UNIVERSITAT POLITÈCNICA DE VALÈNCIA

IN-RED 2017
Congreso Nacional de Innovación Educativa y de Docencia en Red

Congreso In-Red 2017
UPV, 13 y 14 de julio de 2017
Doi: http://dx.doi.org/10.4995/INRED2017.2017.6731

¿Cómo comprobar si 500 alumnos son competentes en el uso de un entorno de programación?

Marisa Llorens, Francisco Marqués y Natividad Prieto

Dep. Sistemes Informàtics i Computació. Universitat Politècnica de València

{*mllorens,pmarques,nprieto*}*@dsic.upv.es*

Abstract

In view of the challenge to evaluate the CT-13 Specific Tools in a subject with more than 500 students, like Introduction to Computing and Programming in the first course of Bachelor's Degree in Computer Engineering in the UPV, we have developed a computer application which, from the assessment tasks designed by the teacher, proposes it to the student, pick the evidences up and provides a qualification. The teacher can check the gathered information, and, where appropriate, she can change this qualification. In this communication, we describe the developed system, we cross-check the efficiency of the proposal and debate the strong and weak points, pointing out possible improvements.

Keywords*: transversal competences, specific tools, rubric, automatic assessment, computer programming.*

Resumen

Ante el reto de evaluar la CT-13 Instrumental específica en una una asignatura de más de 500 alumnos, como es Introducción a la Informática y la Programación de primer curso del Grado en Ingeniería Informática de la UPV, se ha desarrollado una aplicación informática que, a partir de la tarea de evaluación diseñada por el profesor, la propone al alumno, recoge las evidencias y provee una calificación. El profesor puede revisar la información recogida y, en su caso, modificar dicha calificación. En esta comunicación se describe el sistema desarrollado, se contrasta la efectividad de la propuesta y se discuten los puntos fuertes y débiles, señalando posibles mejoras.

Keywords*: competencias transversales, instrumental específica, rúbrica, evaluación automática, programación de ordenadores.*

1 Introducción

Al menos en lo legislativo, ya parece aceptado, también en los niveles educativos superiores, la necesidad de explicitar la adquisición de competencias transversales, esto es, habilidades relacionadas con el desarrollo personal, que no son (solo) específicas de un área de conocimiento y son requeridas para el desarrollo profesional y académico. En el curso 2013/14, la Universitat Politècnica de València (UPV) lanza el Proyecto de Competencias Transversales (ICE-UPV 2017) en el que se definen trece competencias transversales, se establece una estrategia de evaluación sistemática de las mismas y se dan las pautas sobre dónde se adquieren, cómo deben ser evaluadas y las formas de acreditar dicha adquisición. Una de las formas previstas de incorporar estas competencias transversales es a través de los propios planes de estudios. Las ERTs son, entonces, las encargadas de asignar el desarrollo de las competencias a las distintas materias, definiendo lo que se denominan *puntos de control*, esto es, materias que, además de desarrollar la competencia, deben calificarla. El proyecto institucional concreta tres niveles de dominio para cada una de las 13 competencias definidas y establece las actividades formativas y las formas de evaluación más adecuadas para cada una de ellas.

Una de las competencias transversales definidas es la denominada *CT-13: Instrumental específica*, que hace referencia al uso de las herramientas y tecnologías necesarias para el ejercicio profesional asociado a cada titulación. En (ICE-UPV 2017) se describe como sigue:

> *"El estudiante será capaz de identificar las herramientas más adecuadas en cada caso, conociendo sus utilidades y siendo capaz de integrarlas y combinarlas para poder resolver un problema, realizar un proyecto o un experimento."*

En una asignatura como *Introducción a la Informática y a la Programación* (IIP), en el semestre A de primer curso del Grado en Ingeniería Informática (GII), punto de control de esta competencia, se concreta en el siguiente resultado de aprendizaje:

> *Capacitar al alumno en el uso de las herramientas básicas del entorno de programación a partir de las instrucciones que se le proporcionan.*

Nótese que es un resultado de aprendizaje del nivel más bajo (nivel 1), como típicamente corresponde a asignaturas de los primeros cursos. En lo que sigue se detallarán los objetivos de la asignatura IIP y se describirán las actividades formativas para el desarrollo de la competencia y las formas previstas para autoevaluación y evaluación (evaluación formativa y sumativa).

El objeto de esta comunicación es describir una propuesta de evaluación sumativa de la CT-13 en IIP, realizada mediante un sistema (semi)automático de evaluación; así como contrastar la efectividad de la misma.

1.1 Objetivos generales de la asignatura IIP del GII

El objetivo general de una asignatura de introducción a la programación, como IIP, es facultar a los alumnos en la escritura de programas correctos de baja complejidad en un lenguaje orientado a objetos como es Java (Gosling y col. 2014); todo ello como parte de su formación inicial en la construcción de sistemas informáticos. Se trata de que el alumno aprenda a resolver problemas utilizando el computador; para ello, debe aprender a definir clases Java y aplicaciones que, utilizando objetos de las clases definidas o predefinidas en el lenguaje, resuelvan el problema planteado. En concreto, debe aprender nociones básicas de la programación orientada a objetos, como la definición de atributos y métodos de una clase y el uso de la misma mediante la creación y manipulación de objetos, que incluye aspectos propios de la programación imperativa como son el manejo de variables de tipos elementales o agregados como los arrays, el operador de asignación y las estructuras de control condicionales e iterativas. Los problemas abordados son de tipo numérico o de manejo de secuencias de datos y su complejidad se restringe a aquellos que puedan ser resueltos aplicando las estrategias algorítmicas de búsqueda secuencial, recorrido o combinación de estos.

En la enseñanza de la programación de hace unas décadas había una brecha importante entre lo que se podría llamar el diseño de la solución algorítmica del problema y el proceso de programación o traducción a un cierto lenguaje de programación, lo cual se manifestaba incluso utilizando lenguajes distintos para la descripción del algoritmo y para su traducción al programa ejecutable en el computador. La evolución de los lenguajes de programación para facilitar la descripición de los algoritmos y la accesibilidad de computadores con entornos de programación que facilitan todos los procesos relacionados con la escritura de programas para ser ejecutados en ellos, hacen que capacitar al alumno en el uso y manejo de estos entornos sea indiscutible en la actualidad (Robins, Rountree y Rountree 2003; Pettit y col. 2015). Es precisamente este aspecto el que se ha vinculado con la competencia transversal CT-13.

Esta competencia se desarrolla esencialmente a través de las actividades que se proponen en las prácticas de laboratorio y también en la resolución de los problemas propuestos en las clases de teoría y seminario. En concreto, las sesiones de prácticas se organizan en torno al desarrollo de programas para la resolución de problemas de complejidad creciente. Con la realización de estas actividades el alumno debe ser capaz de, además de diseñar la solución algorítmica del problema que se le plantee:

1. Acceder a un sistema compartido en una red de computadores como la del Departamento de Sistemas Informáticos y Computación (DSIC) utilizando el sistema operativo GNU/Linux[1].
2. Mantener organizado su directorio de trabajo siguiendo las pautas que se le proporcionan, para lo cual debe saber manejar de forma básica el sistema operativo, tanto a nivel de consola como de interfaz gráfica.
3. Usar un entorno de programación, como BlueJ[2], para editar código, compilarlo, depurarlo y ejecutarlo.
4. Interpretar los mensajes del compilador para corrección de errores sintácticos.

[1] http://www.gnu.org/
[2] http://www.bluej.org/

5. Utilizar las indicaciones del corrector de estilo.
6. Utilizar el depurador para detectar y corregir errores lógicos y de ejecución.
7. Escribir conjuntos de prueba para asegurar la corrección del código producido.
8. Utilizar las JUnits[3] o programas de prueba que proporcionan los profesores para constatar la corrección del código diseñado.
9. Documentar el código siguiendo las pautas establecidas.

La evaluación de la capacidad del alumno en el diseño de las soluciones algorítmicas a los problemas planteados se realiza de forma completa mediante exámenes escritos en pruebas comunes. La evaluación del resto de capacidades, las que se han asociado a la CT-13, no se pueden (deben) realizar de esta misma manera.

1.2 Evaluación de la competencia del alumno en la utilización del entorno de trabajo

Para conseguir la evaluación de la capacidad del alumno para manejar el entorno de trabajo y, por ello, para realizar las tareas anteriormente mencionadas, es necesario observar su destreza en la resolución de un ejercicio de programación de dificultad acorde a los objetivos a evaluar, en un escenario idéntico al utilizado en la realización de las prácticas y que refleja lo que sería un posible entorno de trabajo profesional, aunque adaptado al nivel de formación en el que se encuentra el alumno. Esto es, el alumno puede consultar la documentación de las prácticas, del lenguaje, del entorno, y el código que el propio alumno ha ido desarrollando en las sesiones de prácticas.

Para que la evaluación sea objetiva y fiable, es necesario que se realice de forma individual, presencial, en el mismo momento del proceso de aprendizaje y con ejercicios de evaluación y criterios de corrección comunes. La idea es asegurar la igualdad de oportunidades a los alumnos, independientemente del instante de evaluación o de los criterios de corrección del profesor. El problema es entonces el elevado número de alumnos matriculados en esta asignatura (más de 500) así como el número de profesores (11) y los recursos de los que dispone el departamento.

Para ello, se ha desarrollado en Java una aplicación propia que plantea al alumno los ejercicios a resolver y le proporciona ciertas clases Java que debe utilizar. El alumno debe ser capaz de ubicar las clases en las carpetas que se le indica y resolver los ejercicios propuestos utilizando el entorno de programación. La aplicación se encarga de almacenar las soluciones de los alumnos en un servidor y de ejecutarlas con los casos de prueba, que definen la rúbrica de evaluación, haciendo una primera calificación de forma automática que después debe ser ratificada por el profesor. De este modo, se pueden organizar 3 sesiones de evaluación consecutivas que, utilizando prácticamente todos los recursos disponibles del departamento (9 laboratorios con 20 computadores cada uno), permiten evaluar de forma individual, presencial y de la forma más equitativa posible a todos los alumnos en un período de tiempo de aproximadamente 3 horas y 30 minutos.

La calificación obtenida de esta manera se utiliza para asignar el nivel de adquisición de la CT-13 siguiendo la rúbrica que se muestra en la Tabla 1; en ella se se detallan los

[3] http://junit.org/

indicadores a considerar, los descriptores o marcas de evaluación y la calificación que se asigna: A si la competencia se supera excelentemente; B si se alcanza completamente; C si se alcanza parcialmente y D si la competencia no se alcanza, es decir, el alumno no logra el nivel mínimo.

Tabla 1: Rúbrica de la CT-13 en IIP

<table>
<tr><th>Indicadores</th><th colspan="4">Descriptores</th></tr>
<tr><td rowspan="2">1. Utiliza el entorno de trabajo (GNU/Linux, Java y BlueJ) para situar el material que se le proporciona y compila y ejecuta un programa</td><td colspan="3">Bien</td><td>Mal</td></tr>
<tr><td colspan="3">Lo hace</td><td>No lo hace</td></tr>
<tr><td rowspan="2">2. Resuelve el problema de programación que se le plantea</td><td>Bien</td><td>Regular alto</td><td>Regular bajo</td><td>Mal</td></tr>
<tr><td>Sin errores</td><td>Error leve</td><td>Error grave</td><td>Errores graves</td></tr>
<tr><td>Calificación de la CT-13</td><td>A</td><td>B</td><td colspan="2">C</td><td>D</td></tr>
</table>

El resto del trabajo se organiza como sigue: en la sección 2 se comentan algunos trabajos que comparten, de algún modo, objetivos con la herramienta que aquí se presenta; en la sección 3 se describe el sistema que permite la evaluación automática de la CT-13 y, a continuación, en la sección 4, se muestran evidencias de la adecuación del sistema a los requerimientos; finalmente, en la sección 5, se discuten las conclusiones del trabajo que permiten señalar algunas direcciones de mejora.

2 Trabajos relacionados

El sistema software que aquí se presenta permite proponer y evaluar ejercicios de programación. En (Douce, Livingstone y Orwell 2005) y (Ihantola y col. 2010) se presenta un resumen de los diferentes sistemas que, con este propósito, se han presentado en la literatura entre los años 1960 y 2005 y en los 5 años siguientes, respectivamente. Las diferentes propuestas varían según el uso pedagógico que se haga de ellas (Ala-Mutka 2005) y de la tecnología que usan. Las diferencias más significaticas entre ellos se encuentran en si realizan un análisis sólo dinámico o también estático del código (ISO 2010), la forma en la que se definen los conjuntos de prueba, cómo se manejan las entregas y cómo se garantiza la seguridad.

Desde el punto de vista del uso pedagógico que se realiza de la herramienta, se pueden agrupar los sistemas en dos grandes grupos:

- Aquellos que únicamente comprueban si el código del alumno cumple con las especificaciones dadas; generalmente sólo realizan análisis dinámico de caja negra, siendo la salida del sistema de tipo lógico: cumple o no cumple. Este es el uso que se requiere en las competiciones de programación.
- Aquellos que pretenden tutelar, en mayor o menor medida, el proceso de aprendizaje; estos suelen realizar análisis estático y dinámico del código del alumno para poder identificar errores que permitan proporcionar al alumno pistas que le ayuden en la resolución del ejercicio. La salida del sistema, obviamente, no es de tipo lógico sino que es una lista de los problemas encontrados e incluso recomendaciones para su resolución.

Todos estos sistemas, en mayor medida los segundos, son unas buenas herramientas de autoaprendizaje y autoevaluación. También, casi todos ellos, permiten llevar un registro de la actividad de cada alumno, con lo que tanto alumnos como profesores pueden tener realimentación del proceso de aprendizaje y actuar cuando y como

sea oportuno. Obviamente, siempre que se den las condiciones apropiadas, también pueden usarse como herramienta de evaluación.

Los autores coinciden con (Ihantola y col. 2010) en que existen demasiados sistemas y que el desarrollo de los mismos se ha realizado de forma desorganizada y acientífica (¿será necesario redescubrir la rueda?) debido principalmente a los problemas de documentación y distribución. En la mayoría de los casos, un grupo de profesores desarrollan una herramienta para un curso concreto de forma completamente altruista y pensando única y exclusivamente en la mejora del sistema de aprendizaje y de evaluación, dejando de lado los detalles relacionados con la publicidad y distribución de la misma. Así se entiende la proliferación de este tipo de herramientas, siguiendo quizás este menosprecio, frecuente en las universidades, en la difusión del material desarrollado para mejora de la actividad docente. De hecho, en IIP y en otras asignaturas del DSIC se utiliza CAP (Sapena y col. 2013), una herramienta de ayuda al aprendizaje de la programación que realiza, entre otras funciones, el análisis dinámico y estático del código y proporciona pautas para la resolución de problemas, o ASys (Insa y Silva 2016), que facilita la corrección de ejercicios de programación en Java.

La pregunta evidente es ¿por qué un sistema más? y la respuesta es que ahora el objetivo es "observar" la habilidad del alumno en el manejo del entorno de trabajo concreto en el que desarrolla las prácticas, y entonces comprobar si, utilizando su usuario habitual, en el sistema operativo GNU/Linux y en un entorno de desarrollo concreto de programación en Java, BlueJ, el alumno es capaz de resolver un problema de programación. Dado que el número de alumnos a observar es muy elevado (siempre más de 500), la herramienta desarrollada resulta de gran utilidad. Ninguna de las herramientas anteriores es directamente aplicable, aunque la experiencia en el uso y diseño de CAP ha ayudado en el diseño de la herramienta presentada en este trabajo.

3 Descripción de la herramienta

En esta sección se describen las características más relevantes de la herramienta siguiendo para ello la clasificación en (Ihantola y col. 2010), su descripción técnica y el uso de la misma por parte de profesores y alumnos.

3.1 Características relevantes

Para la resolución de los problemas de programación que se le proponen, el alumno utiliza el lenguaje Java. Del mismo modo, el conjunto de módulos que integran la propia herramienta están escritos, en su mayor parte, en dicho lenguaje.

La herramienta no está integrada en el sistema de publicación de contenidos de la Universidad (una variación de Sakai[4], denominada PoliformaT (Busquets y col. 2006)) aunque sí que utiliza información relativa a los alummos y de su distribución en grupos, que se encuentra centralizada en los servidores de la Universidad.

La evaluación del trabajo de los alumnos se efectúa mediante la ejecución del código remitido por el alumno (análisis dinámico). No se efectúa análisis estático.

[4] https://sakaiproject.org/

La valoración del trabajo del alumno se efectúa mediante una evaluación automática inicial, que está sujeta a una reevaluación, parcial o completa, posterior realizada por el profesor. El profesor siempre puede modificar la calificación del alumno a la vista de los trabajos remitidos.

La valoración inicial del trabajo de los alumnos se realiza mediante la aplicación de una solución de evaluación propia, especializada (no industrial); aunque la misma se basa en el modelo de Test de Unidad (ISO 2010; Craig y Jaskiel 2008) ya que, principalmente, se evalúan operaciones (métodos Java) que realiza el alumno, sometidos a pre y postcondiciones enunciadas textualmente.

Esta valoración se obtiene mediante:

1. La determinación de que el alumno estructura su trabajo de la forma que se le solicita: generalmente un conjunto de directorios, programas fuente y ejecutables, sometidos a algunas restricciones; por ejemplo, se le puede exigir una estructuración similar a la realizada durante el curso en el laboratorio, o la importación de bibliotecas propias (paquetes realizados durante el curso), etc.
2. La revisión del resultado obtenido por la ejecución del método (o métodos) ante distintos casos de entrada significativos, elegidos de forma que, del resultado obtenido en la ejecución del envío, puedan inferirse las características del mismo.
3. Y, ocasionalmente, del examen del estado de los objetos manipulados (por ejemplo, es posible examinar la evolución del valor de un atributo de un objeto) para determinar la adecuación de la solución.

El alumno puede enviar su solución tantas veces como desee durante el tiempo que dura una prueba que, por lo general, es de una hora. El sistema, además, es configurable para que la evaluación automática se realice bien de la última entrega, bien de aquella en la que obtuvo una mejor calificación. Al alumno se le informa, cuando efectúa una entrega de que la misma ha tenido lugar, aunque solo se le indica que esta se considera correcta cuando el sistema automático le otorga la máxima calificación.

Las pruebas se realizan en ordenadores individuales pertenecientes a la infraestructura docente del DSIC, mediante los que el alumno tiene acceso a su espacio de trabajo habitual en condiciones idénticas a las que tiene durante el resto del curso; a excepción de que tiene limitado el acceso externo, esto es, no puede acceder a internet ni conectarse remotamente ni mediante mensajería, etc., a otros equipos.

El alumno, tras identificarse y obtener acceso legítimo al sistema, descarga al inicio de la prueba algunos elementos de la misma, como enunciados, estructura básica del sistema de corrección, etc. El sistema está preparado para detectar, rechazar e informar de conexiones no autorizadas como, por ejemplo, desde ordenadores fuera del ámbito del laboratorio o desde ambientes no GNU/Linux (que es el s.o. que se utiliza en los laboratorios para la realización de las prácticas).

Además, el sistema identifica y anota el ordenador en el que se trabaja, así como al alumno que realiza la prueba, usando para ello la gestión de cuentas de los laboratorios de prácticas, proporcionada por la infraestructura docente, cuya información se hereda, a su vez, del sistema de cuentas de la propia Universidad. Idénticamente, se anota la fecha y hora en la que se produce cada envío.

La evaluación inicial se realiza en el ordenador del propio alumno, sin que este último tenga acceso al servidor ni al trabajo o calificación de ningún otro compañero.

Todos los envíos del alumno se almacenan, con su calificación inicial, en un servidor realizado en lenguaje PHP[5] sobre una infraestructura Apache[6].

Posteriormente, todos los envíos junto con los datos relevantes de cada uno son clasificados y resumidos, para que cada profesor pueda acceder, a través de una gestión de datos común a los mismos, bien de forma individual (es posible acceder a cualquier entrega de cualquier alumno) como, de forma resumida, a los resultados de un grupo o de todos los alumnos.

Por último, indicar que el sistema se puede configurar para que las pruebas se realicen en cualquiera de las tres lenguas de uso en la UPV (Castellano, Valenciano e Inglés).

3.2 Descripción técnica

La herramienta de evaluación consta de un conjunto de módulos cuyas funcionalidades básicas son:

- Administrativa: permite distribuir, organizar y recoger los enunciados y todas las respuestas de los alumnos de forma eficiente y segura.
- Evaluación de la respuesta del alumno: mediante la que se comprueba la estructuración de cada solución del alumno y su corrección.
- Acceso a los resultados: para lo que se utiliza un repositorio común.

La función administrativa comprende módulos con los siguientes objetivos:

- Distribución del material de la prueba. Se efectúa mediante un grupo de *scripts* en GNU/Linux, ejecutados en el momento del comienzo de cada turno.
- Identificación y validación de los usuarios e implementación de políticas de seguridad. El día del examen solo pueden efectuar una entrega los alumnos debidamente identificados. Estas entregas solo pueden hacerse desde un conjunto de ordenadores reconocidos (que deben utilizar GNU/Linux). La identificación se hace con el sistema de cuentas de la infraestructura docente (que utiliza, a su vez, el de la propia Universidad).
- Remisión a la base de datos de las entregas. Se hace siguiendo el protocolo HTTP[7] (un comando `put`), siguiendo un modelo *cliente-servidor*, en el que el cliente es ejecutado en el espacio del alumno mientras que el servidor, que interactúa con la base de datos, se ejecuta en la infraestructura docente. Estos espacios de ejecución están completamente separados. La remisión incluye, además del código respuesta, los datos de identificación personal y horaria del envío, así como la calificación automática del mismo.

[5] http://php.net/
[6] https://www.apache.org/
[7] https://www.w3.org/Protocols/

En el lado del servidor se anotan en un fichero todas las entregas realizadas por un alumno, junto con las características de cada una de ellas (en Figura 1).

```
ALUMNO: hutarsan  PC: pc0604  FECHA: 09/11/16 13:11  NOTA: T3Ej1 0,60
ALUMNO: hutarsan  PC: pc0604  FECHA: 09/11/16 13:11  NOTA: T3Ej2 0,40
ALUMNO: hutarsan  PC: pc0604  FECHA: 09/11/16 13:14  NOTA: T3Ej1 0,60
ALUMNO: hutarsan  PC: pc0604  FECHA: 09/11/16 13:29  NOTA: T3Ej1 2,00
ALUMNO: hutarsan  PC: pc0604  FECHA: 09/11/16 13:30  NOTA: T3Ej1 2,00
ALUMNO: hutarsan  PC: pc0604  FECHA: 09/11/16 13:31  NOTA: T3Ej1 6,00
ALUMNO: hutarsan  PC: pc0604  FECHA: 09/11/16 13:55  NOTA: T3Ej2 1,20
ALUMNO: hutarsan  PC: pc0604  FECHA: 09/11/16 13:58  NOTA: T3Ej2 1,20
ALUMNO: hutarsan  PC: pc0604  FECHA: 09/11/16 13:59  NOTA: T3Ej2 1,20
ALUMNO: hutarsan  PC: pc0604  FECHA: 09/11/16 14:00  NOTA: T3Ej2 3,20
ALUMNO: hutarsan  PC: pc0604  FECHA: 09/11/16 14:01  NOTA: T3Ej2 3,20
ALUMNO: hutarsan  PC: pc0604  FECHA: 09/11/16 14:02  NOTA: T3Ej1 6,00
ALUMNO: hutarsan  PC: pc0604  FECHA: 09/11/16 14:02  NOTA: T3Ej2 3,20
ALUMNO: hutarsan  PC: pc0604  FECHA: 09/11/16 14:02  NOTA: T3Ej2 4,00
```

Fig. 1: Resumen de las entregas realizadas por un alumno

- Clasificación posterior de resultados y obtención de información resumen. Después de una copia de seguridad de las entregas efectuadas que se realiza al final de cada turno de examen, se distribuye la información (varios miles de ficheros) en distintos directorios de forma que sea fácilmente accesible por los profesores. Simultáneamente a lo anterior, se elaboran listados resumen con los resultados obtenidos (uno por cada grupo de laboratorio y uno para toda la asignatura).

El proceso de evaluación, que ya ha sido descrito, se concreta en una clase Java, que deriva de una clase más general con las características de la estructura de una prueba, cuyo objetivo es la evaluación inicial del envío efectuado por el alumno. Normalmente supone la ejecución de un conjunto de tests, de los cuales es posible inferir la calidad de la solución propuesta.

Finalmente, el acceso a los resultados por parte de los profesores, se organiza mediante un repositorio común en el que cada profesor puede acceder a todas las entregas (y calificaciones) de cada alumno, así como a los listados resúmenes obtenidos tras el procesamiento correspondiente. El soporte de este repositorio (`svn`[8]) lo dota la infraestructura docente, y es el utilizado por los profesores para compartir a lo largo del curso todo el material común.

3.3 Uso de la herramienta: profesor y alumno

3.3.1 Rol del profesor

El profesor debe elegir los ejercicios que reflejen los objetivos de las prácticas a evaluar, preparar el enunciado asociado y el material necesario para que los alumnos realicen el examen usando la herramienta.

El enunciado debe ser lo más claro posible y sin ambigüedades. En (Douce, Livingstone y Orwell 2005) se insiste en que la especificación de los requisitos para una evaluación automatizada debe ser más precisa que para la evaluación manual equivalente. En (Ala-Mutka 2005) se advierte que no se permiten ambigüedades en la especificación del problema, especialmente cuando se consideran los formatos de entrada/salida. Como se indica en (Pieterse 2013), si la especificación es ambigua, son posibles diferentes

[8] https://subversion.apache.org/

interpretaciones y, por tanto, es probable que algunas soluciones válidas de los alumnos puedan ser rechazadas por la herramienta de evaluación automática simplemente porque no esté configurada para reconocerlas.

El diseño de un ejercicio requiere, en primer lugar, la definición de su rúbrica (ver Tabla 1), en particular, establecer una escala de valoración para el indicador 2:

> *Resuelve el problema de programación que se le plantea.*

Se definen 4 descriptores: *bien*, *regular alto*, *regular bajo* y *mal* que se corresponden, respectivamente, con una solución sin errores, con un error leve, con un error grave y con más de un error. El profesor debe ponerse en el papel del alumno y tratar de prever los errores que el alumno puede cometer y cuantificarlos, esto es, debe identificar soluciones correctas y aproximadas, definiendo las distintas instancias y casos de prueba y la nota asociada. La elección de los casos de prueba es un aspecto muy importante de la evaluación automática (Pieterse 2013); por un lado, se debe evitar que programas erróneos pasen los tests y, por otro lado, se debe asegurar que se cubren todas las posibles soluciones.

Una vez definida la rúbrica, el profesor la implementa en un test que comprueba la solución del alumno, identificando el descriptor correspondiente al indicador 2.

3.3.2 Rol del alumno

Al alumno se le proporciona un enunciado con las condiciones para la realización de la prueba y los pasos a seguir para efectuarla. Las condiciones habituales de realización de la prueba son las siguientes:

- 60 minutos de duración.
- Uso exclusivo de los recursos del laboratorio en GNU/Linux, en la cuenta personal del alumno. No se pueden usar, por tanto, portátiles, tablets o cualquier otro tipo de instrumento electrónico.
- Sin acceso a internet.

En la Figura 2 se muestra el esquema de la realización de una prueba de laboratorio por parte de un alumno.

El alumno tiene que copiar en un directorio determinado un fichero `.jar` que se le proporciona y abrir dicho fichero como un proyecto BlueJ. El proyecto contiene los ficheros `.class` y `.java` necesarios para la realización de la prueba. Es condición necesaria tener dicho proyecto BlueJ y ser capaz de ubicar adecuadamente los ficheros que se proponen para poder realizar la prueba. Para comprobar que el alumno ha creado el proyecto con los ficheros indicados, tiene que compilar el código y ejecutar los métodos `main` correspondientes y pulsar *Enviar solución*; lo que corresponde con el indicador 1 de la rúbrica (ver Tabla 1):

> *Utiliza el entorno de trabajo (GNU/Linux, Java y BlueJ) para situar el material que se le proporciona y compila y ejecuta un programa.*

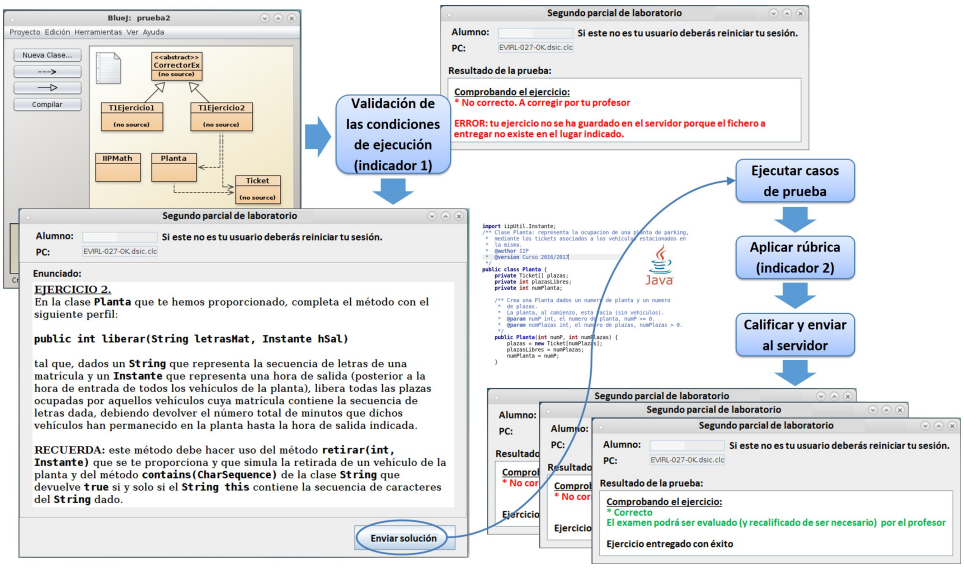


Fig. 2: Esquema de realización de una prueba de laboratorio por un alumno

Al alumno se le muestra el mensaje "*Ejercicio entregado con éxito*" o un mensaje de error si no ha llevado a cabo el indicador 1 correctamente.

Una vez validadas las condiciones de ejecución, el alumno ya puede resolver los ejercicios que se le proponen. Típicamente, un ejercicio consiste en completar un método de una clase Java dada. Una vez resuelto, tiene que volver a pulsar el botón *Enviar solución*. Se ejecutan los diferentes casos de prueba definidos para el ejercicio y se aplica la rúbrica correspondiente al indicador 2 (ver Tabla 1):

> *Resuelve el problema de programación que se le plantea.*

Se le asigna una calificación de acuerdo a la rúbrica y la solución se envía al servidor. El alumno únicamente recibe un mensaje que indica si la solución enviada es correcta o incorrecta. Si no lo es, puede corregir su código y volverlo a enviar todas las veces que quiera, asignándosele la mejor nota obtenida. Esta primera calificación automática, como ya se ha comentado, debe ser ratificada después por el profesor.

4 Resultados

En esta sección se aportan evidencias que permiten asegurar que la herramienta presentada permite evaluar de manera efectiva esta instancia de la CT-13.

El escenario de uso de la herramienta en el curso 2016/17 fue, de forma resumida y aproximada, el que sigue:

- Personal involucrado: 510 alumnos, 11 profesores, 2 becarios y 2 técnicos de laboratorio.

- Recursos técnicos: 3 turnos cada uno de ellos en 9 laboratorios, 20 puestos de trabajo por laboratorio, 1 servidor Apache e infraestructura proporcionada por el departamento.
- Volumen de entregas: más de 4000 ficheros Java y más de 500 ficheros resumen, esto es, 8 entregas en promedio por cada alumno.

La nota que proporciona el sistema, una vez revisada, se utiliza para:

1. Calificar la CT-13, siguiendo los criterios ya comentados en Tabla 1.
2. Contribuir como Nota de Prácticas de Laboratorio (NPL) en un 20 % a la calificación final del alumno en Actas. El resto de la calificación se corresponde en un 60 % a la Nota Ponderada de Parciales (NPP) y en un 20 % a la Nota de Actividades de Seguimiento (NAS) en clase.

En la Figura 3 se muestra el histograma de las diferentes notas de los alumnos por intervalos de calificación. La calificación de la CT-13 se ha hecho según estos intervalos en NPL. Se observa que los alumnos obtienen peores calificaciones en los exámenes de laboratorio. Por el contrario, las notas de actividades de clase son las más elevadas por el carácter motivacional que suelen tener.

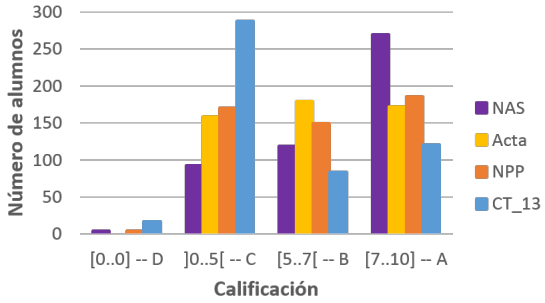
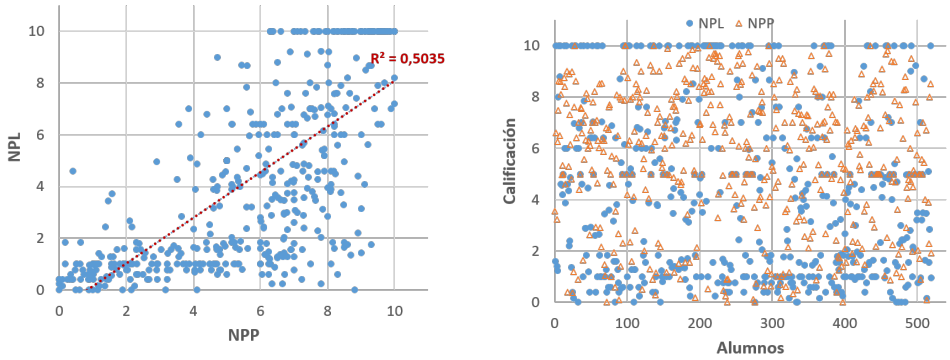


Fig. 3: Relación entre número de alumnos e intervalos de calificación en Acta, NPP, NAS y CT-13

Del 56, 47 % de alumnos que en la CT-13 tienen la calificación C (NPL $\in$]0.,5[), más del 41 % (23, 53 % del total) tienen $0 < \text{NPL} \leq 1$, lo que corresponde con el indicador 1 de la rúbrica; el 3, 33 % tiene la calificación D (NPL $= 0$). Por lo tanto, casi un 27 % del total de alumnos no ha sido capaz de puntuar en la resolución del problema (indicador 2). Casi el 33 % tienen la calificación C con $1 < \text{NPL} < 5$ por haber cometido algún error grave. Más del 16 % tienen la calificación B (NPL $\in$ [5.,7[) por haber cometido algún error leve y casi el 24 % tienen la calificación A (NPL $\in$ [7.,10]) por haber resuelto el problema planteado, al menos de forma aproximada.

En los gráficos de dispersión de la Figura 4, se muestra la relación entre la Nota Ponderada de Parciales (NPP) y la Nota de Prácticas de Laboratorio (NPL). En el gráfico de la Figura 4(a) se confirma que existe una relación lineal positiva entre las dos variables medidas, si bien la correlación existente entre ambas es moderada baja. El coeficiente de determinación, R^2, indica que solo un 50 % de la NPP queda explicado por la NPL. Se puede observar que la mayor parte de los alumnos que no tiene

aprobados los parciales de la asignatura tienen una nota inferior a 2 en NPL. También que los alumnos que tienen la máxima calificación en NPL obtienen notas superiores a 7 en NPP, en la mayor parte de los casos. Del grupo de alumnos con NPP ≥ 5 se observa que hay tanto alumnos que superaron las pruebas de laboratorio como que no lo hicieron. En la Figura 4(b), se observa que las calificaciones de NPL presentan una estratificación debida a la discretización de las notas por el sistema automático. En definitiva, se puede concluir que las pruebas de laboratorio, sorprendentemente (aunque es lo habitual), dan lugar a resultados peores que en el resto de pruebas.

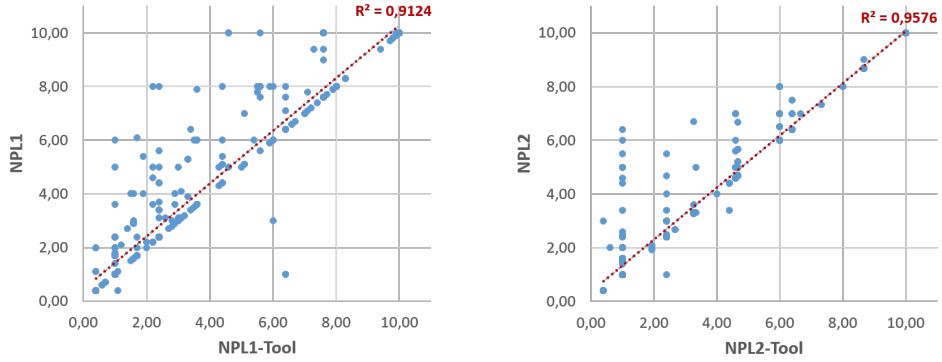


(a) Dispersión y correlación entre NPP y NPL

(b) Distribución de NPL y NPP

Fig. 4: Relación entre NPL y NPP

En la Figura 5 se muestra la relación entre la nota de prácticas obtenida con la herramienta (NPL1-Tool y NPL2-Tool) y dicha nota tras la revisión del profesor para cada uno de los parciales realizados (NPL1 y NPL2). Se observa, en primer lugar, una muy alta correlación entre la nota automática y la nota final, es decir, la calificación obtenida por la herramienta es ratificada por los profesores. Las diferencias se deben principalmente a redondeos de las calificaciones y a que, en un porcentaje reducido de casos, el profesor ha considerado modificar la calificación. En segundo lugar, se observa, una vez más, una discretización de las notas.



(a) NPL1-Tool y NPL1

(b) NPL2-Tool y NPL2

Fig. 5: Dispersión y correlación entre la Nota de Prácticas de Laboratorio con la herramienta (NPL-Tool) y tras la revisión del profesor (NPL)

5 Conclusiones

En este artículo se ha descrito una propuesta de evaluación sumativa de la CT-13 Instrumental específica, realizada mediante un sistema automático de evaluación para IIP en el GII, que se puede extender a otras asignaturas relacionadas con la programación en Java (como, de hecho se utiliza, en la asignatura de segundo curso Estructuras de Datos y Algoritmos).

Se demuestra la efectividad de la propuesta al permitir evaluar la CT-13 a más de 500 alumnos de una manera objetiva, equitativa y con criterios de calificación homogéneos. Descartando cualquier tipo de corrección no realizada en el laboratorio y buscando una evaluación en un contexto similar al que se desarrolla la propia actividad de la programación, la aquí presentada reduce significativamente el tiempo de evaluación. A pesar del automatismo del sistema, el profesor siempre puede revisar la calificación final del alumno.

La preparación y realización de una prueba es laboriosa y exige una planificación cuidadosa de los laboratorios y los recursos. Sin embargo, siempre supone menos trabajo que el que implicaría la evaluación individual de cada alumno. Además, no requiere que todos los profesores se involucren en la preparación de cada prueba.

En lo que respecta a las líneas de actuación a medio y largo plazo, se proponen la siguientes mejoras en la herramienta: limitar el número de envíos para evitar la programación "compulsiva"; introducir análisis estático (evaluando el estilo de codificación); sistematizar la definición del enunciado y desarrollo de los tests; y por último mejorar la realimentación de la nota alumno justificando, de esta manera, la calificación que obtiene.

Finalmente, los autores consideran que este tipo de actividades docentes adicionales, aunque imprescindibles para una evaluación de calidad, deberían tener un reconocimiento explícito en la carga docente del profesor.

Agradecimientos

Los autores quieren mostrar su agradecimiento a los compañeros que colaboraron, en algún momento, en el desarrollo del sistema; en particular, a los profesores Óscar Sapena, Isabel Galiano y Carlos Herrero. También al DSIC, cuyos técnicos proporcionaron siempre la mejor de las ayudas, y, por supuesto, a los estudiantes que realizaron los exámenes.

Referencias bibliográficas

Ala-Mutka, Kirsti M. (2005). "A Survey of Automated Assessment Approaches for Programming Assignments". En: *Computer Science Education* 15.2, págs. 83-102. DOI: 10.1080/08993400500150747.

Busquets, Jaime y col. (2006). "PoliformaT: una estrategia para la formación on-line en la Educación Superior". En: *Virtual Educa*.

Craig, Rick David y Stefan P. Jaskiel (2008). *Systematic Software Testing*. Artech House computing library. Boston: Artech House. ISBN: 1580535089, 9781580535083.

Douce, Christopher, David Livingstone y James Orwell (2005). "Automatic Test-based Assessment of Programming: A Review". En: *Journal of Educational Resources in Computing* 5.3. ISSN: 1531-4278. DOI: 10.1145/1163405.1163409.

Gosling, James y col. (2014). *The Java Language Specification, Java SE 8 Edition*. 1st ed. Addison-Wesley Professional. ISBN: 013390069X, 9780133900699.

ICE-UPV (2017). *Portal de competencias transversales de la Universitat Politècnica de València*. http://www.upv.es/contenidos/COMPTRAN/. [Online; acceso 07-Marzo-2017].

Ihantola, Petri y col. (2010). "Review of Recent Systems for Automatic Assessment of Programming Assignments". En: *Proceedings of the 10th Koli Calling International Conference on Computing Education Research*. Koli Calling '10. Koli, Finland: ACM, págs. 86-93. DOI: 10.1145/1930464.1930480.

Insa, David y Josep Silva (2016). "Corrección Semi-Automática de Programas Java". En: *In-Red 2016 - II Congreso Nacional de Innovación Educativa y Docencia en Red de la Universitat Politècnica de València*. Ed. por Vicente Botti Navarro y Miguel Ángel Fernández Prada. Colección Congresos UPV. Editorial Universitat Politècnica de València. DOI: 10.4995/INRED2016.2016.4326.

ISO, International Organization for Standardization (2010). *ISO/IEC/IEEE 24765: 2010 - Systems and Software Engineering – Vocabulary*, págs. 1-418. DOI: 10.1109/IEEESTD.2010.5733835.

Pettit, Raymond Scott y col. (2015). "Are Automated Assessment Tools Helpful in Programming Courses?" En: *2015 ASEE Annual Conference & Exposition*. Seattle, Washington: ASEE Conferences. DOI: 10.18260/p.23569.

Pieterse, Vreda (2013). "Automated Assessment of Programming Assignments". En: *Proceedings of the 3rd Computer Science Education Research Conference on Computer Science Education Research*. CSERC '13. Arnhem, Netherlands: Open Universiteit, Heerlen, 4:45-4:56.

Robins, Anthony, Janet Rountree y Nathan Rountree (2003). "Learning and Teaching Programming: A Review and Discussion". En: *Computer Science Education* 13.2, págs. 137-172. DOI: 10.1076/csed.13.2.137.14200.

Sapena, Óscar y col. (2013). "Aprender, enseñar y evaluar con CAP, un Corrector Automático de tareas de Programación". En: *Actas de las XIX Jornadas sobre la Enseñanza Universitaria de la Informática: Jenui 2013: Castellón, del 10 al 12 de julio de 2013*. Ed. por Mercedes Marqués Andrés, José Manuel Badía Contelles, Sergio Barrachina Mir y col. Publicacions de la Universitat Jaume I, págs. 343-350. DOI: 10.6035/e-TIiT.2013.13.