anterior.

La segunda ventana es el panel de edición de instancias. Permite introducir las instancias mediante pulsaciones de ratón en el área de dibujado, o cargarlas desde un archivo, bien sea en formato propio (ARFF – Formato de Archivo Atributo-Relación) o, y esta característica es experimental, desde un archivo de imagen gif o jpeg. Si se utiliza la inserción directa mediante pulsaciones del ratón, este panel permite la selección del color (clase) de los puntos (instancias) introducidos, así como permite la posibilidad de introducir un número arbitrario de instancias. Algunas funciones ofrecidas a través de los componentes de este panel son:

- Deshacer / rehacer la inserción de instancias.
- Configurar la función empleada en la inserción masiva de instancias, pudiendo elegir entre diversas funciones de distribución (normal, uniforme… o incluso las creadas por un usuario avanzado mediante del empleo de una simple API) para poder crear problemas interesantes o visualmente atractivos.
- Trasladar y escalar las coordenadas de las instancias para poder mostrarlas correctamente o más claramente en el área de dibujado.
- Dividir las instancias en conjuntos de entrenamiento y de prueba.
- Salvar las instancias en formato ARFF o una instantánea del contenido del área de dibujado.

La tercera ventana se corresponde con el panel de opciones visuales, que es el encargado de manejar todo lo referente a la visualización del problema, desde qué aspectos del mismo se muestran u ocultan hasta el tamaño de los puntos que representan las instancias, y de lanzar la creación de la imagen que represente la hipótesis del clasificador seleccionado y construido en el panel de clasificación, pudiendo configurar su nivel de precisión

Otro de los puntos fuertes del programa es la capacidad de hacer "zoom ilimitado" sobre cualquier región del universo del problema y

construir la imagen de hipótesis sólo sobre dicha región, de manera que es posible revelar las peculiaridades del clasificador en cuestión, como por ejemplo los límites de decisión de un árbol o de un clasificador basado en instancias.

.

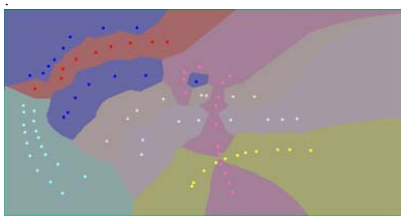


Figura 3. Hipótesis obtenida de un clasificador basado en instancias.

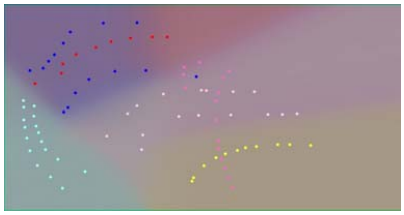


Figura 4. Hipótesis generada por una red neuronal

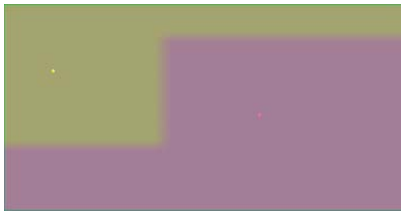


Figura 5. Zoom sobre una región de la figura 3

4. Conclusión

Se ha presentado una herramienta que permite representar, de manera gráfica y general, las hipótesis lanzadas por algoritmos de aprendizaje computacional basados en clasificación.

Referencias

[1] Fan, Stolfo, Zhang y Chan. *AdaCost: Misclassification cost-sensitive boosting*. Proceedings of The Sixteenth International Conference on Machine Learning. pp. 97-105. San Francisco: Morgan Kaufmann.

[2] Günther Eibl y Karl Peter Pfeiffer. *How to make AdaBoost.M1 work for weak base classifiers by changing only one line of the code*. ECML'02 - European Conference on Machine Learning

[3] Jerome Friedman, Trevor Hastie y Robert Tibshirani. *Additive logistic regression: a statistical view of boosting*. The Annals of Statistics, 38(2): 337-374, April 2000.

[4] Robert E. Schapire y Yoav Freund. *A decision-theoretic generalization of on-line learning and an application to boosting*. Journal of Computer and System Sciences, 55(1):119-139, 1997. van Leunen, M.C. *A handbook for scholars*. Oxford University Press, 1992

[5] Robert E. Schapire y Yoram Singer. *Improved boosting algorithms using confidence rated-predictions*. Machine Learning, 37(3):297-336, 1999

[6] Robert E. Schapire. *Using output codes to boost multiclass learning problems*. In Machine Learning: Proceedings of the Fourteenth Internatinal Conferences, pages 313-321, 1997.

[7] Ting, K.M.. *Cost Sensitive Classification Using Decision Trees, Boosting and MetaCost*. Book chapter in Heuristic and Optimization for Knowledge Discovery. Edited by Sarker, R., Abbass, H. & Newton, C. Idea Group Publishing. 2002.

[8] Venkatesan Guruswami y Amit Sahai. *Multiclass Learning, Boosting and Error Correcting Codes*. Proceedings of COLT'99

[9] [http://www.cs.huji.ac.il/~yoavf/adaboost/index.html], visitada el 23-1-03.

[10] [http://www.cs.technion.ac.il/~rani/LocBoost/], visitada el 23-1-03.

[11] [http://www.cs.waikato.ac.nz/ml/weka], visitada el 23-1-03.

[12] [http://svm.dcs.rhbnc.ac.uk/pagesnew/GPat.shtml], visitada el 23-1-03.

[13] [http://neuron.eng.wayne.edu/java/AHK/EPM_pps.html], visitada el 23-1-03

La herramienta ArtEM: aritmética entera y modular

Alfonso Gutiérrez, Violeta Migallón, José Penadés
Dpto. de Ciencia de la Computación e Inteligencia Artificial
Universidad de Alicante
03080 Alicante
e-mail: algutlan@hotmail.com
violeta@dccia.ua.es
jpenades@dccia.ua.es

Héctor Migallón
Dpto. de Física y Arquitectura de Computadores
Universidad Miguel Hernández
03202 Elche (Alicante)
e-mail: hmigallon@umh.es

Resumen

Los contenidos de matemática discreta en las titulaciones de informática incluyen una parte relativa a aritmética entera y modular. En este artículo presentamos una herramienta que ha sido diseñada para la realización de las prácticas de dicha parte y que actualmente se está utilizando en la asignatura Matemática Discreta de la Universidad de Alicante.

1. Introducción

La herramienta ArtEM (*Arit*mética *E*ntera y *M*odular) [3], es una aplicación informática programada en Visual Basic [5] y desarrollada con el fin de ser utilizada en las prácticas de cualquier asignatura que incluya como tópicos los relacionados con la aritmética entera y modular [1], [2], [4]. Está estructurada en 5 menús básicos:

- Euclides.
- Ecuaciones diofánticas.
- Números primos.
- Aritmética modular.
- Aplicación a la criptografía.

Los tres primeros menús están dedicados a la aritmética entera, el cuarto menú proporciona cálculos básicos en la aritmética modular como los cálculos del representante de clase, inverso de un elemento, función de Euler y potencias. El quinto menú constituye una aplicación a la criptografía centrándose en dos criptosistemas, uno de clave privada y otro de clave pública.

Todos los algoritmos disponibles en ArtEM se desarrollan de tal forma que el usuario es capaz de reconocer los pasos que se han seguido para su ejecución, de manera que se obtiene un importante valor pedagógico.

En las siguientes secciones describiremos el contenido de ArtEM, estudiando cada uno de sus menús por separado.

2. Menú Euclides

En este menú se desarrolla el algoritmo de Euclides para el cálculo del máximo común divisor de dos enteros. Además de describir el algoritmo de forma genérica se tiene la opción de mostrar todos los cálculos del propio algoritmo, tal y como se muestra en la Figura 1.

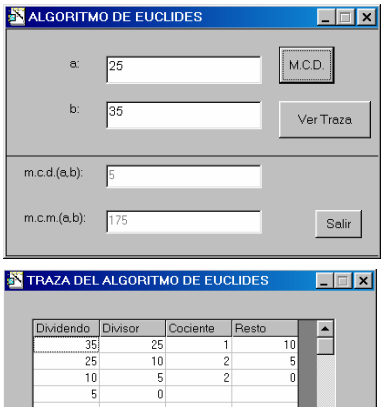


Figura 1. Algoritmo de Euclides

3. Menú ecuaciones diofánticas

En este menú se ofrece la posibilidad de resolver ecuaciones diofánticas, es decir, ecuaciones de la forma $ax+by=c$, donde a, b, c son enteros y x, y son las incógnitas que también son números enteros. Además de mostrar una descripción de los resultados teóricos necesarios para la correcta resolución de estas ecuaciones, se muestra el algoritmo necesario para el cálculo de una solución particular de una ecuación diofántica. En la ejecución del algoritmo, el usuario debe introducir los valores de a, b y c, obteniendo una solución particular de la ecuación diofántica correspondiente -cuya traza puede ser consultada- y la solución general. Como muestra presentamos la solución de la ecuación $2700x + 1500y = 234000$ en la Figura 2.

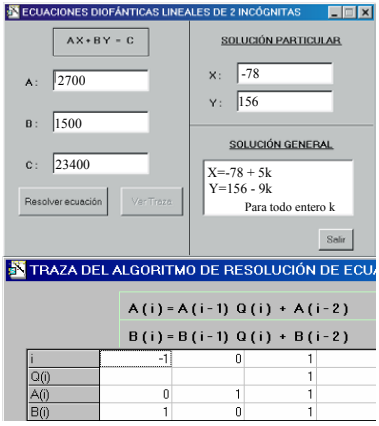


Figura 2. Ecuaciones diofánticas

4. Menú números primos

Se desarrollan en este menú procedimientos para crear una lista de números primos, averiguar si un número entero es primo y factorizar un entero en producto de sus primos. Estos algoritmos vienen acompañados de su descripción formal. La complejidad de estos algoritmos limita su uso a enteros pequeños. Las opciones que presenta este menú vienen indicadas en la Figura 3.

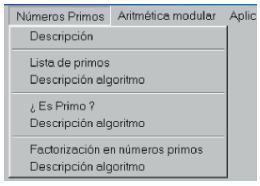


Figura 3. Opciones del menú números primos

5. Menú aritmética modular

Presentamos en este menú diversos cálculos básicos relacionados con la aritmética modular. Éstos son el cálculo del representante de clase en el conjunto de los enteros congruentes módulo n, que representamos por Z_n, el cálculo del inverso en Z_n, el cálculo de la función de Euler y el cálculo de potencias en Z_n. Como muestra presentamos el cálculo de la potencia $[5]^{75}$ en Z_{23}. El programa identifica que el $mcd(5,23)=1$ y por tanto, como el valor de la función de Euler en 23 es 22, se tiene que $[5]^{22}=[1]$. Así, como $[5]^{75}=([5]^{22})^3\ [5]^9$ sólo será necesario calcular $[5]^9 = [5]^8\ [5]$, que en este caso es $[11]$. Mostramos, en la Figura 4, la salida que se obtiene de la ejecución correspondiente a la traza del algoritmo.

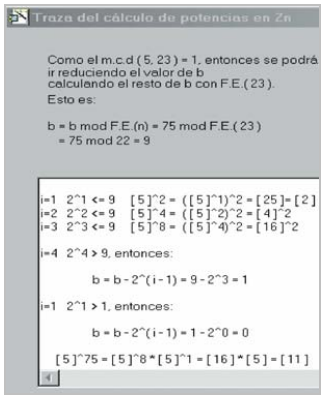


Figura 4. Cálculo de potencias en Z_n

6. Menú aplicación a la criptografía

En este menú pretendemos familiarizarnos con ciertas aplicaciones de la aritmética modular a la criptografía. Tiene dos partes claramente diferenciadas: la elección del alfabeto a utilizar y la elección del sistema criptográfico. En lo que se refiere a la elección del alfabeto, la aplicación tiene preestablecidos una serie de alfabetos que pueden ser seleccionados con el correspondiente menú, como muestra la Figura 5.

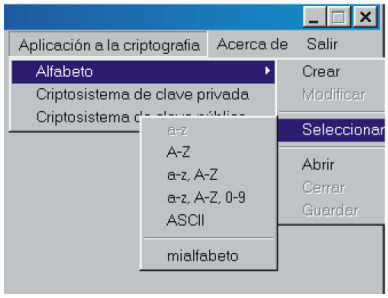


Figura 5. Elección del alfabeto

También se permite crear un alfabeto propio e incluso leerlo de disco si previamente se había creado. Para crear un alfabeto lo único que se debe hacer es ir asignando valores numéricos a cada uno de los caracteres que queremos que formen parte de nuestro alfabeto. El módulo con el que se trabajará en la codificación y descodificación vendrá dado en función del valor numérico asignado mayor. Como ejemplo, en la Figura 6, mostramos el alfabeto *{A,B,C,D,E,F,G}* al que se le han asociado las equivalencias numéricas *{11,16,1,23,20,17,24}* respectivamente y que en la Figura 5 viene definido con el nombre de *mialfabeto*.

Ya sea con un alfabeto creado por el usuario o con un alfabeto predefinido por la aplicación se dispone de dos tipos de criptosistemas: uno de clave privada y otro de clave pública. El criptosistema de clave privada corresponde con un criptosistema clásico cuyas funciones de cifrado y descifrado calculadas sobre Z_n son respectivamente:

$C_{r,s}([m]) = [r][m] + [s], \ / \ mcd(r,n)=1.$
$D_{r,s}([m^*]) = [r]^{-1}([m^*]-[s]).$

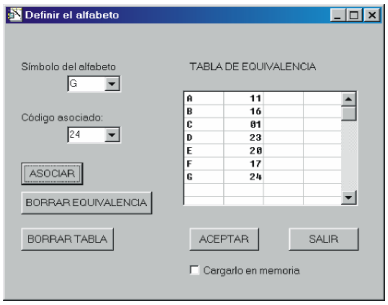


Figura 6. Definición de un nuevo alfabeto

Por su parte, el criptosistema de clave pública corresponde con el código RSA.

Como ejemplo de utilización de la aplicación ArtEM para este tipo de problemas, vamos a suponer que se ha seleccionado el alfabeto predefinido formado por los caracteres de la *A* a la *Z*, de la *a* a la *z*, y el espacio en blanco. Esto hace un total de *55* caracteres por lo que trabajaremos en Z_{55}. Vamos a realizar una codificación utilizando el criptosistema de clave privada. En primer lugar el programa nos pedirá *r* y *s*. Como *mcd(r,55)* debe ser *1*, el programa nos indica posibles valores de *r* a partir de un valor mínimo que el usuario introduce.

Si por ejemplo seleccionamos *s=8* y *r=6*, podremos, a través del botón *continuar*, iniciar una codificación con estas claves. La Figura 7 muestra la codificación de la frase "Esto es una prueba" usando este sistema criptográfico de clave privada y las claves anteriores. La primera ventana contiene la frase en cuestión que queremos codificar, la segunda ventana contiene la transcripción inmediata según el alfabeto que hayamos elegido y que se encuentra en la tabla de conversión, la tercera ventana contiene los valores numéricos de la codificación y la última ventana ya reproduce los caracteres codificados.

Así, con este sistema criptográfico la frase "Esto es una prueba" ha quedado codificada como "fJOñCcJCUhFCsDUcLF". La aplicación también permite invertir el proceso para descodificar un texto determinado. El proceso se realiza paso por paso pinchando en la correspondiente pestaña y en cada paso la aplicación nos da información de qué es lo que está haciendo.

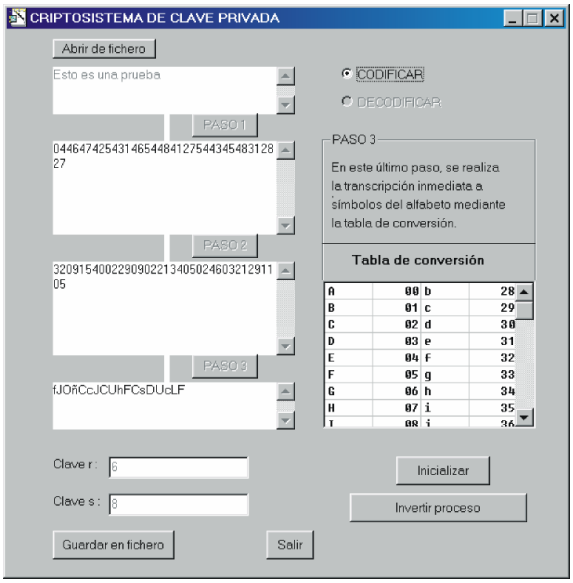


Figura 7. Ejemplo de codificación

7. Conclusión

El objetivo que nos marcamos con el diseño de la herramienta ArtEM fue el intentar impulsar el aprendizaje, experimentación, asimilación y ampliación de algunos de los contenidos de la matemática discreta, por parte del alumnado, con el uso del ordenador. No se trata de aprender a programar, pues para ello ya existen otras asignaturas, sino de aprovechar las capacidades pedagógicas del ordenador en beneficio de la calidad de nuestra docencia. La experiencia ha mostrado que el interés por parte del alumnado es muy aceptable y que además dichas prácticas facilitan la asimilación y comprensión de los contenidos de la aritmética entera y modular.

Tengamos en cuenta que, para el alumnado de informática en particular, esta materia tiene un grado de dificultad bastante considerable.

Referencias

[1] Biggs, N.L. *Matemática discreta.* Vicens Vives, 1994.

[2] Dierker, P.F., Voxman, W.L. *Discrete mathematics*. HBJ, 1986.

[3] Gutiérrez, A., Migallón, H., Migallón V., Penadés, J. *ArtEM*. Disponible en http://www.dccia.ua.es/~jpenades/ArtEM.html.

[4] Grimaldi, R.P. *Matemáticas discretas y combinatoria. Una introducción con aplicaciones*. Addison-Wesley, 1998.

[5] Petroutsos E., *Visual Basic 6*. Ediciones Anaya Multimedia S. A., 1999.

Uso de las referencias bibliográficas en la Ingeniería Informática[♣]

Francisco J. García Peñalvo
Dpto. de Informática y Automática
Universidad de Salamanca
37008 Salamanca
e-mail: fgarcia@usal.es

Roberto Therón Sánchez
Dpto. de Informática y Automática
Universidad de Salamanca
37008 Salamanca
e-mail: theron@usal.es

Ana B. Gil González
Dpto. de Informática y Automática
Universidad de Salamanca
37008 Salamanca
e-mail: abg@usal.es

Resumen
Las referencias bibliográficas que se incluyen en cualquier trabajo académico constituyen un importante valor a la hora de evaluar o clarificar los antecedes y/o las fuentes del trabajo en cuestión. No se suele considerar un factor crítico el correcto manejo de las citas bibliográficas hasta que los alumnos no están en el Tercer Ciclo de sus estudios, sin embargo, sería deseable que los alumnos de Primer y Segundo Ciclo tuvieran una mínima conciencia de la importancia que tiene una correcta lista de referencias, y fueran creando un hábito de organización de las fuentes que consultan. En este artículo se presenta una herramienta de soporte a la gestión de las fuentes bibliográfica, BiblioRef, desarrollada en el Departamento de Informática y Automática de la Universidad de Salamanca.

1. Introducción

Cualquiera que se haya enfrentado a la tarea de escribir un texto científico, habrá tenido que mediar con la labor de manejar las referencias bibliográficas que se pretenden incluir en el cuerpo del trabajo.

A la necesidad de obtener, seleccionar, consultar, comprender y sintetizar los contenidos de aquellas referencias que se consultan, hay que añadir la tarea de registrar las fuentes de nuestros conocimientos para poder referenciarlas en nuestras producciones científicas, siendo muy conscientes de que las listas de referencias bibliográficas que se incluyen en éstas son un criterio ampliamente utilizado para medir la calidad y la utilidad nuestras publicaciones.

La tarea de adaptar las referencias a un estilo concreto de citado se convierte en una labor tediosa, que se acrecienta cuando el número de referencias aumenta notablemente, especialmente para aquellos usuarios que trabajan en un entorno de procesamiento como puede ser MS Word.

Si bien es cierto que el correcto manejo de las fuentes bibliográficas no se les inculca a los alumnos hasta que llegan a su período doctoral, una lista de referencias escasa o inexistente empobrece enormemente los trabajos realizados durante sus estudios y especialmente los Proyectos Fin de Carrera (PFC) con los que culminan su carrera.

Los principales problemas que, en relación con la bibliografía, se han detectado en los PFC son:

- Carencia de cualquier tipo de información bibliográfica.
- Inclusión de una sección de bibliografía al final del documento, pero carencia de citas en el texto.
- Inclusión de una sección de bibliografía al final del documento en la que se mezclan tanto ciertas citas que se han incluido en el texto como otras obras consultadas pero no citadas.
- Aparición de citas en el cuerpo del documento, que luego no aparecen en la sección de referencias.
- Errores o carencia de un estilo de citado concreto, cuando no mezcla de varios estilos.

Nuestro Departamento está preocupado por facilitar la labor de gestión de las fuentes bibliográficas, así se ha desarrollado BiblioRef, herramienta que permite la gestión de los datos

[♣] Este trabajo ha sido parcialmente subvencionado por la Junta de Castilla y León y la Unión Europea a través del Fondo Social Europeo mediante el proyecto de investigación SA017/02.

asociados de las fuentes bibliográficas y su integración con MS Word, facilitando la inclusión de citas en un documento para, automáticamente, generar la sección de referencias de acuerdo a un estilo de citado.

Aunque BiblioRef fue desarrollada para los miembros del Departamento, algunos profesores estamos facilitándola a los alumnos para que gestionen las fuentes bibliográficas en sus PFC, obteniéndose una mejora en la calidad de la documentación de los mismos, a la vez que se les inculca un método de trabajo científico.

2. BiblioRef

Cuando el número de referencias a manejar crece, un procedimiento manual para la gestión de las fuentes bibliográficas se vuelve ineficiente y propenso a los errores. Este crecimiento plantea dos tipos de problemas: la gestión de los datos de las fuentes bibliográficas y el esfuerzo de ajustar la bibliografía a un formato concreto de citado.

Para solucionar el primer problema se puede recurrir a crear una base de datos personal, que facilite esa gestión de las fuentes bibliográficas. Pero para el segundo problema no existe una solución tan sencilla. Quizás el paradigma de actuación para este caso es el que se propone en LaTeX [2] donde, gracias a su interacción con BibTeX [3], se facilita enormemente la labor de adaptar las fuentes a un estilo de citado. Sin embargo, la utilización de LaTeX encuentra muchas reticencias en un amplio sector de usuarios, que se decantan por el otro estándar de facto a la hora de escribir un texto, MS Word. No obstante, MS Word no incorpora por sí mismo herramientas adecuadas para facilitar la gestión automática de las referencias bibliográficas.

BiblioRef [4] es una aplicación que cumple esta doble función de gestión de bibliografía y de inserción automática de las referencias en la sección de bibliografía conforme a un estilo de citado en un documento MS Word.

La interacción de BiblioRef con MS Word, viene a paliar una importante carencia de este último, ya que MS Word no aporta herramientas para la gestión bibliográfica, recayendo en el autor toda la responsabilidad del mantenimiento, normalmente realizado de forma manual, de la lista de referencias bibliográficas a incluir en un determinado documento.

La relación entre BiblioRef y MS Word es tal que cuando el autor requiere la inserción de una nueva cita, utiliza BiblioRef para seleccionarla de la colección existente (y ampliable en cualquier momento) y copiarla, pegándola en el punto de inserción dentro del documento Word. Una vez terminado el documento, se podrá generar automáticamente la sección de bibliografía del mismo, utilizando el estilo de citado elegido entre los soportados por BiblioRef. Además, no se olvida a BibTeX, ya que se siguen patrones de uso similares a los de este entorno, y se incorporan facilidades de exportación/importación de ficheros BibTeX, facilitando a los usuarios de BiblioRef no limitarse a un entorno concreto como es Word.

La Figura 1 ilustra cuál es el ciclo de vida de una fuente desde el momento en que se capturan sus datos hasta el instante en que se desea incluir en un documento.

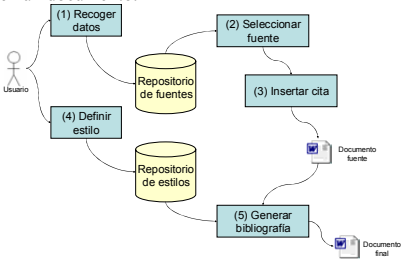


Figura 1. Ciclo de vida de una fuente bibliográfica

2.1. Recogida de datos

La primera fase en el ciclo de vida de una fuente bibliográfica es la extracción de los datos de la publicación, esto es, la información relevante que la define y que va a servir para catalogarla, buscarla o simplemente citarla.

Esta actividad que puede parecer trivial, pero nos puede obligar a invertir mucho tiempo, especialmente cuando no somos metódicos al extraer esos datos al obtener la publicación (ya sea físicamente, digitalmente o a través de referencias que aparecen citadas en otras publicaciones).

Para la correcta recolección de los datos de una fuente, con objeto de almacenarla en el repositorio de fuentes de la herramienta BiblioRef, debe conocerse cuál es el tipo de la publicación que está tratando. Los tipos de fuente bibliográfica que soporta BiblioRef, están muy influenciados

por los tipos soportado en BibTeX, aunque no existe una equivalencia completa. Concretamente se soportan: Artículos de revista, Libros, Capítulos de libros, Actas de congresos, Artículos en actas de congresos, Tesis, Documentos técnicos, Documentos electrónicos y Otras.

Para insertar los datos de una publicación (así como para editarlos) existe un asistente compuesto por cinco pasos, donde en el primer paso se introducen los datos básicos (entre ellos el tipo de la fuente, que incidirá en los datos concretos solicitados posteriormente), en el siguiente formulario se puede indicar si se tiene una versión digital de la publicación, en el tercero se introducen los autores por orden de firma (pudiéndose indicar si son o no editores de la publicación), en el cuarto paso es posible asociarle unos datos de clasificación a la fuente en base a áreas temáticas y materias concretas dentro de dichas áreas, y en el quinto y último paso se introducen los datos que completan la información de la fuente y que son dependientes del tipo de publicación de que se trate.

2.2. Selección de una fuente

A la hora de seleccionar una fuente para proceder a su inserción como cita en un documento se debe localizar ésta.

Cuando el número de citas que se manejan es pequeño, esa localización puede hacer a través de la ventana principal de la herramienta donde aparecen las fuentes en una rejilla, mostrándose los valores de algunos de sus atributos (título, tipo, año…) y con la característica de ordenar ascendente o descendentemente todas las fuentes por un campo, sin más que hacer clic con el puntero del ratón sobre el nombre de la columna que representa al atributo (Figura 2). Sin embargo, esta herramienta incluye una completa interfaz de búsqueda que permitirá obtener un subconjunto de las fuentes que se tienen almacenadas.

Figura 2. Presentación de las fuentes en la pantalla principal de BiblioRef

2.3. Inserción de una cita en un documento

Una vez que se tiene localizada la fuente que se desea citar, este paso es tan sencillo como dejar el punto de inserción del documento Word en la posición donde se quiere que se inserte la cita, una vez hecho esto se conmuta a la aplicación BiblioRef y se ejecuta la acción "Insertar Referencia" sobre la fuente seleccionada, y automáticamente se inserta un objeto BiblioRef en el documento Word (ver Figura 3).

2.4. Definición del estilo

A la hora de construir una lista de referencias en un documento, nosotros debemos indicar a BiblioRef cuál va a ser el estilo de citado que queremos emplear. Esta decisión implicará tanto el formato de las citas o llamadas que aparecerán en el cuerpo del documento, como el formato de los datos con que se muestra cada cita en la lista de referencias del documento.

Por defecto BiblioRef incluye tres estilos de citado: ACM, APA y un estilo de citado propio. Si estos estilos satisfacen al usuario podría generar de forma automática una sección de referencias en un documento. En caso contrario, el usuario cuenta con una interfaz avanzada donde puede definir de forma gráfica el estilo de citado que necesite.

2.5. Generación de la lista de bibliografía

El último paso sería la generación automática de la lista de referencias de un documento, en este caso BiblioRef sólo necesita que se le diga cuál es

el fichero que contiene los objetos referencia, cuál es fichero que contiene la definición del estilo y cuál es fichero destino que contendrá la sección de referencias generadas.

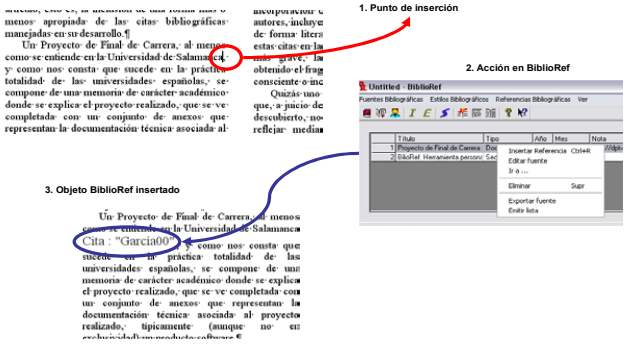


Figura 3. Inserción de una cita en MS Word

3. Trabajos relacionados

Existen numerosas herramientas de gestión de fuentes bibliográficas tanto de libre distribución, BibEdit (http://www.iui.se/staff/jonasb/bibedit/) o BibDB (http://www.mackichan.com), como de carácter comercial, ProCite (http://www.procite.com) o EndNote (http://www.endnote.com), pero estas herramientas se quedan en el ámbito de la gestión con la capacidad de exportar a formato BibTeX, pero no interaccionan con Word, uno de nuestros objetivos fundamentales.

Una herramienta que cuenta con características similares a BiblioRef, por cumplir el doble objetivo de la gestión y la integración con Word, es BibWord (http://mariachi.dsic.upv.es/bibword) [1]. En este caso la integración con Word se hace a través de macros programadas en WordBasic y no mediante una aplicación externa, y BibWord carece de soporte para la incorporación de nuevos estilos de citado.

4. Conclusiones

En este artículo se han realizado una serie de reflexiones sobre el escaso manejo que, en general, los alumnos de la Ingeniería Informática tienen sobre la gestión de la bibliografía y cómo esta circunstancia se refleja de una manera significativa en los PFC, que suelen quedar empobrecidos por la poca información bibliográfica que contienen.

Tratando de paliar este problema, se ha presentado BiblioRef, una herramienta que posibilita la gestión del conocimiento derivado de las fuentes bibliográficas, y que a su vez facilita el citado en entornos como MS Word y LaTeX. Esta herramienta, inicialmente desarrollada para facilitar la labor de los investigadores, está siendo puesta al alcance de los alumnos, obteniéndose prometedores resultados, al mejorar la calidad de los documentos en cuanto a las referencias y por crear en ellos un hábito de trabajo al organizar el conocimiento que adquieren de las fuentes bibliográficas que consultan.

Por último, decir que esta herramienta se puede conseguir para su libre utilización en http://tejo.usal.es/~fgarcia/docencia/isoftware/case/biblio.html.

Referencias

[1] Canós, J. H. A Bibliography Manager for Microsoft Word. *ACM Crossroads*, Vol. 6, Nº 4, 2000.

[2] Lamport, L. *A Document Preparation System*. Addison-Wesley, 1986.

[3] Patashnik, O. BibTeXing. BibTeX Distribution, 1988.

[4] Sánchez, J. M. García, F. J., Hernández, J. A. BilioRef. Herramienta personal para la gestión de citas bibliográficas. *Actas de ISKO 2003*.

SldDraw: Un trazador de árboles SLD

Francisco Gutiérrez
Dpto. de Lenguajes y Ciencias de la Computación
Universidad de Málaga
e-mail: pacog@lcc.uma.es

Mª del Carmen de Castro
Dpto. de Lenguajes y Sistemas Informáticos
Universidad de Cádiz
e-mail: maricarmen.decastro@uca.es

Resumen

Se presenta una aplicación software, en el ámbito de la Programación Lógica, que dibuja árboles de ejecución del método de resolución SLD. La aplicación está diseñada tanto como herramienta didáctica para el aprendizaje del método de resolución SLD con todas sus variantes como herramienta para el tutor que le permite encontrar árboles SLD con las características deseadas. Está destinada a alumnos y profesores que imparten docencia en niveles universitarios en asignaturas relacionadas con la Programación Lógica.

1. Introducción

En un principio, se pensó diseñar una herramienta de ayuda al profesor a la hora de proponer ejercicios que tracen árboles de ejecución por el método SLD [1] (árboles SLD). Antes de pasar a la realización de la aplicación se buscaron utilidades que ya resolvieran ese problema. La más parecida es la que se ofrece en un paquete junto el texto "Computational Intelligence. A Logical Approach" [2]. De todas formas, aunque esta herramienta genera árboles SLD, su propósito no es el mismo que el que guía a la aquí presentada.

SldDraw[3] pretendía cubrir una necesidad docente. Normalmente, a la hora de proponer ejercicios de traza de árboles SLD, el profesor se encuentra con la dificultad de encontrar árboles adecuados, en espacio y complejidad. El trazado a mano de los árboles es una tarea de cierta complejidad y encontrar esos árboles adecuados requiere mucha experiencia y sobre todo tiempo. La herramienta conseguida permite generar árboles SLD de forma automática a partir de un programa y un objetivo, e ir modificando tanto el programa como el objetivo hasta conseguir el árbol de tamaño adecuado.

Posteriormente se pensó que la aplicación también podía servir como herramienta didáctica en manos de los alumnos y profesores por lo que se le incorporaron más facilidades como cambios de reglas de búsqueda y selección, descripción detallada de las unificaciones que aparecen en cada rama del árbol, ejecuciones paso a paso, de éxito en éxito, puntos de ruptura, etc.

Con objeto de poder hacer trazas de ejecución de programas escritos en Prolog, se incorporan también características propias del lenguaje como aritmética extralógica, listas, predicados sobre metaprogramación entre los que cabe destacar el predicado =.., corte, etc.

Con SldDraw, el profesor puede seleccionar ejercicios en el que se signifiquen las diferencias que se producen al cambiar de regla de búsqueda, de regla de selección, al reordenar los objetivos, al realizar distintos usos de un predicado, etc. Por otro lado, al alumno le vale como test para comprobar que entiende los mecanismos que comporta la realización de árboles SLD y en particular, el modelo de ejecución de Prolog, así como el uso de ciertos predicados específicos de este lenguaje de programación o el efecto del corte.

SldDraw no incluye un intérprete de Prolog eficiente sino simplemente un simulador. Por tanto, no es posible realizar trazas de programas

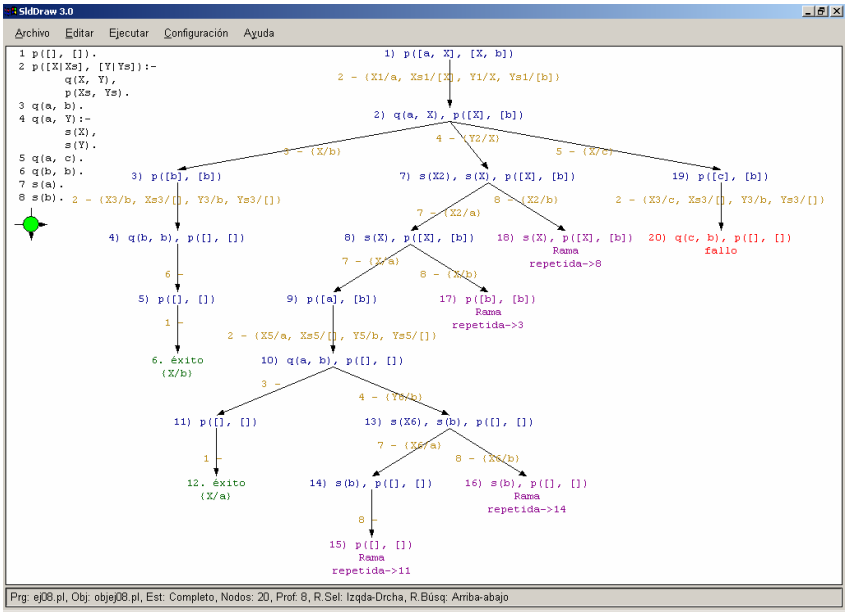


Figura 1. Ejemplo de ejecución de SldDraw

Prolog que generen una gran cantidad de nodos. Como ejemplo puede decirse que un árbol con 600 nodos y profundidad 40 se consigue visualizar en un tiempo razonable.

2. Descripción

Se trata de un trazador de árboles de ejecución SLD que muestra el árbol SLD y permite manipularlo hasta conseguir una representación adecuada. Al contrario del modelo de ejecución de Prolog, el árbol se genera por niveles y, dado que un árbol puede tener ramas infinitas, se fija inicialmente un nivel máximo de desarrollo pudiendo en cada momento modificarse. Un círculo verde o rojo nos indica claramente si el árbol está completo o no.

La aplicación admite como programas y objetivos a un subconjunto de los programas Prolog lo suficientemente amplio para que sea de utilidad. En particular, no incorpora características de entrada de datos y sólo incorpora los predicados básicos de salida. Está desarrollada en SWI-Prolog con XPCE [4] y este hecho resulta ser un aliciente para los alumnos que observan como el lenguaje que están aprendiendo es válido para la realización de aplicaciones de cierta complejidad. Existen versiones para Windows y para Linux: en realidad, esta es una facilidad de SWI-Prolog. Tiene un entorno similar al de cualquier aplicación actual basada en ventanas. Por otro lado, también se ha desarrollado una utilidad Java, llamada JsldDraw[5] bastante parecida a SldDraw que permite ser ejecutada como Applet y como aplicación.

Cada elemento del árbol es mostrado en un color diferente permitiendo distinguirlos y detectar fácilmente los distintos tipos de nodos; por ejemplo, un nodo fallo aparece en rojo y un nodo éxito en verde.

Se permite también modificar el programa o el objetivo mediante un editor que puede ser seleccionado por el usuario.

La aplicación dispone de un fichero de configuración en el que entre otras cosas, se permite modificar los colores de los nodos y unificaciones, el idioma de la aplicación, el editor utilizado, etc.

El algoritmo para la representación gráfica ha sido adaptado del propuesto para la representación de árboles en ML [6].

3. Ejecución

Para comenzar debemos disponer de un programa en Prolog que esté libre de errores y almacenado en un fichero de extensión *.pl*. Si el programa contiene errores, y se intenta abrir desde la aplicación se muestra una ventana de aviso indicando el error producido pero en la que la descripción del error no está suficientemente clara. A continuación se debe disponer de otro fichero con la misma extensión en el que estará escrita la consulta que queremos realizar.

Ya desde SLDdraw abrimos ambos ficheros, el que contiene el programa y el de la consulta y pedimos a la aplicación que nos represente el árbol. Otras opciones nos permiten iniciar el proceso de representación paso a paso.

Tanto el programa como el objetivo pueden modificarse desde la aplicación generando árboles para cada caso. Podemos imprimir el árbol resultante. En este caso podemos ajustar el número de páginas horizontales y verticales que ocupará la representación. Esto permite imprimir árboles cuya representación exceda del tamaño de una página. Además, podemos obtener un fichero encapsulado postscript con la representación.

Ya se ha mencionado que la herramienta no es un intérprete de Prolog. A diferencia de Prolog la aplicación trabaja con chequeo de ocurrencias (occur check) de manera que no produce unificación cuando esta situación se detecta. Por otro lado, detecta ramas infinitas y ramas cíclicas siendo la detección de estas últimas configurable.

4. Ámbito de utilización

Es una herramienta muy útil en las clases de prácticas de las asignaturas de programación lógica para el aprendizaje del lenguaje Prolog y en general de la resolución SLD. Desde el programa más simple se puede comprobar el árbol que genera mostrando unificaciones y respuestas computadas.

Dado que el aprendizaje de un lenguaje de programación declarativo como es Prolog suele resultar complicado, los alumnos agradecen disponer de una utilidad de estas características.

Es una aplicación relativamente pequeña y sencilla de aprender y utilizar. La distinción de los elementos que participan en la ejecución mediante colores permite desde el primer momento al alumno identificar cada parte y cada paso.

El hecho de que las unificaciones que se realizan aparezcan escritas en las ramas también hace más rápida su comprensión y aprendizaje. Así mismo, al desplegarse, aparece una ventana con la descripción paso a paso del proceso de obtención del unificador. Esto también ayuda a la comprensión del algoritmo de unificación de Robinson.

En la actualidad se está utilizando con éxito en dos universidades, la de Málaga y la de Cádiz. Así mismo, sabemos que se está utilizando también en algunas universidades de Centroamérica.

En particular, en Málaga se utiliza desde hace dos años, tanto en la Ingeniería Informática como en las Ingenierías Técnicas y aunque no se ha realizado ningún test para comprobar la ventaja que ha supuesto el uso de la herramienta, los alumnos, y sobre todo los repetidores que no la habían usado en años anteriores, han manifestado su reconocimiento por poder contar con una herramienta de estas características. Según sus comentarios, y en los laboratorios se pone de manifiesto, usan la aplicación a menudo para aclarar significados, variaciones según las formas de uso, etc. Con respecto al profesorado, ha aumentado la variedad y diversidad de ejercicios relativos a la

creación de árboles SLD tanto en las prácticas como en los exámenes.

En Cádiz, es el primer año que los alumnos de Ingeniería Técnica en Informática de Gestión están utilizando la herramienta. Se les ha pasado una encuesta al final del cuatrimestre y, en su mayoría, les parece muy positivo contar con una aplicación complementaria al intérprete que les facilite el aprendizaje. Les resulta fácil de manejar, y la consideran práctica para detectar errores y depurar los programas. También están de acuerdo, en su mayoría, en que comprenden mejor el método de Resolución SLD y la forma en que se ejecutan los programas en Prolog gracias a la aplicación.

En realidad el ámbito de utilización se podría extender hacia cualquier persona que necesite programar en Prolog .

5. Problemas o dificultades que resuelve

A continuación enumeramos las dificultades que a nuestro entender resuelve la herramienta:

- Dificultad inicial para comprender este tipo de programación por parte de personas que siempre (o en la mayoría de los casos) han programado y aprendido a programar en lenguajes basados en otro paradigma (imperativo, etc.).
- Comprensión de que la ejecución de programas en Prolog se realiza mediante el método de resolución SLD. Es decir, la búsqueda de soluciones se realiza en *búsqueda primero en profundidad con retroceso* sobre el árbol, una vez fijadas las reglas de selección y de búsqueda. Esto aclara de inmediato el porqué de la no completitud de Prolog.
- La depuración de programas mediante esta traza gráfica es mucho más clara que la depuración tradicional sobre programas lógicos.
- Aprendizaje del efecto de los predicados de control, en especial del corte.

6. Ventajas

Comentamos a continuación, algunos elementos de la aplicación que hacen atractivo su uso y extensible a otros ámbitos:

- Facilidad de instalación y uso, así como su reducido espacio.
- Posibilidad de configurar el idioma.
- Existencia de versiones para varios sistemas operativos.
- Inclusión de numerosos ejemplos.
- Posibilidad de imprimir o guardar el árbol.
- La modificación de la visualización permite obtener el mismo árbol de diferentes formas.

7. Conclusiones y posibles mejoras

En definitiva es una herramienta útil y didáctica dentro del ámbito de la Programación Lógica, que sirve tanto de apoyo al docente en su trabajo como al alumno en la tarea de aprender y asimilar.

Para el futuro se pretende dotar a la aplicación de un analizador sintáctico que resuelva errores en la construcción de programas y objetivos, la posibilidad de no desarrollar trozos del árbol que no sean significativos y la posibilidad de abrir ventanas con subramas (contemplada en JsldDraw).

8. Referencias

[1]J.W. Lloyd. Foundations of Logic Programming. Second, Extended Edition. Springer-Verlag, 1987.

[2] D. Poole, A. Mackworth y R. Goebel. Computational Intelligence. A Logical Approach. Oxford University Press, 1998.

[3]Antonio Barranquero Campos. Visualización de árboles SLD. Proyecto Fin de Carrera. Dpto. de Lenguajes y Ciencias de la Computación. Universidad de Málaga. 2002.

[4] Jan Wielemaker. SWI-Prolog/XPCE. University of Amsterdam.

[5]Antonio Jesús Paredes García. JApplet para visualizar árboles de refutación SLD. Proyecto Fin de Carrera. Dpto. de Lenguajes y Ciencias de la Computación. Universidad de Málaga. 2002.

[6] Andrew J. Kennedy. Drawing Trees. Journal of Functional Programming, Cambridge University Press, Mayo 1996.

Sigraf: SImulador de GRAFos.

Judith Antolín Sendino
María Ruiz Ruiz
Escuela Politécnica Superior,
Universidad de Burgos
{jas00, mrr0012}@alu.ubu.es

Carlos Pardo Aguilar
Juan J. Rodríguez Díez
Dpto. de I. Civil, Lenguajes y Sistemas Informáticos
Universidad de Burgos
{cpardo, jjrodriguez}@ ubu.es

Resumen

En el presente artículo se presenta una herramienta de apoyo al aprendizaje en un campo tan importante de la informática como las Estructuras de Datos, centrándose en el diseño, desarrollo y aplicación del Tipo Abstracto de Datos (TAD) grafo para la resolución de diversos tipos de problemas.

Aunque se encuentran disponibles diversas animaciones de algoritmos sobre grafos, normalmente éstas se limitan a un solo algoritmo (e.g., Dijkstra), o a un solo problema (e.g., árboles expandidos mínimos). La posibilidad de introducir un grafo y poder ejecutar interactivamente toda una gama de algoritmos sobre el mismo, es claramente ventajosa con respecto a tener que introducir el mismo grafo para diversos algoritmos, utilizando una interfaz diferente para cada uno de ellos.

1. Introducción

La disponibilidad de herramientas que permitan al alumno trabajar con conceptos adquiridos en las clases teóricas, así como resolver dudas, es escasa, además las horas de prácticas de las que se dispone en la universidad son limitadas. Por ello, es importante la disponibilidad de herramientas que proporcionen un entorno de pruebas y desarrollo de las diferentes cuestiones que van aprendiendo, que redunde en un mayor aprovechamiento de las horas dedicadas a clases prácticas.

Se considera de especial interés el estudio de los grafos que se usan para modelar cualquier situación en la que existan elementos unidos con otros tales como redes de alcantarillado, redes de comunicación, circuitos eléctricos, entre otros. Una vez modelado el problema mediante un grafo se pueden hacer estudios sobre diversas propiedades, para ello se utilizan algoritmos concretos que resuelvan ciertos problemas. La teoría de grafos ha sido aplicada en el estudio de problemas que surgen en áreas diversas de las ciencias, como la química, la ingeniería eléctrica o la investigación operativa. El primer paso siempre será representar el problema como un grafo. En esta representación cada elemento, cada objeto del problema se representa mediante un nodo. La relación, comunicación o conexión entre los nodos da lugar a una arista, que puede ser dirigida o bidireccional (no dirigida) [1].

En el ámbito de asignaturas de Estructuras de Datos, que tienen gran carga teórica y aplicación práctica, se pueden encontrar demostraciones de algunos algoritmos sueltos [3], [6]. Sin embargo, no se ha encontrado disponible ninguna herramienta que abarque aquéllos que normalmente se incluyen dentro de estas asignaturas. La utilización de una herramienta distinta para cada algoritmo es incómoda, puesto que cada una de ellas tiene su peculiar manera de editar un grafo. Por ello se planteó el desarrollo de una aplicación para el estudio de este TAD y de varios de sus algoritmos. Aunque sería deseable disponer de un entorno de visualización de todas las estructuras de datos habituales, entendemos que la cantidad de algoritmos que se pueden realizar sobre grafos justifica una herramienta para su estudio. Por otro lado, dentro de las estructuras de propósito general, la de grafo es la más compleja, así que se considera que es el primer paso más adecuado hacia un sistema de animación de cualquier estructura de datos.

El resto del artículo se organiza del siguiente modo. La sección 2 presenta los objetivos planteados. La herramienta se presenta en la

sección 3. Finalmente se exponen las conclusiones en la sección 4.

2. Objetivos

El objetivo era crear una aplicación que permitiera diseñar grafos, guardarlos, modificar otros ya existentes, imprimirlos y ver de forma didáctica y atractiva los distintos algoritmos que se estudian sobre ellos. Todo ello de forma gráfica y fácil de utilizar para ayudar en su comprensión a los alumnos.

La herramienta se ha desarrollado utilizando el paradigma Orientado a Objetos, ya que facilita la reutilización de las aplicaciones y es probable que esta herramienta en un futuro sea ampliada en posteriores proyectos finales de carrera u otros trabajos de investigación

Uno de los aspectos que se tuvo en cuenta fue la necesidad de diferenciar entre dos modos de funcionamiento del grafo, por un lado el grafo abstracto que es el que se encarga de modelar la realidad y sobre el que se ejecutan los algoritmos, y por otro lado las propiedades gráficas para poder representarlos en la pantalla (posición, color, etc.).

Como consecuencia se generaron dos clases para cada modo de funcionamiento, con las relaciones mostradas en la Figura 1.

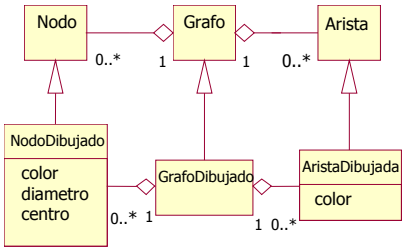


Figura 1: Diagrama de clases

3. La herramienta

Para la realización de esta aplicación se ha estudiado el comportamiento de otros programas en situaciones específicas, intentando obtener un producto con el que el usuario estuviera familiarizado y su uso fuese lo más intuitivo posible, e intentando adaptar su modo de trabajo y sus funcionalidades a un entorno como el de la enseñanza, así como mejorar los aspectos más importantes. Se consideró relevante incrementar al máximo la disponibilidad y portabilidad de la aplicación, estando disponible ésta en la página web de la asignatura para que los alumnos puedan descargarla cuando deseen, y pudiendo ser ejecutada en distintos sistemas operativos, ya que el hecho de estar implementada en Java, lenguaje robusto, seguro y de gran portabilidad ayuda a la obtención de estas características.

La aplicación consta de una interfaz gráfica sencilla y fácil de manejar, que permite al usuario trabajar con esta estructura de datos de forma rápida y eficaz (ver Figura 2). Sobre este aspecto se ha de comentar uno de los requisitos más relevantes, la facilidad y comodidad de manejo que incluye que todas las operaciones de la aplicación se podrán realizar tanto por ratón como por teclado. Se debe tener en cuenta que si además del temario de la asignatura y de una herramienta específica, el alumno debe aprender a trabajar con otra adicional, ésta debe intentar facilitar su trabajo sin añadir complejidad.

Desde el menú principal, se puede acceder en cualquier momento a la ayuda de la aplicación para obtener información sobre cualquier aspecto relacionado, tanto con la forma de diseñar un grafo como con la aplicación de los algoritmos con la herramienta, además de las explicaciones teóricas sobre la estructura para facilitar su aprendizaje. Para todas estas funcionalidades se proporciona una barra de herramientas con botones significativos para cada opción (ver Figura 3).

Durante todo el proceso de desarrollo y modificación del grafo se realizan diversas comprobaciones, evitando por ejemplo que se dibujen nodos solapados, que antes de poder dibujar una arista existan los nodos que esta desea unir, que cuando se borra un nodo se borren todas sus aristas asociadas. En la barra de estado de la aplicación se indican mensajes de error sobre las comprobaciones anteriores, se visualizan las coordenadas del puntero de ratón en todo momento y si se selecciona un elemento del grafo pueden obtenerse sus propiedades.

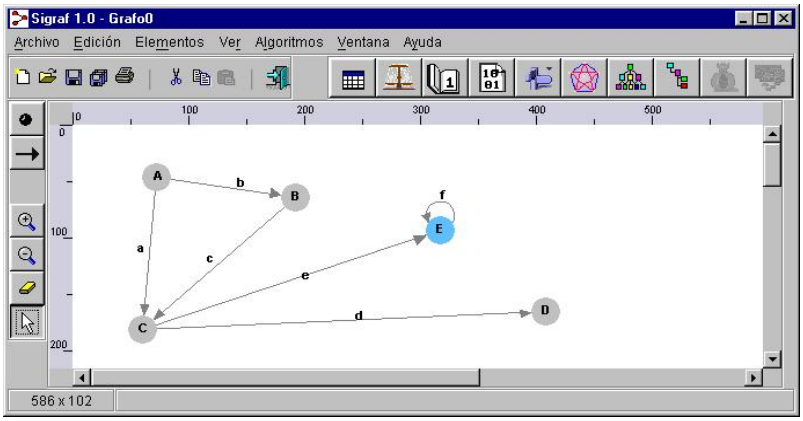


Figura 2: Interfaz de la aplicación.

Figura 3 Barra de herramientas de Dibujo

3.1. Representaciones de los grafos

Dentro de las diferentes visualizaciones de los grafos se encuentran la matriz de adyacencias, la matriz de costes y la lista de adyacencias. Cada una de estas puede obtenerse a través del menú *'Ver'* de la aplicación, desde el cual también podemos configurar qué propiedades de las aristas del grafo se desean mostrar y obtener más información sobre cada uno de los nodos, del grafo.

3.2. Ejecución de los algoritmos sobre grafos

La ejecución y visualización de los algoritmos, igual que el resto de opciones de la herramienta, se puede realizar a través del menú o desde la barra de herramientas específica de estos. Dichas visualizaciones se realizan en una ventana auxiliar, bien de forma gráfica mostrando los cambios mediante colores sobre el propio grafo, bien con los datos obtenidos(matrices de caminos, caminos mínimos…) o con ambas cosas según el algoritmo.

En la mayoría de ellos puede realizarse su ejecución paso a paso para seguir la evolución del algoritmo de forma más didáctica para el alumno.

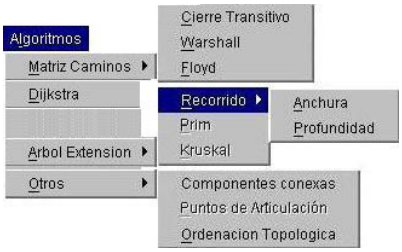


Figura 4 Menú de algoritmos.

3.3. Tratamiento de ficheros

Para el almacenamiento de los grafos se generan ficheros de texto con un formato específico y extensión ".grf" para identificarlos. Las operaciones sobre éstos (crear, abrir, cerrar,

guardar...), se realizan a través del menú *'Archivo'* o desde la barra de herramientas estándar. Al permitir tener varios ficheros abiertos a la vez, se lleva un control de los que han sido modificados y se proporciona la posibilidad de cambiar de uno a otro desde el menú *'Ventana'*.

3.4. Ejemplo de ejecución de un algoritmo.

El *cálculo de las componentes conexas* de un grafo consiste en obtener aquellos subconjuntos de nodos entre los cuales la comunicación es total. En este caso no se proporciona la opción de realizarlo paso a paso, ya que no se cree necesario para que el alumno comprenda el desarrollo del mismo. Un grafo no dirigido G es conexo si existe un camino entre cualquier par de nodos que forman el grafo. En el caso de que el grafo no sea conexo se pueden determinar todas las componentes conexas del mismo

En un grafo dirigido podemos distinguir entre grafo dirigido conexo y grafo fuertemente conexo. Un grafo dirigido es conexo si para cada par de vértices existe una cadena que los une. Y un grafo dirigido es fuertemente conexo si para cada par de vértices existe un camino que los une. Por tanto, para saber si un grafo es o no conexo se calculan sus componentes, si tras visitar todos sus nodos se obtienen una única componente se dice que el grafo es conexo.

La visualización realizada para este algoritmo consta de una pantalla dividida en dos paneles. En el de la izquierda se visualiza el grafo formado por las componentes conexas obtenidas, mientras que en la parte derecha se muestran los nodos que forman la componente elegida o todo el grafo. Para facilitar la comprensión al alumno, se realiza una asociación de colores usando el mismo para una componente en el panel de la izquierda, que para los nodos por los que está formado en la parte derecha.

4. Conclusiones y trabajos futuros

Debido a que la herramienta se ha desarrollado recientemente y dada la situación temporal del tema dedicado a grafos al final de la asignatura de Estructuras de Datos, es demasiado pronto como para establecer conclusiones respecto a su utilidad. No obstante, se considera que aunque sólo sea como "transparencias interactivas" la herramienta es útil. Queda por determinar hasta que punto los alumnos utilizan la herramienta por su cuenta, y su valoración de la misma.

Dentro de las posibles ampliaciones, algunas de las cuales ya están en marcha, se pueden comentar las siguientes. Por un lado, las mejoras en la capacidad de edición y visualización, como mostrar la traza del algoritmo además del grafo, permitir arcos curvados o colocar automáticamente los nodos y arcos. Para este último aspecto se planea utilizar alguna herramienta como Graphviz [4]. El formato de fichero utilizado para almacenar los grafos es actualmente un formato propio, por lo que parece interesante incluir la posibilidad de importar y exportar a otros formatos, por ejemplo GraphML [2]. Aunque en la versión actual están disponibles bastantes algoritmos, este apartado se puede completar, con por ejemplo, cuestiones relativas a flujo en redes o problemas NP.

Referencias

[1] Mark Allen Weiss "Estructuras de datos en Java". Adisson-Wesley.

[2] U. Brandes, M. Eiglsperger, I. Herman, M. Himsolt, and M.S. Marshall: GraphML Progress Report: Structural Layer Proposal. Proc. 9th Intl. Symp. Graph Drawing (GD'01), LNCS 2265, pp. 501-512. Springer-Verlag, 2002.

[3] Peter Brummund, Ngozi V. Uti. "The Complete Collection of Algorithm Animations". http://www.cs.hope.edu/~alganim/ccaa/

[4] Emden R. Gansner "Drawing graphs with GraphViz" 2003. http://www.research.att.com/sw/tools/graphviz/libguide.pdf

[5] Luis Joyanes Aguilar, Ignacio Zahonero Martinez "Estructuras de datos. Algoritmos, abstracción y objetos". Mac Graw Hill 1999.

[6] M. Gloria Sánchez Torrubia, Víctor M. Lozano Terrazas. "Algoritmo de Dijkstra. Un tutorial interactivo". VII Jornadas de Enseñanza Universitaria de la Informática (JENUI 2001)

Autoevaluación a través de Internet por medio de test

Pedro A. Castillo Alberto Prieto Antonio Cañas Beatriz Prieto

Dpto. de Arquitectura y Tecnología de Computadores
Universidad de Granada
18071 Granada
e-mail: pedro@atc.ugr.es

Resumen

En este trabajo se presenta un sistema automático de *generación* y *corrección* de exámenes tipo test (SAGET). El sistema selecciona aleatoriamente de una base de datos un conjunto de preguntas tipo test, con varias opciones de respuesta; tanto el número como su índice de dificultad son elegidos por el usuario. Una vez que el alumno contesta el test, éste es corregido automáticamente, y se muestra nuevamente en pantalla el cuestionario destacando tanto las respuestas del usuario como las respuestas correctas. El sistema es útil en una doble vertiente: para que el alumno se autoevalúe desde cualquier lugar donde tenga acceso a Internet, y para que el profesor genere automáticamente cuestionarios de test para examinar a sus alumnos.

1. Introducción

Los tests se presentan como una herramienta de gran utilidad para facilitar el aprendizaje, y de hecho han sido utilizados ampliamente y desde sus orígenes en numerosos sistemas CAI (*Computer Assisted Instruction*) o CAL (*Computer Aided Learning*).

Por otra parte, una de las tareas más tediosas de la enseñanza es la evaluación de los alumnos. Por lo general, la evaluación de las asignaturas de materias científico-técnicas se efectúa por medio de ejercicios de *problemas* y por *prácticas* de laboratorio, complementados con ejercicios de test. Los exámenes tipo test presentan una gran dificultad en su diseño [1] (elección de las preguntas y las opciones de respuestas posibles) ya que implican seleccionar las cuestiones más adecuadas a los objetivos de medición establecidos; sin embargo, se adaptan muy bien a una corrección completamente objetiva y automática, ventajas que no se obtienen con otras modalidades de exámenes.

En la bibliografía podemos encontrar trabajos en los que se proponen sistemas para corrección automática de exámenes tipo test [2], sin embargo, este trabajo se centra en conseguir una mayor versatilidad y comodidad de uso del sistema, realizándose todas las fases de forma automática, y en la introducción de un índice de dificultad asociado a cada cuestión.

1.1. Objetivos del trabajo

Los principales objetivos y características de SAGET son:

- Generación automática de exámenes de tipo test para cualquier asignatura.
- Selección por parte del usuario del número de preguntas que componen el test.
- Selección de un nivel de dificultad por parte del alumno o profesor.
- Corrección y evaluación automáticas *on-line*, bien para realizar exámenes reales, o como apoyo al estudio realizando el alumno ensayos a través de Internet (autoevaluación).
- Generación de la plantilla de corrección que ayude al profesor en una corrección manual.
- Adaptación de la dificultad de cada pregunta de acuerdo al número de veces que se haya contestado correcta, incorrectamente, o se haya dejado sin contestar.

Todo el sistema se ha desarrollado de forma que sólo hace falta una conexión a Internet para acceder al servidor donde se encuentra SAGET y tener un navegador en el que visualizar los documentos HTML.

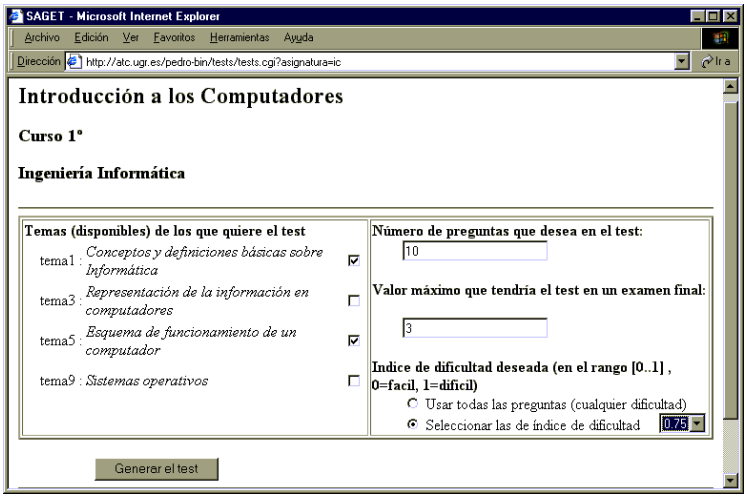


Figura 1. Formulario en el que seleccionaremos los temas de los que queremos realizar el test, el número de preguntas y la dificultad de éstas.

A continuación describiremos en detalle el funcionamiento del sistema y la implementación llevada a cabo.

2. Descripción del funcionamiento del sistema

El funcionamiento esquemático del sistema queda descrito en los siguientes puntos:

1. El estudiante, a través de la página principal del sistema selecciona la asignatura sobre la que desea el test.
2. El sistema obtiene ciertos datos sobre esa asignatura para mostrar un primer formulario HTML (Figura 1) en el que se solicitan los temas a incluir en la evaluación, el número de preguntas en el test, y el nivel de dificultad (de 0 a 1) deseados. A continuación, el estudiante podrá pedir la generación del test y pasar a resolverlo.
3. El sistema recibe el número de preguntas y la dificultad de éstas, y extrae de los temas especificados, de forma aleatoria, las preguntas que se ajusten a dicha dificultad. Devuelve al navegador cliente un formulario HTML con las preguntas y botones asociados a cada una de las cuatro respuestas posibles (Figura 2).
4. Una vez contestadas, el servidor recibe las respuestas y comprueba la corrección o incorrección de cada una de ellas.
5. El servidor vuelve a presentar el test, pero ahora destacando en color rojo las respuestas incorrectas dadas por el usuario y en verde las respuestas correctas (Figura 3). Con esta revisión personalizada el estudiante tiene la oportunidad de conocer qué preguntas son correctas y aprender de sus errores.

3. Implementación del sistema

SAGET se ha concebido como una herramienta que pueda ser utilizada por profesores de distintas asignaturas y titulaciones (no sólo de Informática).

A cada titulación se le asocia una base de datos distinta, dentro de cada una de las cuales se incluyen carpetas para las diferentes asignaturas. Cada carpeta de una asignatura se compone de diferentes archivos, uno de ellos describe la

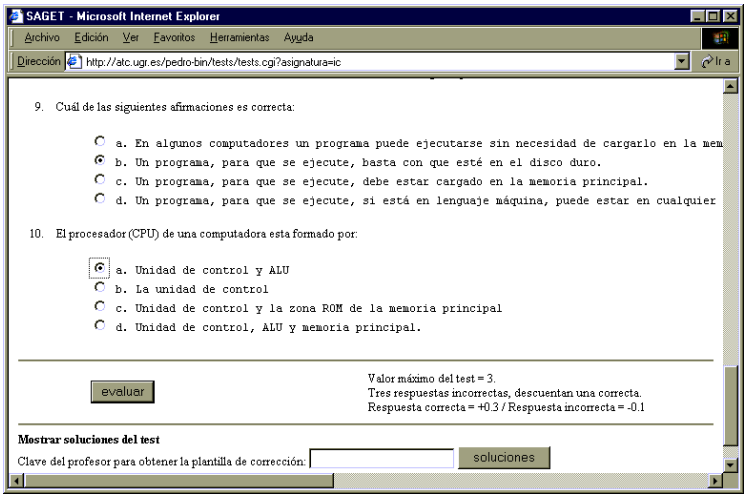


Figura 2. Formulario que muestra las preguntas del test (cuatro opciones por pregunta) y las opciones para evaluar las contestaciones o para generar la plantilla de corrección (opción para el profesor).

asignatura (curso, titulación, número de temas disponible) y los otros corresponden a cada uno de los temas de las asignaturas incluyendo las distintas cuestiones de cada uno de ellos (enunciado, opciones, contestación correcta, e índice de dificultad). Para añadir un tema a una asignatura, simplemente se añade un nuevo archivo, con las cuestiones de dicho tema, a la carpeta correspondiente a dicha asignatura.

Otra característica destacable de SAGET es poder especificar el nivel de dificultad deseado para el test. Así, según la preparación del estudiante (o del nivel que se requiera en el examen) se buscarán preguntas más o menos difíciles (que hayan sido contestadas más veces incorrectamente o se hayan dejado sin contestar).

Para establecer el índice de dificultad de un test, el sistema asocia a cada cuestión un índice de dificultad, que varia entre 0 y 1, y que es establecido y adaptado continua y automáticamente por el sistema, en función de las contestaciones recibidas previamente por el sistema. El índice de dificultad de una cuestión tiene el valor 0 (fácil) al principio, y se actualiza cada vez que es planteada, de acuerdo con la siguiente expresión:

$$i = \frac{n_{erroneas}}{n_{planteadas}}$$

Donde $n_{erroneas}$, y $n_{planteadas}$ son, respectivamente, el número de veces que dicha cuestión ha sido contestada erróneamente y el número de veces que el sistema la ha planteado a los alumnos. El sistema elige las cuestiones aleatoriamente pero tratando de que el índice de dificultad media de las cuestiones propuestas se aproxime al máximo al deseado por el usuario.

La puntuación del test se lleva a cabo con una corrección estadística, tratando de que si se contestase aleatoriamente a las cuestiones, la puntuación media fuese 0. Así, la ecuación que calcula la puntuación final, p, del test es:

$$p = \left(N_{correctas} - \frac{N_{incorrectas}}{3} \right) \times \frac{V_{\max}}{N_{cuestiones}}$$

donde V_{max} es la puntuación máxima del test, $N_{cuestiones}$ es el número total de cuestiones incluidas en el mismo, y $N_{correctas}$ y $N_{incorrectas}$ son el número total de cuestiones contestadas correcta e incorrectamente.

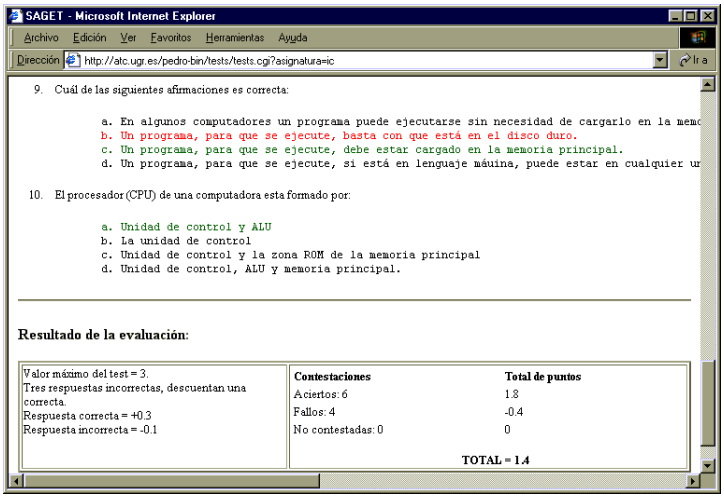


Figura 3. Página de evaluación en la que se muestran las correcciones a las contestaciones incorrectas. Las estadísticas muestran el total de puntos por los aciertos, fallos y no contestadas.

El sistema ha sido implementado teniendo en cuenta su portabilidad y facilidad de ampliación con nuevas asignaturas. Por ello se usa un programa CGI escrito en Perl. En la actualidad se utiliza un conjunto de archivos de texto para almacenar las preguntas, aunque está desarrollándose una base de datos relacional y la interfaz necesaria para mantenerla más fácilmente.

El sistema se está aplicando a la asignatura de Introducción a los Computadores, impartida en diversas titulaciones de la Universidad de Granada, y que sigue el texto indicado en la referencia [3]. Las preguntas que conforman la base de datos actual han sido diseñadas por los profesores que imparten dicha asignatura.

4. Conclusiones y trabajo futuro

Se ha descrito un nuevo sistema totalmente automático de generación de test, con preguntas y respuestas seleccionadas aleatoriamente de una base de datos, pudiendo el usuario seleccionar los temas a incluir, el número de preguntas, el índice de dificultad y la puntuación máxima dada al test. SAGET se ha proyectado como ayuda para el estudio, de forma que los estudiantes puedan autoevaluarse desde cualquier ordenador con acceso a Internet, y como ayuda al profesor para el diseño de exámenes de test. El sistema ha tenido una buena acogida entre los alumnos que lo han utilizado ampliamente para preparar la asignatura en el primer cuatrimestre.

Como mejoras al sistema, está previsto añadir nuevas asignaturas y, algo que ya está siendo implementado, añadir el módulo de identificación de los alumnos para examinarlos en aulas de PC conectados a Internet.

Referencias

[1] J. Muñiz, *Teoría clásica de los test,* Pirámide, 1998.

[2] N. Pavón, José R. Cano, F. Márquez, A. Sainz. SCRAE'Web: *Sistema de Corrección y Revisión Automática de Exámenes a través de la WEB.* JENUI 2002, pp.231-235. 2002.

[3] A. Prieto, A. Lloris, J. C. Torres, *Introducción a la Informática,* 3ª Ed., McGraw-Hill, 2002.

Una Componente "e-Learning" de Aprendizaje Colaborativo para el Proyecto IDEFIX

T. Hernán Sagástegui Ch., José E. Labra G., Juan M. Cueva L.
José M. Morales G., María E. Alva O., Eduardo Valdés, Cecilia García

Departamento de Informática
Universidad de Oviedo
C/ Calvo Sotelo S/N,
33007 Oviedo
e-mail: thsagas@correo.uniovi.es, labra@lsi.uniovi.es, cueva@lsi.uniovi.es, jmmoral@correo.uniovi.es, malva360@correo.uniovi.es, eduardovr@telecable.es, cgpelayo27@hotmail.com

Resumen

Este artículo presenta el diseño y la arquitectura de implementación de un modelo de aprendizaje colaborativo a través de Internet. El sistema dará soporte a una experiencia piloto que se está desarrollando en la asignatura de lógica de la EUITIO de la Universidad de Oviedo, permitiendo el desarrollo colaborativo de ejercicios en "e-reuniones" a grupos de alumnos y el entrenamiento en la resolución de exámenes mediante un juego de realidad virtual. Se basa en la utilización de servicios Web como una componente del proyecto IDEFIX.

1. Motivación

El proyecto IDEFIX (*Integrated Development Environment Frameworks based on Internet and eXtensible technologies*) se centra en el desarrollo de entornos integrados de desarrollo a través de Internet [1]. IDEFIX puede extender su dominio a la enseñanza por Internet de cualquier asignatura de la Escuela Universitaria de Ingeniería Técnica de la Universidad de Oviedo EUITIO, así como a la enseñanza compartida de asignaturas de libre elección a través de Internet del proyecto AulaNet [2]. En este contexto, en este artículo se presenta una componente del proyecto IDEFIX, referida al diseño y arquitectura de un modelo de aprendizaje colaborativo, cuyo objetivo es apoyar la imparticion de la asignatura de lógica de la EUITIO a través de la Web.

Actualmente la asignatura de lógica se imparte en el primer año de la carrera de ingeniería informática de la EUITIO, las clases se dan presencialmente y como material pedagógico de apoyo se dispone de una guía didáctica y de un cuaderno que recoge los exámenes realizados en la asignatura los años anteriores que sirven como ejercicios.

Se desea que los alumnos de la asignatura de lógica puedan acceder de forma remota al cuaderno que recoge los exámenes /ejercicios y puedan resolverlos en grupo a través de la Web, bien desde los laboratorios disponibles en la facultad, o desde sus casas, con la ventaja de la libertad total de horario. Se tienen los siguientes requisitos:

a) Implementar el desarrollo colaborativo de los ejercicios del cuaderno de exámenes a través de la Web.

b) Motivar a los alumnos a realizar un entrenamiento para los exámenes. Para ello se ha propuesto la creación de una especie de juego mediante realidad virtual en Internet.

2. Antecedentes

El aprendizaje colaborativo asistido por ordenador (CSCL) apoya el trabajo en grupo de una tarea común, proporciona una interfaz compartida para

que el grupo trabaje en ella [3,4,12] y usa la tecnología de ordenadores como herramienta que ayuda a los aprendices a comunicarse y colaborar en actividades conjuntas, mediante una red de ordenadores, apoyando la coordinación, y la aplicación del conocimiento en cierto dominio [5,6]. CSCL se usa en el ambiente educativo y sirve de soporte a los estudiantes en el aprendizaje, facilitando el proceso de trabajo en grupo y la dinámica de grupos de una manera que no se lograría cara a cara [3].

Existen numerosas aplicaciones colaborativas según la respectiva taxonomía de aplicaciones y el tipo de actividad de aprendizaje que apoyan: tutoriales, resolución de problemas, simulaciones [9], debates, modelación y CourseWare (Blackboard: Virginia Commonwealth University, LearningSpace: Lotus & IBM [11], WebCT: University British Columbia, TopClass: WBT Systems [10], etc). Para el aprendizaje de la lógica existen proyectos como JAPE [7,8].

Una aplicación colaborativa dentro de un esquema común de trabajo en grupo posee las siguientes características: memoria grupal, roles, protocolos de colaboración y percepción [3,5]. La memoria grupal es el espacio común donde los miembros de un grupo almacenan información en forma ordenada referente al desarrollo de la actividad [3,5]. Este espacio es creado con la finalidad de proveer al grupo de un dispositivo efectivo de comunicación y es el resultado tanto del proceso de trabajo como del producto final conseguido por el grupo [3,5]. Un rol es un conjunto de privilegios y responsabilidades atribuidas a una persona o a un módulo del sistema (agente). Los roles entre los miembros del grupo pueden ser implícitos o explícitos, para hacer más eficiente y coordinado el logro de los objetivos [3,5]. Los protocolos de colaboración son las distintas maneras de interactuar de las personas consensuadas por el grupo. Son reglas que permiten a los individuos comunicarse entre sí de tal forma que cada uno pueda enviar y recibir señales comprensibles para los demás [3,5]. La percepción es toda información que provee una conciencia grupal al individuo que forma parte de un grupo. En el contexto del objeto de la información tenemos "percepción de usuarios" y "percepción de datos". La percepción de usuarios provee información sobre los miembros del grupo, informa de quiénes están conectados y lo que éstos hacen. La percepción de datos suministra información referente a los cambios efectuados sobre los datos [3,5].

3. Modelo de Aprendizaje Colaborativo

El modelo de aprendizaje colaborativo para el aprendizaje de la lógica está basado en el grupo y en las e-reuniones del grupo, en un entorno distribuido, pudiéndose diseñar un ciclo de vida de las e-reuniones [3,4,5,6]. Las componentes elementales del modelo de aprendizaje colaborativo de la lógica a través de la Web, son:

* **Memoria Grupal:** conformada por la base de datos de alumnos de la universidad (enlaces a asignaturas y profesores), la base de ejercicios / exámenes y la base de diseño de juegos.
* **Roles:** rol del profesor, rol del alumno que desarrolla ejercicios, rol del alumno que da examen, rol del usuario invitado.
* **Protocolos de Colaboración:** son las Reglas establecidas por el profesor, reglas del desarrollo de los ejercicios, reglas para rendir un examen, reglas para los usuarios invitados, reglas del administrador.
* **Percepción:** es la información generada en el desarrollo de las prácticas y de los exámenes; que los miembros del grupo conectados desarrollan, los resultados obtenidos, la generación de ideas, la toma de decisiones. Una posible herramienta para la percepción es el chat.
* **Interfaz:** el usuario interactúa con la memoria grupal y con el conocimiento generado a través de la interfaz, permitiendo la selección del trabajo siguiente; a) el desarrollo colaborativo de los ejercicios del cuaderno de exámenes a través de la Web o, b) la realización de los exámenes de entrenamiento a través de la Web.

4. Arquitectura del Modelo

La arquitectura para la implementación del modelo de aprendizaje colaborativo de la lógica a través de la Web está basado en el esquema cliente/servidor de la Figura 1. Se está implementando como una componente del proyecto IDEFIX sobre la plataforma .NET de Microsoft.

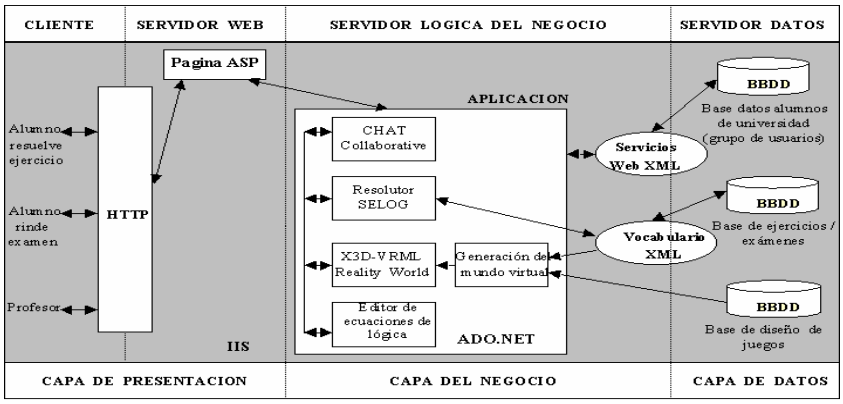


Figura 1. Arquitectura de implementación

4.1. Capa de presentación

Esta capa presenta la interfaz de usuario y se encarga de la tarea de visualización de los clientes y de la entrada de datos. La característica básica de esta capa es que el usuario (alumno, profesor, invitado) es un cliente navegador que utiliza el protocolo HTTP para acceder a la aplicación.

4.2. Capa del negocio

Encargada de la lógica de la aplicación y de la lógica de acceso a los datos y constituida por:

*** Editor de ecuaciones de lógica:** similar al que incorpora el procesador de textos Microsoft Word, y que facilita la creación de las ecuaciones de lógica para almacenar en un vocabulario XML.

*** CHAT colaborativo:** chat de consulta y retroalimentación.

*** Resolutor SELOG:** permite el desarrollo colaborativo de los ejercicios de lógica del cuaderno de exámenes a través de la Web. Presenta una interfaz y realiza funciones como; selección del tipo de ejercicios, obtención de pistas, consulta y evaluación de la dificultad de las preguntas, solución y comentarios. Al finalizar la sesión el usuario obtiene el número de preguntas resueltas correctamente y erróneamente. SELOG enlaza el CHAT colaborativo para la comunicación y retroalimentación del desarrollo de los ejercicios en grupo.

*** Reality World:** está diseñado para dar soporte en el entrenamiento de la realización de exámenes. Reality World es un producto de realidad virtual en X3D y presenta un mundo de objetos. El micro-mundo de la lógica es un mundo formado por esferas que describen los niveles del juego. Para pasar de una esfera a otra hay que aprobar todas las fases del nivel correspondiente. Cada nivel corresponde a un capítulo de la asignatura y en este nivel las fases se corresponden con los contenidos del capítulo. En un nivel determinado se incluyen todos los ejercicios hechos en los anteriores niveles; si no se han hecho todos los ejercicios de los distintos niveles no se podrá entrar en este último nivel. Respecto al resto de niveles, el jugador puede ir cambiando de nivel en nivel, siempre y cuando haya finalizado una fase o no haya comenzado la siguiente.

El documento XML es un documento en un vocabulario XML que define la configuración de la asignatura y modela los ejercicios. A partir del documento XML y los datos del mundo virtual leídos de las bases de datos, un programa

generador del mundo virtual, escrito en C#, permite pasar de XML a X3D generando el mundo en X3D.

4.3 Capa de Datos

En esta capa se encuentran los datos del alumno y sus relaciones con los datos de la asignatura y del profesor, almacenados en una base de datos. Asimismo está creada una base de datos que almacena todos los datos de los exámenes, preguntas y conceptos. También está creada una base de datos de diseño que almacena los juegos y los datos que el generador del mundo virtual necesite.

5. Conclusiones y Trabajos Futuros

La componente de aprendizaje colaborativo del Framework IDEFIX aun esta en desarrollo y actualmente se tiene un prototipo de prueba (http://idefix.sourceforge.net) que se utilizará como apoyo a la enseñanza de la lógica. Para la observación de los beneficios se clasificará a los estudiantes en grupos de usuarios y en grupos de no usuarios y se comparará su rendimiento.

Este trabajo busca contribuir al desarrollo de aplicaciones colaborativas para la educación virtual y "e-learning" en la educación secundaria y superior, satisfaciendo necesidades edu-comunicativas y pedagógicas propias [13]. Asimismo busca acuñar y apoyar el desarrollo de aplicaciones colaborativas de educación virtual de uso libre. Las investigaciones futuras estarán dirigidas a implementar componentes CSCL específicas en la imparticion virtual de las asignaturas de la Escuela Universitaria de Ingeniería Técnica de la Universidad de Oviedo (EUTIO), como apoyo a los métodos presenciales.

Referencias

[1] Labra, J., Morales J., Fernández A., Sagastegui H.: *A Generic e-Learning Multiparadigm Programming Language System: IDEFIX Project.* SIGCSE'03, USA (2003)

[2] Pérez, R., López, A.: *Aulanet, una Experiencia de Aula Virtual*. Spain (2000)

[3] Ellis, C., Gibbs, S., Rein, G.: *Groupware some issues y experiences*, Comm. of the ACM, Vol. 34 No. 1 (1991) 38-58.

[4] Conklin, J.: *Capturing Organizational Memory. Readings in Groupware and Computer-Supported Cooperative Work*. Morgan Kaufmann Publishers, CA (1993) 561-565

[5] Guerrero, L. Fuller, D.: *CLASS: A Computer Platform for the Develpment of Education´s Collaborative Applications*. Proceedings of CRIWG´97, 3rd International Workshop on Groupware, Spaín (1997) 1-3.

[6] Gokhale, A.: *Collaborative Learning Enhances Critical Thinking*. Journal of Techonology Education, Vol. 7, Nº 1, Fall 1995, University Libraries. Virginia (1996)

[7] Aczel, J.: *The Evaluation of a Computer Program for Learning Logic: The Role of Students' Formal Reasoning Strategies in Visualising Proofs*. CALRG Technical report (2000) 192

[8] The Jape Visualisation Project http://iet.open.ac.uk/pp/j.c.aczel/Jape/index.html

[9] LEGO Mindstorms http://mindstorms.lego.com/eng/default.asp

[10] TopClass e-Learning Suite™ http://www.wbtsystems.com/products

[11] IBM/Lotus Software http://www.lotus.com/home.nsf/tabs/learnspace

[12] Lin, W.: CSCL Theories. Texas University. USA (1996) http://www.edb.utexas.edu/csclstudent/Dhsiao/theories.html

[13]Computer-Supported Intentional Learning Environments www.ed.gov/pubs/EdReformStudies/EdTech/csile.html