Actas de las XIX Jenui. Castellón, 10-12 de julio 2013
ISBN: 978-84-695-8051-6 DOI: 10.6035/e-TIiT.2013.13
Páginas: 343-350

Aprender, enseñar y evaluar con *CAP*, un Corrector Automático de tareas de Programación

Óscar Sapena Mabel Galiano Marisa Llorens Natividad Prieto
Departament de Sistemes Informàtics i Computació
Universitat Politècnica de València
46022 Valencia
{osapena,mgaliano,mllorens,nprieto}@dsic.upv.es

Resumen

En este artículo se presenta CAP, un Corrector Automático de tareas de Programación que ha sido diseñado combinando las TIC con un enfoque docente centrado en el estudiante y con el objetivo de hacer sostenibles las tareas de evaluación continua y seguimiento de las asignaturas de Programación en Java de los primeros cursos del Grado en Ingeniería Informática de la Universitat Politècnica de València. Asimismo, se presentan resultados sobre el uso de CAP y se compara con otras herramientas TIC; de ambos, se puede concluir que CAP ayuda al profesor a afrontar con objetividad y eficacia su labor docente y, al mismo tiempo, guía al alumno en el desarrollo de sus habilidades básicas como programador.

Abstract

This paper presents an automatic marking system to assess the learning process of novice students in programming in higher education. Combining ICT with a student centred approach to teaching, the CAP system has been designed, used and tested as an instrument to evaluate the learning outcomes defined for the Java introductory programming courses of Bachelor in Computer Engineering at the Universitat Politècnica de València. Results on its use and its comparison with other ICT tools are provided; from both, it can be concluded that CAP helps lecturers to face objectively and with efficacy their teaching and guides the student in the development of his basic programming skills.

Palabras clave

Corrección automática, evaluación continua, programación en Java, recurso TIC de aprendizaje.

1. Motivación

En este artículo se presenta CAP, un Corrector Automático de tareas de Programación, como una herramienta básica para hacer sostenibles las tareas de evaluación continua y seguimiento de tres asignaturas de programación en Java de los primeros cursos del Grado en Ingeniería Informática (GII) de la Escuela Técnica Superior de Ingeniería Informática (ETSInf) de la Universitat Politècnica de València (UPV). El objetivo general de estas asignaturas es que el alumno conozca la definición e implementación de los tipos de datos fundamentales, los esquemas algorítmicos iterativos y recursivos aplicables a la resolución de problemas de complejidad media y los criterios de eficiencia que conducen a la elección del tipo de datos y esquema algorítmico más adecuados para cada problema.

Este corrector ha contribuido positivamente a alcanzar dicho objetivo poniendo en práctica algunas de las ideas que, tanto con Bolonia como sin ella, los autores de este trabajo tenían sobre su labor docente y que podrían resumirse como sigue:

- La mejor forma de aprender a programar es programando.
- El alumno debe convertirse en el centro del proceso de enseñanza-aprendizaje. El profesor, si quiere asumir su papel en este nuevo esquema, debe aplicar nuevas estrategias para un seguimiento y evaluación objetivos y eficaces de sus alumnos.

Durante el primer curso de implantación del Grado, los autores ya habían experimentado con diversas técnicas que, en mayor o menor medida, implementaban estas ideas en el marco de Bolonia [9]; concluían entonces que era imprescindible el uso de las tecnologías de la información en los procesos de tutorización y evaluación del alumnado, si bien era necesario analizar y elegir entre el amplio abanico de metodologías docentes aquellas que no solo fueran efectivas sino que además pudieran ser usadas con confianza tanto por profesores como por alumnos. En este mo-

mento, los autores creen haber encontrado en CAP una herramienta para sustentar esta conclusión. La descripción de sus características así como su comparación con otras herramientas similares propuestas en la literatura se realiza en la sección 2; luego, en la sección 3, se presentan y analizan los datos que permiten justificar su interés; finalmente, en la sección 4, se extraen algunas conclusiones sobre CAP y se comentan las líneas de trabajo que, a medio y largo plazo, se piensan seguir para mejorarla.

2. Desarrollo y usos de CAP

El corrector CAP se presenta como una herramienta dual, pues pretende servir tanto al alumno como al profesor en un proceso asimismo dual como el de la enseñanza-aprendizaje. Por ello, tanto a nivel de diseño como de uso, se despliega en dos ventanas:

- La del administrador, donde el profesor diseña actividades para la consecución de los objetivos formativos de la asignatura y la evaluación de su enseñanza-aprendizaje; por motivos obvios, el alumno no tiene acceso a esta ventana.
- La del corrector propiamente dicho, a la que accede el alumno para realizar las tareas de programación propuestas, bajo la tutela subyacente del profesor que las ha diseñado.

En lo que sigue, se explican con detalle todos los aspectos que caracterizan esta dualidad de CAP y, además, se comparan con los de otros correctores para señalar las diferencias y similitudes que presentan entre sí, sus pros y contras: cuáles son sus principios de funcionamiento, cómo se han introducido y aplicado las estrategias de innovación por parte del profesor y, finalmente, qué ventajas proporciona su uso.

2.1. Principios de funcionamiento

Al igual que la mayoría de correctores automáticos existentes [6, 10, 11], el funcionamiento de la herramienta CAP, en líneas generales, consiste en compilar el programa y ejecutarlo, para ponerlo a prueba con unos cuantos juegos de datos, comparando el resultado obtenido con el previsto. Ahora bien, a nuestro parecer, CAP se diferencia de estos otros correctores en uno u otro de los siguientes aspectos:

- Compagina satisfactoriamente los criterios de seguridad con los de accesibilidad, facilidad de uso e interactividad, al mismo tiempo que ofrece al alumno una interfaz gráfica muy sencilla de usar y de respuesta amigable e inmediata para realizar los ejercicios en cualquier momento y tantas veces como desee. CAP garantiza implícitamente la seguridad del sistema en el que reside y explícitamente la del sistema del alumno que lo usa.
- Soporta con gran flexibilidad la definición de distintos tipos de ejercicios y proporciona al alumno una realimentación modulada, que puede variar a criterio del profesor que los diseña en función del tipo de ejercicio y de la calidad de la solución que cada alumno propone. Por ejemplo, si se trata de una actividad sumativa en la que se desea un bajo nivel de tutela, la salida del corrector puede ser la habitual de un oráculo (correcto/no correcto); en cambio, si se trata de una actividad formativa, el corrector puede desplegar una gran variedad de comentarios y sugerencias que guíen al alumno en la consecución del objetivo.

En buena parte, CAP posee estas dos características porque ha sido desarrollado como un applet Java. Ello no solo facilita su integración en cualquier sitio web sino que también, a diferencia de lo que sucede con otros lenguajes como C y C++, le permite hacer uso del mecanismo Java *reflection*[1], que permite obtener información sobre clases y objetos durante la ejecución de un programa. Así, con CAP se puede corregir cualquier tipo de programa, clase o método (completo o parcialmente implementado) desarrollado en Java sin exigir una disposición dada para sus datos y resultados, i.e. se pueden corregir programas que no requieran ningún dato de entrada o que no generen ningún tipo de salida o que exijan determinado y muy específico formato de entrada y/o salida o que no lo exijan.

Otras características importantes que presenta CAP son las siguientes: maneja una base de datos MySQL en la que se almacena tanto la biblioteca de tareas de programación como los resultados obtenidos por los alumnos; comparte el sistema de autenticación de la intranet de la UPV, lo que le permite identificar directamente al alumno que realiza el ejercicio, aunque también puede utilizarse sin que el usuario se haya identificado en la plataforma ("Usuario invitado") y, por tanto, sin guardar los resultados obtenidos.

Analizando las herramientas de corrección automática más representativas que se han propuesto hasta la fecha, y cuyas características se resumen en el cuadro 1, pensamos que "la hermana mayor" de CAP en lo que a principios de funcionamiento se refiere es WebToTeach [2], o más concretamente su actual versión comercial CodeLab[2]. Los únicos inconvenientes que se le podrían atribuir son los derivados de su diseño como CGI (*Common GateWay Interfaces*): interactividad restringida, problemas de seguridad y sobrecarga del servidor. No nos parece criticable que solo admita la resolución de micro-ejercicios pues, en base a

[1] `http://docs.oracle.com/javase/tutorial/reflect/`

[2] `http://www.turingscraft.com`

Herramienta	Completar código	Aplicaciones web	Realimentación personalizada	Análisis estático	Análisis dinámico	Calificación automática	Lenguajes
Automatic Marker	No	Módulo Moodle	No	No	Sí	Sí	Java
Backstop	No	No	Sí	No	Sí	No	Java
ELP	Sí	Applet	Sí	Sí	Sí	No	Java, C#, C
JAssess	No	Módulo Moodle	No	No	Sí	No	Java
WebBot	No	CGI	No	No	Sí	Sí	C/C++, Java, Perl, Python, Tlc, SPIM
Web-CAT	No	Servlet	Sí	Sí	Sí	Sí	Java
WebToTeach	Sí	CGI	Sí	No	Sí	No	C/C++, Java, Ada, Fortran, Pascal
CAP	Sí	Applet	Sí	Sí	Sí	Sí	Java

Cuadro 1: Comparación de herramientas de evaluación de ejercicios de programación.

la información disponible, parece ser que se debe a la estrategia educativa que implementa; además, a cambio, permite corregir ejercicios en varios lenguajes (C, C++, Java, etc.), flexibilidad que aún no posee CAP.

No podemos decir lo mismo de otros correctores implementados como aplicaciones web, a los que pensamos que CAP aventaja por uno u otro motivo. Así, por ejemplo,

- ELP [13], que permite corregir ejercicios en varios lenguajes y también es un applet, presenta una arquitectura bastante más compleja que la de CAP. Además, su interfaz de usuario no resulta demasiado amigable para principiantes de programación, pues no parece capaz de transmitirles de forma directa, asequible y (casi) instantánea el resultado de la completísima evaluación que realiza de un ejercicio.
- JAssess [14] y Automatic Marker [12], precisamente por haber sido diseñados como módulos de Moodle[3] para integrarse fácilmente en LMSes (*Learning Management Systems*) como Blackboard[4] o Sakai[5], presentan una interfaz de usuario más rígida y menos interactiva que CAP; el mismo problema, pero por otro motivo, se le puede achacar a Web-CAT [5], que además es un servlet.
- WebBot [4] es un CGI y, por tanto, se le puede aplicar perfectamente lo dicho previamente en este sentido sobre WebToTeach; además, en base a la bibliografía disponible, su interfaz de usuario no parece muy amigable en lo que se refiere a presentar resultados de evaluación a principiantes de programación, lastrada quizás por su escaso desarrollo y su origen como mero calificador de programas.

Finalmente, no podemos dejar de mencionar a Backstop [7], pues usa el mismo mecanismo que CAP, Java *reflection*, para proveer información de depuración de código Java muy completa; sin embargo, al ser una aplicación *stand-alone* no permite al profesor seguir el proceso de aprendizaje de sus alumnos.

2.2. Rol del alumno: aprendizaje autónomo y autoevaluación

Las figuras 1 y 2 son dos instancias de la ventana del alumno en CAP, la interfaz gráfica a través de la cual interactúan alumno y corrector. En cada una de ellas se ilustra una de las dos estructuras de ejercicio que el alumno puede realizar en CAP junto con su correspondiente modo de evaluación (supervisión y calificación): de *completar huecos* con evaluación tipo oráculo, que pretenden hacer "sentir" al estudiante todo el peso que las cuestiones sintácticas y el estilo tienen en la programación; de *implementación* con evaluación tipo profesor-virtual, que pretenden que el alumno aprenda a programar sin cometer errores (sobre todo lógicos y de ejecución) y de forma eficiente.

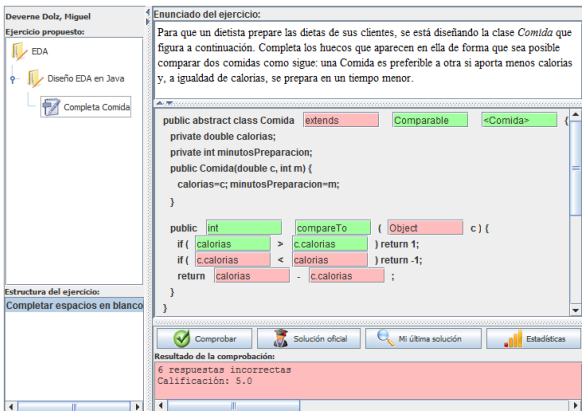

Figura 1: Ventana de CAP para el alumno: ejercicio de completar huecos.

Dicho esto, y como se puede observar en ambas figuras, la ventana del alumno consta de dos partes. En la de la izquierda figuran: el nombre del estudiante si está identificado en PoliformaT [3], la plataforma de la UPV, o "Usuario invitado" si no lo está; la asignatura y el tema en el que se sitúa el ejercicio; la lista de clases

[3] `http://www.moodle.org`
[4] `http://www.blackboard.com`
[5] `http://www.sakaiproject.org`

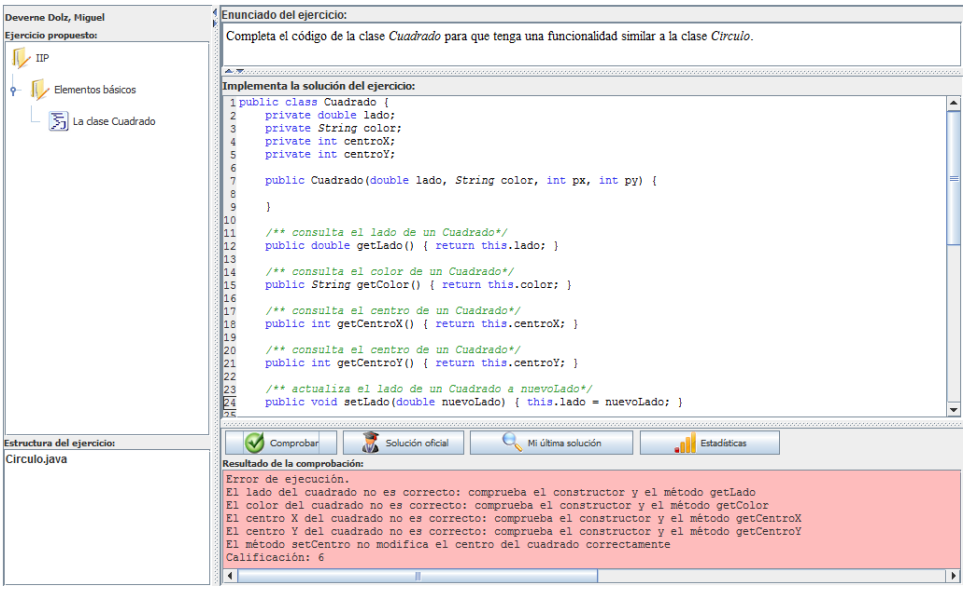

Figura 2: Ventana de CAP para el alumno: ejercicio de implementación.

adicionales que el alumno puede consultar si se trata de un ejercicio de implementación como el de la figura 2, ninguna si se trata de un ejercicio de completar huecos como el de la figura 1.

En su parte derecha se distinguen tres cuadros de texto (enunciado del problema, editor de código solución para el alumno y cuadro de evaluación/calificación de la solución presentada) y una barra de botones que permiten, a petición del alumno, reflejar su diálogo con el corrector. Así, cuando el alumno pulsa el botón:

- "Comprobar"... en el cuadro de evaluación aparece al instante, sobre fondo rojo, un mensaje que le indica los errores detectados en su código y la nota que –siempre sobre 10– le otorga el corrector; si el código fuera correcto y eficiente, sobre fondo verde aparece el mensaje "¡¡¡Código correcto!!! Calificación: 10".
  Además, como se puede deducir al comparar los cuadros de evaluación en las figuras 1 y 2, los mensajes de error de CAP son, intencionadamente, más o menos prolijos según la estructura/modo de evaluación del ejercicio; detallaremos la información que contienen, clave en el diseño de CAP, al concluir la descripción de sus botones.
- "Mi última solución"... se abre al instante el fichero con el código asociado a la máxima nota obtenida en los diferentes intentos realizados por el alumno (guardado en la base de datos de CAP al "Comprobar"). Este botón puede resultar muy útil cuando el alumno usa CAP para completar la resolución de un ejercicio en el que, por el motivo que fuese, no obtuvo un 10.
- "Solución oficial"... se abre al instante el fichero de texto que contiene el código solución propuesto para el ejercicio (guardado en la base de datos de CAP cuando el profesor crea el ejercicio y visible solo a partir de la fecha que este elija). Combinando este botón con el anterior, el alumno siempre puede detectar en qué medida su mejor solución se desvía de la de referencia en corrección, eficiencia y/o estilo.
  Cuando un estudiante accede a dicha solución sin tener un 10, CAP no impide que siga evaluándose pero bloquea su nota actual. Por si el alumno pulsa involuntariamente este botón antes de conseguir un 10, CAP le pide confirmación en una ventana emergente, con lo que el alumno solo ve la solución de referencia cuando así lo decide.
- "Estadísticas"... se abre al instante un fichero con sus estadísticas en las tareas realizadas hasta la fecha. En la figura 3 se puede ver la (completa) información que recibe el alumno sobre su trabajo, la misma de la que dispone el profesor en su ventana: nombre del ejercicio; mejor nota obtenida junto con el número de intentos que realizó para ello; fecha y hora en la que obtuvo dicha nota. Además, si el alumno pincha la línea correspondiente a un cierto ejercicio, puede compararse con el resto de los que han hecho ese ejercicio: dónde se encuentra él con respecto a los que lo han resuelto peor o mejor.
  Este listado de notas le viene bien tanto al alumno como al profesor: evita que ambos pierdan un tiempo precioso en tediosas e imprescindibles tareas administrativas relacionadas con la evaluación (correo viene, correo va; publicación de lis-

tados de notas de actividades de seguimiento sumativas y sus correspondientes revisiones; etc.). Además, les permite conocer en tiempo real el seguimiento del estado del proceso de enseñanza-aprendizaje en el que participan y la detección de los errores o fallos que en él se pueden producir.

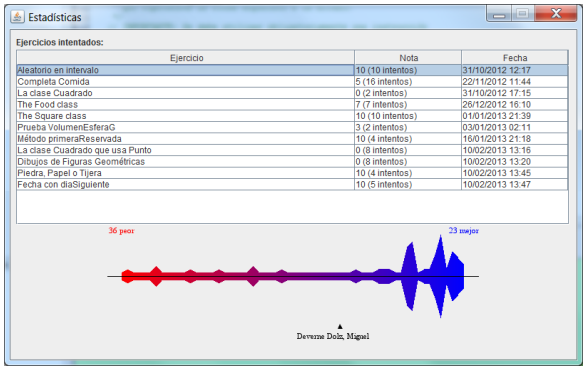

Figura 3: Ventana de CAP para el alumno: estadísticas.

Concluida la descripción de sus botones, pasamos a detallar ahora la información que contienen los mensajes de error de CAP en función de la estructura del ejercicio a resolver. En lo que a evaluación de ejercicios de implementación se refiere, CAP lleva a cabo la corrección de un ejercicio ejecutando un test de prueba del código del alumno; obviamente, solo muestra al alumno un mensaje que resume los resultados del test en lenguaje natural. Como se puede observar en la figura 2, cuando el código contiene errores dicho mensaje ofrece al alumno una guía eficaz para que los solucione y aprenda de ellos, tipo profesor-virtual: además de la localización del código que origina el error, incluye pistas para solucionar tanto errores de ejecución, sobre los que la máquina Java informa de forma escueta y bastante críptica, como errores lógicos, no detectables por la máquina Java. Aunque en la figura no se aprecian, CAP también proporciona otros dos tipos de mensaje de error:

- los asociados a errores de compilación, que son exactamente los mismos que proporciona la máquina Java en cualquier otro entorno y que se califican con un cero, ya que un código que no compila no puede ejecutarse;
- los asociados a un código correcto pero ineficiente, con los que se indica al alumno que si no obtiene una versión eficiente su nota no será la máxima asociada al ejercicio.

Por otra parte, en lo que a evaluación de ejercicios de completar huecos se refiere, CAP compara el contenido de cada hueco con un patrón dado que establece el profesor al crear el ejercicio, el que figura en la solución oficial. Así, tal como se aprecia en la figura 1, el mensaje de error que resume el resultado de este proceso es muy sucinto, tipo oráculo: número de errores en el código y nota; además, a elección del profesor, para indicar su localización, los huecos que contienen el código incorrecto pueden aparecer sombreados en rojo.

Descritas sus características, la experiencia adquirida estos años nos lleva a pensar que, interactuando con esta ventana el alumno puede desarrollar una estrategia inductiva, basada en ejemplos y contra-ejemplos, para aprender a programar programando… ¡y razonando! Comparándola con la de los correctores que se han mencionado en la sección anterior, se puede decir que es de las más efectivas: califica cualquier tarea sin dar pistas sobre su solución más que cuando se comete un error, lo que seguramente es un acicate para el alumno y una ventaja con respecto a aquellos correctores que, según el cuadro 1, no lo hacen; proporciona al alumno una realimentación personalizada, asequible para su nivel y localizada en los errores de su código, conjunción de factores que nos parece que incumplen todos los correctores citados excepto WebToTeach (o CodeLab); finalmente, proporciona toda la información en un mismo plano, haciendo uso de ventanas emergentes en lugar de pestañas u otro tipo de mecanismos.

2.3. Rol del profesor: diseño de tareas y análisis de resultados

El cometido principal del profesor es elegir aquellos ejercicios que plasmen de forma significativa los objetivos concretos de un tema o una práctica. Según la finalidad del ejercicio, el profesor define también su estructura: completar huecos o de implementación.

En cualquier caso, el diseño de un ejercicio requiere rellenar una serie de items: tipo de ejercicio, título, tema y enunciado del mismo, entre otros. Además, si se trata de un ejercicio de completar huecos, se definen los huecos con sus soluciones de referencia y puntuación y si se trata de un ejercicio de implementación (véase figura 4) se generan una serie de clases (visibles o no para el alumno) necesarias para su resolución; entre ellas destaca el test que deberá pasar el código del alumno cuando se ubique en la línea y clase indicadas por el profesor. Como ya se ha comentado, además, este puede añadir una solución de referencia al problema y hacerla visible a partir de una fecha determinada.

La parte más laboriosa e importante es la creación de un tipo de *test-unit* que, explotando la gestión de excepciones Java, permite señalar y, en su caso, ayudar a solventar las dudas y carencias que el alumno pueda tener, tanto si es consciente de ello como si no. El profesor debe ponerse en el papel de alumno y, en base a su experiencia, tratar de prever los errores que el

alumno puede cometer, cuantificarlos para que la respuesta que reciba el alumno pueda variar en función de que se trate de una actividad sumativa en la que se desea un bajo nivel de tutela o una actividad formativa en la que se quieren incluir comentarios y sugerencias que le sirvan al alumno como estímulo para mejorar por si mismo su solución.

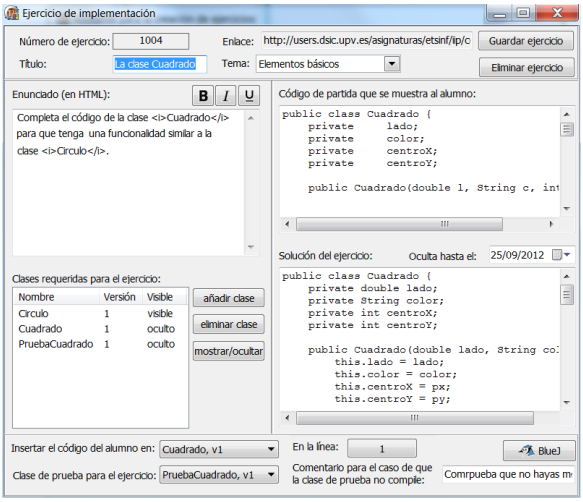

Figura 4: Elementos para el diseño de un ejercicio en CAP.

En la figura 5 se observan los elementos que permiten al profesor el seguimiento y evaluación de la resolución de un ejercicio por parte de los alumnos. Concretamente, puede consultar la siguiente información de un alumno o de un grupo:

- Número de intentos realizados para resolver el problema planteado.
- Fecha en la que ha consultado la solución oficial del problema, en caso de que lo haya hecho.
- Calificación, fecha y código de la mejor solución obtenida por el alumno.
- Histórico de las notas y de los errores cometidos por el alumno en cada uno de los intentos.

Subrayar finalmente que el profesor puede obtener estadísticas sobre los ejercicios planteados mediante la herramienta: número de alumnos que han participado, calificación media, lista de errores más frecuentes, número medio de intentos para encontrar la solución, etc. Constituyen así una valiosa fuente de realimentación para el profesor que, en base a ellas, puede seleccionar con mejor criterio los ejercicios que propone y mejorar los casos de prueba que usará para su corrección.

3. Resultados y discusión

Durante los dos últimos cursos, se le han suministrado al corrector del orden de 56 ejercicios (13 para evaluación de prácticas, algunos con sus versiones en valenciano e inglés, y el resto para actividades de seguimiento) de tres asignaturas de primer y segundo curso del GII de la ETSInf: Introducción a la Informática y la Programación (IIP), Programación (PRG) y Estructuras de Datos y Algoritmos (EDA) (más de 800 alumnos involucrados al año).

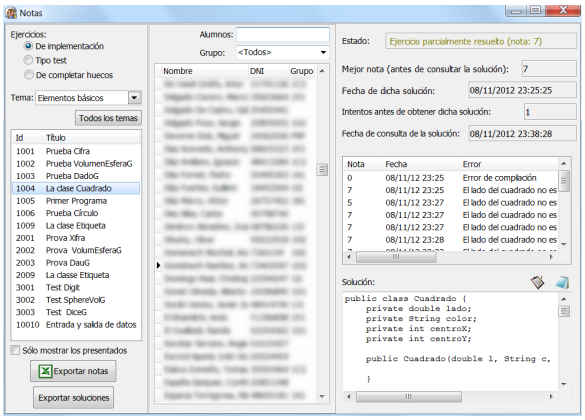

Figura 5: Elementos para evaluar un ejercicio en CAP.

En la figura 6 se representan las calificaciones obtenidas en los parciales y el número de ejercicios realizados por 147 alumnos de IIP; también se representa el número de alumnos que han resuelto un determinado número de problemas.

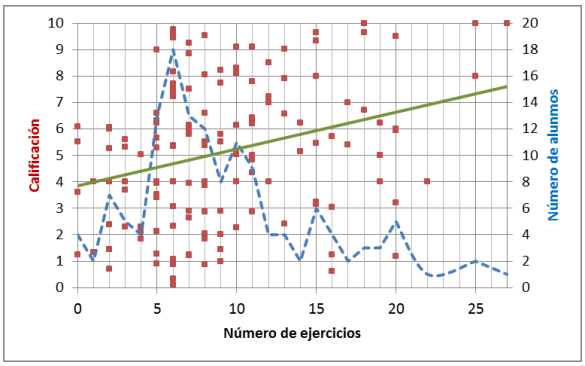

Figura 6: Relación número de ejercicios–calificaciones (puntos y ajuste) y número de ejercicios–número de alumnos (línea discontinua).

Obsérvese en la figura que la línea discontinua tiene su máximo en 6 que corresponde precisamente al número de ejercicios puntuables, el resto, hasta 27, tenían carácter formativo. En cuanto a las notas, se aprecia una relación lineal directa de carácter intermedio entre el número de ejercicios resueltos con CAP y la nota obtenida, lo que apoya la idea de que CAP puede ser una herramienta de aprendizaje efectiva. No obstante, se observan algunos datos que pueden parecer "curiosos" a un lector poco familiarizado con el uso de

correctores automáticos. En el cuadrante superior izquierdo aparecen los datos de unos pocos estudiantes que, habiendo resuelto pocos problemas, obtienen notas altas; en el cuadrante inferior derecho figuran los de otros pocos que, habiendo resuelto muchos problemas, obtienen calificaciones bajas. Mientras que los primeros son estudiantes que pueden aprender a programar con o sin CAP, los segundos son típicamente aquellos a los que les resulta difícil programar y que usan CAP siguiendo una estrategia equivocada de prueba y error.

Un ejemplo significativo que pone de manifiesto la utilidad de CAP, así como la información que proporciona, es un ejercicio sobre el algoritmo de Dijkstra que se utilizó para evaluar a una parte de los alumnos de EDA: 89 alumnos durante hora y media realizaron un total de 1243 intentos (13,97 como media por alumno), obteniendo una nota media de 5,78. Sin tener en cuenta los 624 intentos que dieron error de compilación, el error más frecuente fue que añadían al resultado más vértices de los esperados. A la vista de estos datos, no es difícil deducir el esfuerzo que hubiera supuesto realizar esta misma evaluación sin utilizar CAP, con métodos más tradicionales en los que los mismos profesores atienden y evalúan al mismo número de alumnos en el mismo tiempo. Piénsese también cuál hubiera sido el grado de objetividad que el profesor hubiera podido mantener a lo largo de este proceso.

Por otro lado, se ha diseñado una encuesta para conocer la opinión de los alumnos sobre el uso del corrector. En el cuadro 2 se muestra un resumen de las respuestas de la misma de 195 alumnos.

Enunciado	De acuerdo
Creo que el uso del corrector me puede ayudar a superar la asignatura con éxito.	76 %
Valora el interés de disponer de una biblioteca de ejercicios a resolver para cada uno de los temas.	85 %
Al usar CAP recibo información inmediata sobre algunos errores y eso me ayuda a aprender	78 %
El uso del corrector me ayuda a detectar y solventar algunas dudas y errores que en un ejercicio escrito igual no me surgirían.	77 %
Creo que evaluar un ejercicio utilizando el corrector es más exigente que la evaluación del mismo en papel.	69 %
Al margen de las limitaciones que presenta, valora la eficacia del corrector a la hora de aprender y preparar la asignatura.	66 %
Al margen de las limitaciones que presenta, valora la eficacia del corrector como herramienta de evaluación.	51 %

Cuadro 2: Resultados de la encuesta sobre CAP.

A la vista de las respuestas obtenidas, cabe señalar que más de un 80 % de los alumnos consideran muy interesante poder disponer de una biblioteca de ejercicios a resolver para cada uno de los temas y que lo utilizarían aunque no influyera en su calificación. A pesar de que opinan que evaluar un ejercicio utilizando el corrector es más exigente que la evaluación del mismo en papel, casi un 70 % creen que el uso del corrector les ayuda a detectar y solventar algunas dudas y errores que en un ejercicio escrito no les surgirían y, en definitiva, a superar la asignatura con éxito. Así, en general se puede concluir que el alumno tiene una opinión positiva del uso del corrector aunque lo valoran mejor como herramienta de autoaprendizaje (66 %) que de evaluación (51 %); en esta percepción puede haber influido el resultado obtenido en la evaluación.

4. Conclusiones

En base a los resultados presentados y a la comparación con otros correctores, podemos afirmar que CAP:

- Ayuda al profesor a afrontar con objetividad y eficacia el seguimiento y la evaluación del alumnado, por numeroso que este pueda ser; así mismo, la información que el profesor recibe a partir de las soluciones propuestas por los alumnos es de utilidad para mejorar los ejercicios que se proponen y los casos de prueba asociados.
- Guía al alumno en la resolución de ejercicios de programación, proporcionándole una realimentación instantánea y amigable que le ayuda a resolver sus dudas, corregir sus errores y desarrollar sus habilidades básicas como programador.
- Es una herramienta que combina de manera satisfactoria los criterios de seguridad, facilidad de uso e interactividad; además, soporta con gran flexibilidad la definición de distintos tipos de ejercicios según el objetivo que se pretenda alcanzar.

Los autores tienen la sensación de haber conseguido implementar, y no solo escribir, lo que en [8] se define como un objetivo didáctico: "... expresa con claridad lo que esperamos que el alumno haya aprendido al acabar el curso. Informa sobre el resultado o el cambio esperado en el alumno como consecuencia del proceso de enseñanza-aprendizaje (conoce lo que no conocía, entiende lo que no entendía, hace lo que no sabía hacer...)". En otras palabras, con CAP se puede conseguir evaluar el proceso de enseñanza-aprendizaje o "comparar lo deseado con lo realizado" [1].

Finalmente, en lo que respecta a las líneas de actuación a medio y largo plazo, pensamos incluir mejoras en el análisis de eficiencia y de estilo que en la actualidad realiza CAP. Otro aspecto importante a mejorar es la seguridad de CAP a la hora de evitar copias durante las evaluaciones: monitorización del puesto de trabajo en actividades sumativas y uso de algún sistema de detección de plagio como JPlag[6]. También sería muy

[6] `http://www.ipd.uni-karlsruhe.de/jplag/`

útil integrar la herramienta en la plataforma de formación de la UPV, para que los resultados queden reflejados automáticamente en el libro de calificaciones del alumno que esta proporciona y para poder utilizar su Foro o su Chat para el planteamiento de dudas sobre el funcionamiento de CAP o alguno de sus ejercicios.

Agradecimientos

Los autores quisieran mostrar su agradecimiento a los compañeros que les ayudaron a realizar los experimentos y a los estudiantes que participaron en ellos; agradecer también a la Escuela Técnica Superior de Ingeniería Informática su apoyo y soporte a este tipo de experiencias.

Referencias

[1] M.E. Alfaro. Aspectos prácticos del proceso de programación y evaluación. *Documentación Social*, 81:65–80, 1990.

[2] David Arnow and Oleg Barshay. WebToTeach: An Interactive Focused Programming Exercise System. In *Proceedings of the 29th Annual Frontiers in Education Conference*, volume 3 of *FIE '99*, pages 12A9/39 – 12A9/44, Los Alamitos, CA, USA, 1999. IEEE Computer Society.

[3] Jaime Busquets Mataix, David Roldán Martínez, Susana Martínez Naharro, and Diego Del Blanco Orobitg. PoliformaT: una estrategia para la formación on-line en la Educación Superior. *Virtual Educa*, 2006.

[4] Don Colton, Leslie Fife, and Andrew Thompson. A Web-based Automatic Program Grader. In *Proceedings of the 23rd Information Systems Education Conference*, volume 23 of *ISECON '06*, 2006.

[5] Stephen H. Edwards and Manuel A. Pérez-Quiñones. Web-CAT: Automatically Grading Programming Assignments. In *Proceedings of the 13th annual conference on Innovation and Technology in Computer Science Education*, ITiCSE '08, pages 328–328, New York, NY, USA, 2008. ACM.

[6] Petri Ihantola, Tuukka Ahoniemi, Ville Karavirta, and Otto Seppälä. Review of Recent Systems for Automatic Assessment of Programming Assignments. In *Proceedings of the 10th Koli Calling International Conference on Computing Education Research*, Koli Calling '10, pages 86–93, New York, NY, USA, 2010. ACM.

[7] Christian Murphy, Eunhee Kim, Gail Kaiser, and Adam Cannon. Backstop: A Tool for Debugging Runtime Errors. In *Proceedings of the 39th SIGCSE Technical Symposium on Computer Science Education*, SIGCSE '08, pages 173–177, New York, NY, USA, 2008. ACM.

[8] Juan J. Navarro, Miguel Valero-García, Fermín Sanchez, and Jordi Tubella. Formulación de los objetivos de una asignatura en tres niveles jerárquicos. In *VI Jornadas de Enseñanza Universitaria de la Informática (JENUI 2000)*, pages 457–462, 2000.

[9] Nati Prieto, Marisa Llorens, Germán Moltó, Jon Ander Gómez, Mabel Galiano, and Carlos Herrero. Uso de Metodologías Activas en la Implantación de IIP en el Grado de Informática en la UPV. In *XVII Jornadas de Enseñanza Universitaria de la Informática (JENUI 2011)*, pages 53–60, 2011.

[10] Ricardo Queirós and José Paulo Leal. Programming Exercises Evaluation Systems - An Interoperability Survey. In *Proceedings of the 4th International Conference on Computer Supported Education*, CSEDU '12, pages 83–90. SciTePress, 2012.

[11] Rohaida Romli, Shahida Sulaiman, and Kamal Zuhairi Zamli. Automatic Programming Assessment and Test Data generation. A review on its approaches. In *2010 International Symposium in Information Technology (ITSim)*, volume 3, pages 1186–1192, Los Alamitos, CA, USA, 2010. IEEE Computer Society.

[12] Hussein Suleman. Automatic Marking with Sakai. In *Proceedings of the 2008 annual research Conference of the South African Institute of Computer Scientists and Information Technologists on IT research in developing countries: riding the wave of technology*, SAICSIT '08, pages 229–236, New York, NY, USA, 2008. ACM.

[13] Nghi Truong, Peter Bancroft, and Paul Roe. Learning to Program Through the Web. In *Proceedings of the 10th annual conference on Innovation and Technology in Computer Science Education*, ITiCSE '05, pages 9–13, New York, NY, USA, 2005. ACM.

[14] Norazah Yusof, Nur Ariffin Mohd Zin, and Noor Shyahira Adnan. Java Programming Assessment Tool for Assignment Module in Moodle E-learning System. *Procedia - Social and Behavioral Sciences*, 56(0):767–773, 2012.