Necesidades específicas para la docencia de programación en un entorno virtual

Ma. Jesús Marco Galindo
Josep Prieto Blázquez

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

Resumen

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

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

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

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

1. Introducción

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

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

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

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

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

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

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

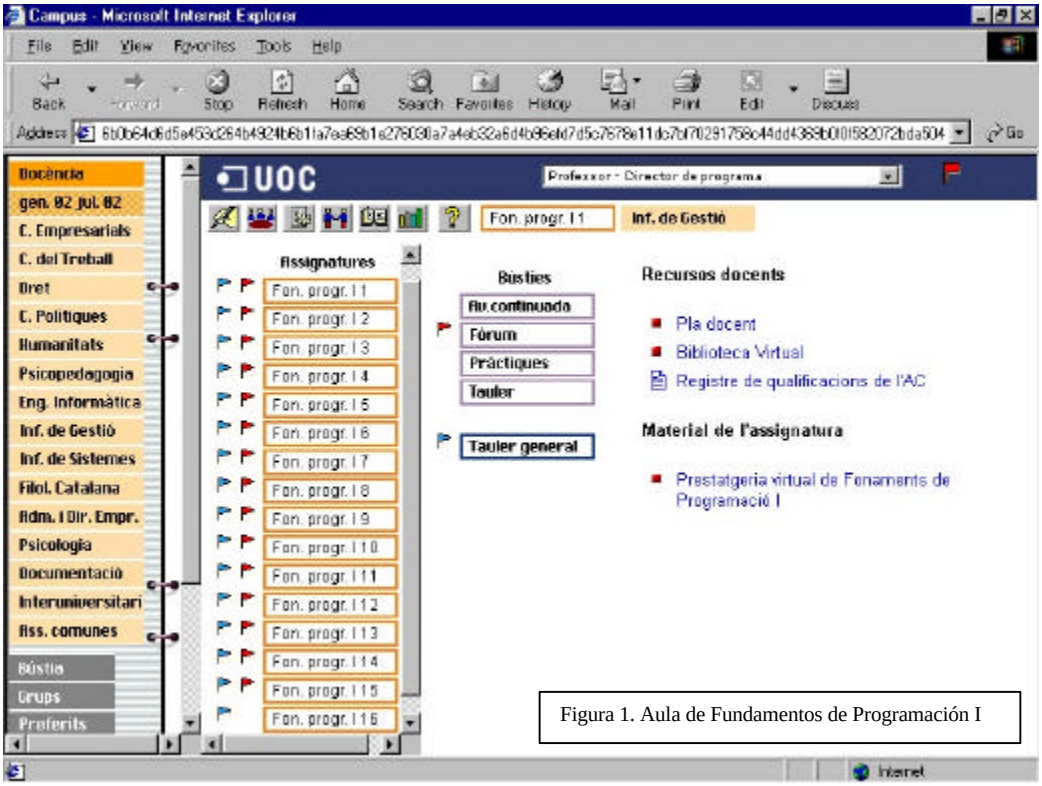

Figura 1. Aula de Fundamentos de Programación I

2. Entorno de la asignatura

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

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

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

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

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

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

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

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

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

2.1. El laboratorio virtual

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

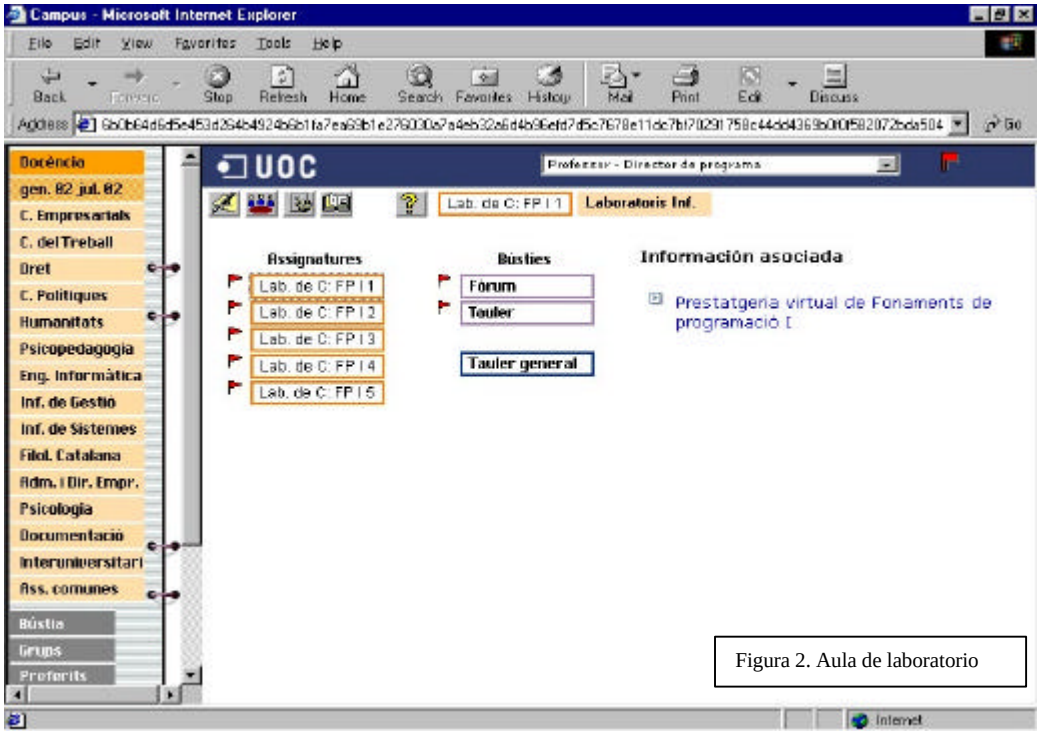

Figura 2. Aula de laboratorio

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

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

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

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

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

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

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

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

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

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

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

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

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

2.2. El corrector automático de programas

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

mucho tiempo de dedicación por parte del profesor.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

2.3. Simulador de algoritmos

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

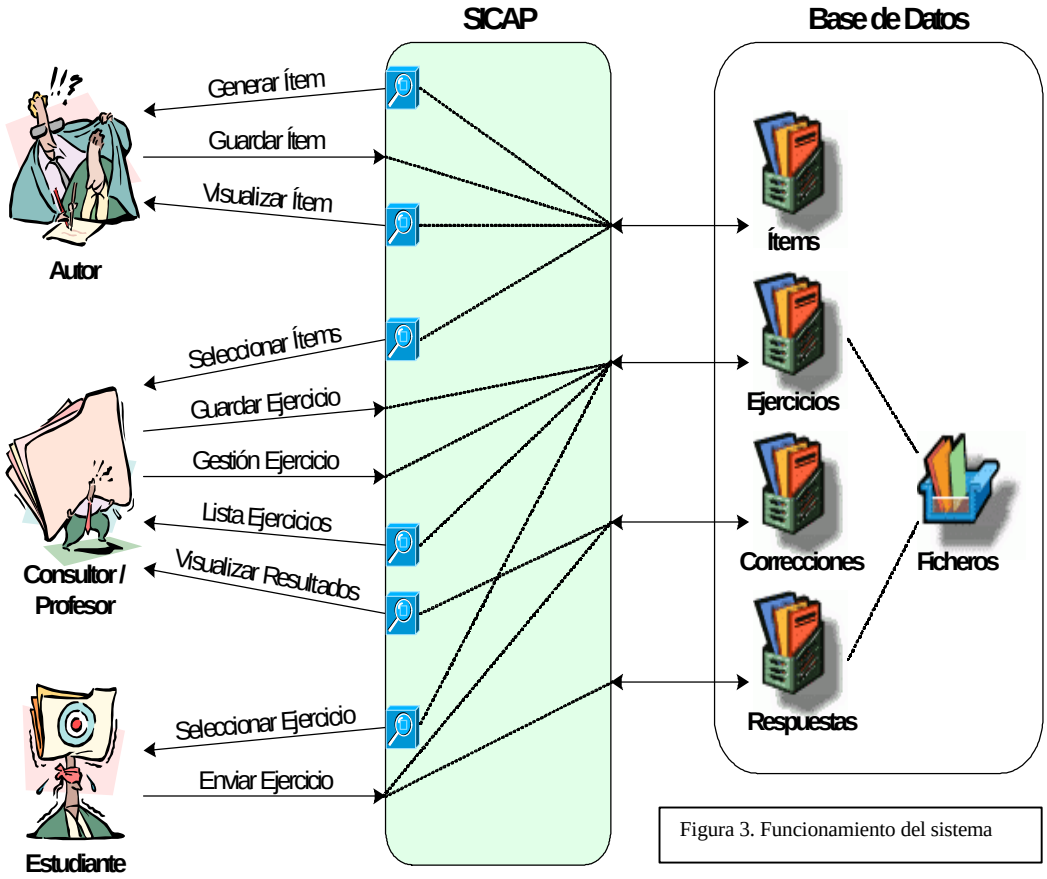

Figura 3. Funcionamiento del sistema

La figura 3 muestra el funcionamiento básico del sistema.

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

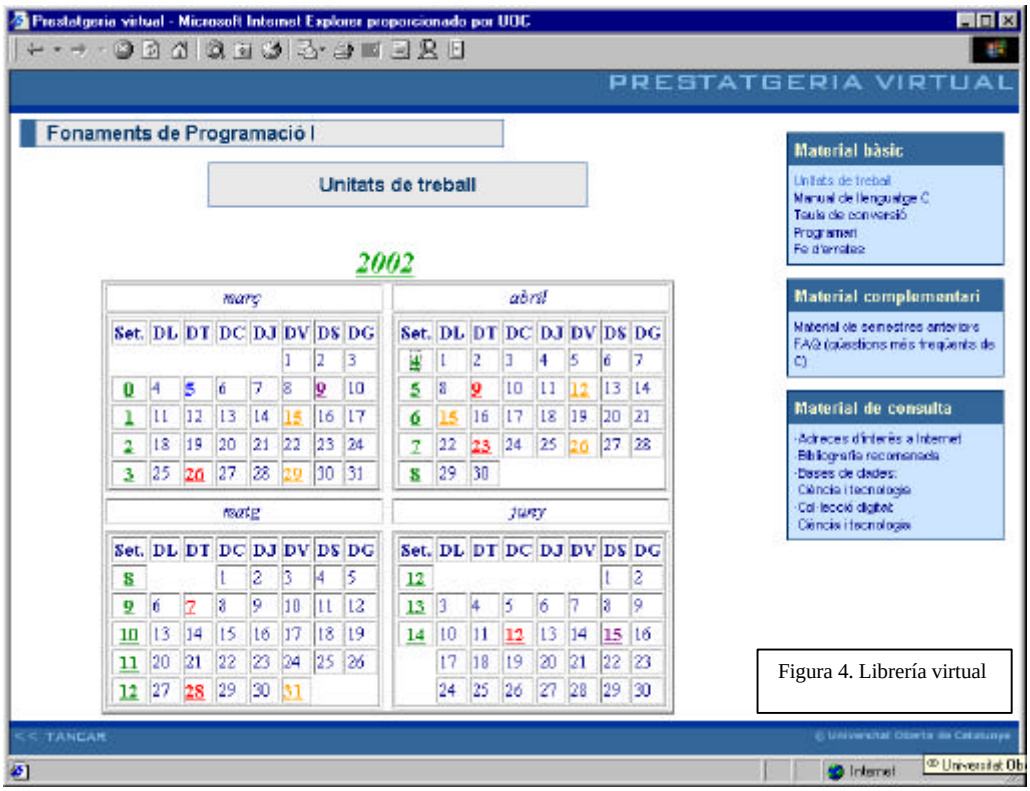

Figura 4. Librería virtual

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

Para ello, debe dar respuesta a las siguientes funcionalidades:

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

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

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

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

Conclusiones

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

Estas dificultades pueden sintetizarse en dos:

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

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

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

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

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

Referencias

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

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

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

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

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

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

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