Colecciones, correctores y generadores automáticos de ejercicios de programación

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

Resumen

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

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

1. Introducción

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

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

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

Las ventajas que puede proporcionar esta automatización son conocidas:

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

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

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

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

2. Definiciones

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

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

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

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

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

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

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

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

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

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

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

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

3. Herramientas de apoyo a ejercicios

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

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

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

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

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

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

3.1. Colecciones de ejercicios

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

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

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

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

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

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

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

3.2. Generación automática de ejercicios

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

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

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

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

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

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

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

3.3. Corrección automática de ejercicios

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

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

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

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

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

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

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

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

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

3.4. Algunas herramientas integradas

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

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

4. Evaluación

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

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

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

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

5. Conclusiones

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

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

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

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

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

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

Agradecimientos

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

Referencias

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

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

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

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

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

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

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

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

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

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

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