Actas de las XXIII JENUI. Cáceres, 5 al 7 de julio de 2017
ISBN: 978-84-697-4077-4
Páginas: 61-68

Experiencia con una herramienta de pruebas de caja negra para el aprendizaje de asignaturas de programación en evaluación continua

Luis Arévalo Rosado [1], Francisco J. Rodríguez[1], Rafael M. Luque-Baena[2], Francisco Luna[2]
Dpto. Sistemas Informáticos y Telemáticos. Centro Universitario de Mérida (UEX)[1];
Dpto. Lenguajes y Ciencias de la Computación. Universidad de Málaga[2]
ljarevalo@unex.es[1], fjrodriguez@unex.es[1], rmluque@lcc.uma.es[2], flv@lcc.uma.es[2]

Resumen

La programación es una de las materias requeridas en la gran mayoría de los grados actuales con perfil tecnológico, siendo un aspecto crítico en cualquiera de los Grados en Ingeniería Informática. Aprender a programar requiere de una práctica constante para hacerse tanto con la sintaxis, semántica y ejecución del lenguaje previamente seleccionado por el profesorado, como con la forma de pensar necesaria para resolver problemas con un ordenador. Tener una retroalimentación rápida sobre los errores cometidos en los ejercicios planteados es crucial para mejorar esta habilidad. Sin embargo, la corrección manual de estas prácticas se vuelve una tarea ardua y compleja en sistemas de evaluación continua. El objetivo de este trabajo es mostrar la experiencia llevada a cabo mediante una herramienta de caja negra existente, VPL (Virtual Programming Lab), para: 1) proporcionar al alumno una retroalimentación inmediata mediante la verificación de los resultados del ejercicio con una batería de pruebas automáticas y 2) proporcionar consecuentemente al profesor una evaluación automática de dichas tareas. La herramienta esta disponible en Moodle, siendo esta plataforma ampliamente utilizada en nuestra Universidad, lo que facilita su uso al estudiante. Se ha implantado en distintas titulaciones, cursos, lenguajes de programación y a distintos niveles de prueba de caja negra, obteniendo unos resultados bastante satisfactorios.

Abstract

Programming has become a core subject in most of the current engineering degrees, being a critical topic for any of the Computer Science specializations. Learning programming requires from the students to steadily doing practical exercises so as to both assimilate the syntax and grammar of the chosen language, and thinking the proper way to address problems with a computer. A swift feedback on the errors made on these exercises is crucial to improve this skill of the students. However, the manual grading by the teacher is a hard, complex, time-consuming task in the ongoing evaluation systems currently deployed. The aim of this work is to show the experience of using an existing black-box-based automatic grading system, VPL (Virtual Programming Lab), to: 1) provide the students with the quick feedback resulting from executing their programs over a set of test cases, and 2) providing the teacher with the automatic grading of these programming exercises. VPL integrates perfectly in Moodle, the software on which the online campus is built on, thus easing its usage by the students, who are already familiar with it. The experience has been developed in different degrees, years, programming languages, and at different levels of black-box testing, achieving a satisfactory results in all of them.

Palabras clave

Pruebas de caja negra, Evaluación automática, Retroalimentación automática, Moodle, Programación

1. Introducción

El profesorado involucrado en la enseñanza de asignaturas de programación es consciente de que los estudiantes deben adquirir habilidades que, generalmente, implican la realización de multitud de ejercicios prácticos. El objetivo de esta metodología es hacerse tanto con la sintaxis y semántica del lenguaje concreto con el que implementar los programas, como con la forma estructurada, lógica y abstracta de pensar para poder resolver problemas correctamente mediante algoritmos implementados en un ordenador [6]. Está muy bien documentado en la literatura que proporcionar retroalimentación a los estudiantes es un factor fundamental en el proceso de aprendizaje [10]. No obstante, la situación ideal en la que se planifica un gran número

de prácticas de programación y su evaluación posterior supone una ingente cantidad de tiempo y esfuerzo para el profesor, e impiden una realimentación adecuada durante su desarrollo. El profesorado entonces ha de decidir entre disminuir el número de prácticas o asumir la sobrecarga de trabajo que le supone entregas frecuentes [3].

Además, la enseñanza en la Universidad de Extremadura (UEx) sigue un modelo cada vez más basado en el llamado "blended-learning" que apoya la enseñanza presencial con un campus virtual donde no solo se recoge la documentación de cada asignatura sino que es donde se proponen y evalúan diferentes actividades para los estudiantes. Tal cual está configurado actualmente, en este campus no es posible ofrecer actividades específicas de programación, sino que simplemente se realizan entregas del código fuente o proyectos para su posterior evaluación por parte del profesorado, lo que resulta bastante ineficiente ya que existen en la actualidad herramientas para tal fin [5].

Otro factor que está muy presente tradicionalmente en este tipo de prácticas es la copia o plagio [1, 7]. Resulta poco gratificante para el profesor, después de descargar el programa, compilarlo, ejecutarlo para comprobar que funciona, y analizar si el estilo de programación es el correcto, reconocer que esa práctica "le suena". A la hora de programar, los estudiantes creen que cambiar el nombre a determinados elementos del programa, o añadir comentarios adicionales, es suficiente para que no se detecte el plagio.

Estos factores, catalizados por una propuesta de proyecto de innovación educativa para varias asignaturas y titulaciones dentro de la UEx, nos llevaron a buscar estrategias para la corrección automática de dichos ejercicios prácticos. El objetivo que se busca es mejorar el proceso de enseñanza y aprendizaje en las asignaturas de programación, ofreciendo al alumno un mayor número de ejercicios y prácticas y proporcionándole una evaluación instantánea de los mismos. Para ello, se ha utilizado la herramienta VPL (Virtual Programming Lab) [9], desarrollada en la Universidad de Las Palmas de Gran Canaria y se ha elegido, entre otras razones, por su integración en Moodle, por el soporte a varios lenguajes y por sus mecanismos de detección de plagio. Para evaluar la experiencia, se han realizado encuestas de satisfacción a los estudiantes y, por otro lado, se ha registrado información sobre la calificación automática y el número de intentos por ejercicio.

El trabajo se estructura como sigue. En la Sección 2 se incluye una revisión de la literatura y las funcionalidades deseadas en nuestro sistema. La Sección 3 detalla el desarrollo de la experiencia, introduciendo VPL, la metodología seguida y algunos ejemplos prácticos de casos de uso. La Sección 4 analiza y discute los resultados obtenidos, mientras que la Sección 5 presenta las principales conclusiones del trabajo e incluye posibles líneas de actuación futura.

2. Requisitos del sistema de evaluación y breve estado del arte

El objetivo de esta sección es doble, por una parte, describir los requisitos que debe cumplir el sistema de evaluación automático para nuestra experiencia docente y, por otra parte, y a partir de varias referencias bibliográficas contrastar dichos requisitos con las características de distintos sistemas.

Como se ha mencionado previamente este trabajo surge a partir de un proyecto de innovación docente para la evaluación automática de asignaturas de programación. El sistema a usar debe poseer las siguientes características:

- Validación de tareas de programación en distintos lenguajes mediante la ejecución de una serie de casos de prueba en los que, dado un valor de entrada, la evaluación consiste en comprobar si el valor de salida es el esperado.
- Funcionalidad basada en entorno Moodle para garantizar una adaptación rápida de nuestros estudiantes al sistema, es la plataforma usada en la UEX. Además se tendría adicionalmente control de estudiantes, tareas, foros, etc.
- Soporte para múltiples lenguajes y así su aplicación a distintas asignaturas.
- Soporte para la detección de plagios en las entregas realizadas.

Bajo el paraguas de las palabras clave *automatic grading systems* o *automatic programming assessment* (entre otros), se pueden encontrar multitud de trabajos relacionados con la corrección automática de programas, solo a partir de 2015. Es un área de investigación en claro auge, si bien las primeras herramientas de apoyo a la docencia de asignaturas de programación datan ya de la década de los 60 [4].

En [2] se muestra un resumen bastante pormenorizado de distintas herramientas para la evaluación de entregas de programación, detallando sus principales características tales como plataforma de ejecución, tipo de licencia, lenguajes soportados, modo de trabajo (web o escritorio) y las métricas para la evaluación. Este trabajo es un buen punto de partida para descubrir las diferentes herramientas existentes (JavaUnitTest, Marmoset, WebCat, JAssess, VPL, JavaBrat, entre otras).

Un estudio más reciente sobre este tema publicado en 2016, que el lector interesado puede encontrar en [8], ha identificado hasta 69 herramientas diferentes y se informa que la lista no es completa ni exhaustiva. Los autores han clasificado dichas herramientas en base al tipo de retroalimentación que proporcionan, por

lo que permite tener una visión muy precisa sobre cuál de ellas es la que mejor se ajusta a las necesidades de cada contexto particular.

Entre los distintos sistemas existentes, se ha seleccionado Virtual Programming Lab (VPL)[9] por cumplir las características indicadas.

3. Descripción de la Experiencia

En esta sección se detalla la herramienta VPL (Sección 3.1), se describe la experiencia docente con VPL (Sección 3.2) y se muestran varios ejemplos de uso (Sección 3.3).

3.1. Virtual Programming Laboratories

VPL *(http://vpl.dis.ulpgc.es/)* es una herramienta de código abierto creado por la Universidad de Las Palmas de Gran Canaria. Entre sus características podemos destacar:

- Integración dentro de la plataforma Moodle, suavizando de este modo la curva de aprendizaje para los estudiantes (al tratarse de la plataforma educativa usada por la UEX). Además, al integrarse como una tarea Moodle el sistema hace el control completo de la entrega.
- Incorpora un editor simple e intuitivo permitiendo las principales funcionalidades: copiar, pegar, deshacer, rehacer, etc.
- Permite la ejecución, depuración y evaluación de varios ficheros Java así cómo de otros tantos lenguajes de programación: ADA, C/C++, VHDL, SQL, PROLOG, etc. (pueden consultarse todos los lenguajes en la web del proyecto).
- Dispone de mecanismos para verificar la autoría de las prácticas y tratar de evitar así su plagio. Para ello, analiza la similitud entre prácticas atendiendo a diferentes métricas.

A nivel de funcionamiento, la arquitectura de VPL se encuentra compuesta por tres módulos:

- Plugins Moodle: Integrado como una nueva tarea dentro de esta plataforma. En la actividad se pueden configurar aspectos como: la fecha de entrega, el número de ficheros, la puntuación mínima o máxima, las opciones de ejecución o evaluación así como la batería de caja negra. En la actualidad, febrero 2017, VPL se encuentra disponible para la última versión de Moodle.
- Un servidor de ejecución (jail-celda) que se encarga realmente de la ejecución de las pruebas. Por cada evaluación se crea una orden de ejecución en el servidor, independiente del resto de ejecuciones del sistema, lo que garantiza el aislamiento de las distintas ejecuciones.

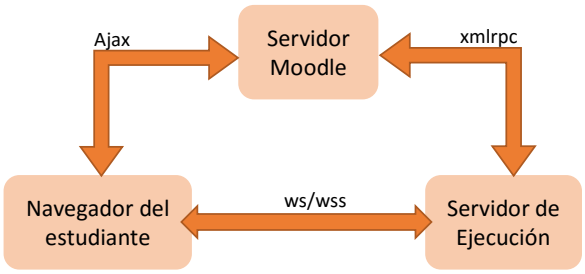

Figura 1: Arquitectura de VPL

- Editor de código fuente. Permite desde el propio navegador editar los ficheros subidos para la entrega. El editor permite las operaciones típicas de edición, sin tener en ningún momento acceso a la máquina de ejecución.

De este modo, el servidor de ejecución y el servidor de entrega se encuentran en distintas máquinas, tal y como puede verse en la Figura 1. El acceso al sistema Moodle se realiza mediante el navegador. Cuando el usuario solicita una ejecución o evaluación, el servidor jail recibe peticiones del módulo y da respuestas a éste mediante el protocolo XML-RPC.

3.2. Metodología

Este trabajo surge a partir de un proyecto de Innovación Docente de la Universidad de Extremadura denominado "Pruebas de caja negra como herramienta para el seguimiento continuo de asignaturas de programación", asociado al área de Lenguajes y Sistemas Informáticos. El objetivo del mismo era mejorar el proceso de enseñanza-aprendizaje en diversas asignaturas de programación, aumentando el número de ejercicios y prácticas con los que el estudiante pueda valorar la evolución de su aprendizaje, proporcionándole retroalimentación inmediata sobre la corrección y calificación de su entrega, y con una mayor objetividad.

Se ha realizado la instalación de VPL en un Moodle distinto al usado en nuestra Universidad para realizar el experimento denominado *Campus Virtual Alternativo (http://vpl-cum.unex.es/moodle)*. Actualmente se esta llevando a cabo en 3 asignaturas impartidas por dicho área en el Centro Universitario de Mérida:

- Estructura de Datos y de la Información (EDI). Impartida en los Grados en Ingeniería Informática en Tecnologías de la Información y Telemática, continúa con la enseñanza de la materia de programación adquirida en Fundamentos de Programación. Introduce los conceptos de programación orientada a objetos y se desarrolla en Java.
- Metodología y Desarrollo de Programas (MDP). Impartida en el Grado en Ingeniería Informática en Tecnologías de la Información, representa la

tercera asignatura de la materia de programación, desarrollándose también en el lenguaje Java.

- Informática (INF). Impartida en el Grado en Ingeniería en Diseño Industrial y Desarrollo de Productos, presenta a los alumnos de esta titulación los conceptos básicos de programación, utilizando como lenguaje vehicular Python.

En todas las asignaturas propuestas la dinámica de la realización de tareas sigue un esquema de dificultad incremental: las primeras tareas son más livianas y sirven para familiarizarse con los lenguajes estudiados y los conceptos básicos introducidos, mientras que las últimas involucran manejar todos los recursos y técnicas expuestas durante el curso. El diseño de los casos de prueba será también más elaborado a medida que la complejidad de la tarea aumenta, pasando de ejercicios básicos que implican subir un solo fichero, a ejercicios que requieren una colección de clases, librerías dependientes y ficheros de entrada y/o salida. Es posible la reutilización de casos de prueba entre asignaturas que compartan los mismos ejercicios en lenguajes de programación diferentes, pero será necesario configurar tanto la compilación del fuente (si el lenguaje de programación lo requiere) como la ejecución en cada lenguaje. En la Sección 3.3 se muestran varios ejemplos que plasman la idea comentada.

Para evaluar la experiencia se ha elaborado una encuesta individual con el propósito de conocer la opinión de los estudiantes con respecto a la nueva estrategia planteada en lo que se refiere a la utilización de la plataforma virtual. Esta encuesta se responde de manera anónima y voluntaria, una vez han finalizado los exámenes en las distintas asignaturas. Se aportará más datos de la misma en la sección 4.1.

Además, para completar el estudio realizado en este trabajo, se han analizado los resultados obtenidos por los alumnos en las distintas asignaturas (Sección 4.2), comparando sus calificaciones con respecto a años anteriores donde no se utilizaba la herramienta.

3.3. Ejemplos de Uso

En este apartado, se muestran un par de ejemplos de uso de esta herramienta. A la hora de añadir una tarea VPL en Moodle es necesario establecer entre otras la siguientes configuraciones: el número de ficheros de código fuente, la nota mínima que el estudiante debe sacar así como el periodo habilitado para la entrega. Una vez realizado este proceso, es necesario configurar el entorno de ejecución, depuración y evaluación de la tarea mediante los ficheros *vpl_run*, *vpl_debug*, *vpl_evaluate* y *vpl_evaluate.cases*. Los tres primeros ficheros son donde se establece las instrucciones para la ejecución de la tarea en el servidor de ejecución, mientras que el último contiene la batería a comprobar.

El primer ejemplo se corresponde con una entrega sencilla, de las primeras sesiones, realizada en las asignaturas de INF y EDI. El enunciado es como sigue: *"Dada una cantidad de dinero en euros (número positivo, si no mostrar el mensaje error) calcule el número mínimo de billetes y monedas necesario, sabiendo que disponemos de billetes de 500, 200, 100, 50, 20, 10 y 5 euros, y monedas de 1 y 2 euros, y 50, 20, 10, 5, 2 y 1 céntimo. Las monedas o billetes que no se necesiten para el cálculo no deben mostrarse. Si se necesita más de una moneda o billete se debe mostrar el plural ('3 monedas' en lugar de '3 moneda')"*.

Para este ejemplo, no ha sido necesario realizar una configuración adicional de los ficheros de ejecución y evaluación (*vpl_run*,*vpl_evaluate*) a como se realizan por defecto, pues al tratarse de un proyecto simple no es necesario configurar ningún parámetro adicional. La batería de test en ambos casos es la misma y puede observarse en la Figura 2.

vpl_evaluate.cases

```
case = Test 1
Grade reduction = 20%
input = 342.78
output = 1 billete de 200 euros
1 billete de 100 euros
2 billetes de 20 euros
1 moneda de 2 euros
1 moneda de 50 céntimos
1 moneda de 20 céntimos
1 moneda de 5 céntimos
1 moneda de 2 céntimos
1 moneda de 1 céntimo

case = Test 2
Grade reduction = 20%
```

Figura 2: Ejemplo básico de VPL

Esta batería se encuentra constituida por 5 casos, de los cuales solo se muestra el primero de ellos en la Figura 2. En él se establece el valor de entrada (*input*) a 342.78 y la salida (*output*) correspondiente ha dicho valor. De este modo, cuando el estudiante selecciona evaluar, VPL comprueba que la salida de la ejecución se corresponde con la salida pre-establecida. En este fichero se ha configurado un 20 % de reducción (*Grade reduction*) por cada uno de los casos que el estudiante no supere correctamente.

El segundo ejemplo que mostraremos en este apartado se trata de un ejemplo más completo donde ha sido necesario configurar diversos parámetros. Éste se ha llevado a cabo en la asignatura de MDP (2º curso de Informática) y está constituido por 9 clases Java. La evaluación se realiza a partir de la ejecución de una batería de test jUnit. El sistema por sí solo permite la ejecución de jUnit pero se ha tenido que adaptar para obtener una calificación con este tipo de baterías.

El desarrollo de la práctica consistía en la in-

formatización de un desguace constituido por las siguientes clases: `Persona`, `Pieza`, `Vehiculo`, `Coche`, `Moto`, `Camion`, `Empleado`, `Proveedor` y `Desguace` así como todo un conjunto de relaciones entre ellas. La batería jUnit se realiza sobre la clase `Desguace` donde se encuentra la signatura de los métodos a testar. Para no extendernos en exceso, solo mostraremos uno de los métodos (15 en total) `public Vehiculo getVehiculoBastidor(int bastidor)`.

```
//script1
@Test
  public void testGetVehiculoBastidor() {
   assertNull(d.getVehiculoBastidor(1234));
   d.addVehiculo(v1); //id = 123
   assertNull(d.getVehiculoBastidor(15));
   assertNotNull(d.getVehiculoBastidor(123));
  }
```

En el `script1` se muestra el código de la batería `testGetVehiculoBastidor()` que se encarga de recuperar un vehículo a partir de su bastidor. En primer lugar, con la colección vacía comprueba que al recuperar un vehículo no existente genera un valor nulo el método y a continuación añade un vehículo que posteriormente recupera.

```
//script2
1. rm -rf es
2. export
     CLASSPATH=/usr/share/java/junit4.jar
3. mkdir es/unex/cum/mdp/sesion4
4. PACKDIR=es/unex/cum/mdp/sesion4
5. PACKFILES="*.java"
6. cp $PACKFILES $PACKDIR
7. javac -Xlint:unchecked -J-Xmx16m
     $PACKDIR/ *.java >> compilation_errors
8. cat common_script.sh > vpl_execution
9. echo "java -enableassertions -cp
     $CLASSPATH:/usr/share/java/junit4.jar
     org.junit.runner.JUnitCore
          es.unex.cum.mdp.sesion4.TestDesguace"
          >> vpl_execution
```

El fichero de ejecución (*vpl_run*) se muestra en el `script2`, donde se han omitido algunas líneas debido a su longitud. La primera línea ha sido necesaria porque al trabajar con paquetes, se ha comprobado que a veces quedaba rastro de la anterior ejecución y por tanto es necesario borrarla por completo. En la línea 7 se procede a la compilación de los ficheros `*.java` que han sido incluidos en los directorio oportunos (líneas 3-6). Finalmente, en la línea 9, se ejecuta la batería `TestDesguace` sobre los ficheros subidos.

El anterior script se ejecuta correctamente en VPL pero no permite obtener una calificación automática de la entrega, únicamente indica por cada batería si su ejecución ha sido correcta o incorrecta. Para resolver esta problemática, se ha implementado un programa java (`script3`) que se encarga de realizar la calificación a partir de la bateria jUnit. Como puede comprobarse, por cada método, si la ejecución es correcta, incrementa en 5 la puntuación de la evaluación mientras que si se produce cualquier excepción, muestra el oportuno mensaje sin realizar calificación positiva alguna.

```
//script 3
public static void main(String[] args) {
  ..
  try {
        TestDesguace t = new TestDesguace();
     t.setUp();
     t.testGetVehiculoBastidor(); //
     t.tearDown();
     System.out.println(formatOutput("Test
          testGetVehiculoBastidor", "5",
          null));
     grade += 5; //incrementa en 5
   } catch (AssertionError e) {
     System.out.println(formatOutput("Test
          testGetVehiculoBastidor", "5", e));
   } catch (Exception e) {
     System.out.println(formatOutput("Test
          testGetVehiculoBastidor", "5", new
          AssertionError(e.getMessage())));
   }
 ...
```

4. Resultados

En esta sección, analizamos los resultados obtenidos en la experiencia realizada. En la Sección 4.1, se analizan los resultados de las opiniones de los estudiantes a través de las encuestas descritas en la Sección 3.2. En la Sección 4.2, estudiamos los resultados de la experiencia docente a través de la comparación de las calificaciones de los alumnos en las tareas de programación asignadas, comparando dichas calificaciones con las obtenidas en las mismas tareas en cursos anteriores donde no se utilizaba dicha herramienta. Y en la última sección, se muestran algunas aportaciones personales de la experiencia realizada.

4.1. Encuestas

La encuesta está compuesta por seis preguntas, las tres primeras de tipo cuantitativo (valoración de 1 a 5) y otras tres preguntas reflexivas en las que el alumno puede expresar mediante texto libre su opinión. Las preguntas planteadas tienen la siguiente formulación:

1. *Es útil el campus virtual alternativo para comprobar si la práctica está correctamente realizada.*
2. *La operabilidad / funcionalidad del campus virtual es (difícil 1 a fácil 5).*
3. *Grado de satisfacción con el campus virtual alternativo.*
4. *Aspectos positivos más destacables del campus virtual alternativo.*

5. *Aspectos negativos más destacables del campus virtual alternativo.*
6. *Sugerencias de mejora y observaciones.*

En esta primera sección, mostraremos los resultados de las encuestas en las asignaturas de INF durante los cursos 15/16 y 16/17, EDI en el curso 15/16 y MDP en el curso 16/17. Destacar que el porcentaje de alumnos que responde a las encuestas es bastante alto, entre el 54 % y el 82 % de los alumnos matriculados y entre el 84 % y el 95 % de los presentados, indicando que los resultados de las mismas son significativos. La valoración de los estudiantes para las tres primeras preguntas de la encuesta puede verse en la Figura 3. Para la asignatura de Informática solo se disponen de las respuestas a las preguntas 1 y 3 para el curso 15/16, pues este primer año la encuesta fue ligeramente diferente.

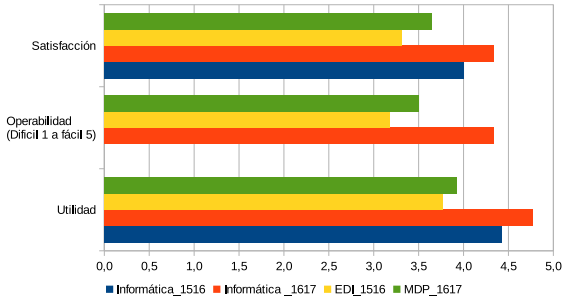

Figura 3: Encuesta sobre el uso de la herramienta

A partir de los resultados de las encuestas mostrados en la Figura 3, podemos extraer las siguientes conclusiones:

- La valoración de la utilidad es muy satisfactoria por parte de los estudiantes, con valores que van desde un 3,8 de media de los alumnos de EDI en el curso 15/16 hasta un 4,8 de los alumnos de INF en el curso 16/17.
- La opinión sobre la operabilidad o facilidad de uso de la herramienta son también bastante positivos, con valores que van desde un 3,2 por parte de los alumnos de EDI hasta un 4,3 de los alumnos de INF. Destacar la importancia de este punto, ya que la facilidad de uso es un punto crítico a la hora de evitar que la operabilidad suponga una barrera de entrada para los alumnos.
- En general, la satisfacción de los alumnos con el sistema automático de evaluación de tareas es muy positiva, desde un 3,3 de los alumnos de EDI hasta un 4,3 de los alumnos de INF.
- Finalmente, es interesante ver la consolidación del uso de la herramienta en la única asignatura (INF) en la que disponemos de datos de dos cursos académicos. Podemos ver como tanto la valoración de la utilidad como la satisfacción general de los alumnos ha crecido en este último curso.

A tenor de estos datos, se pueden observar diferencias relativamente importantes entre las valoraciones de los alumnos de las diferentes titulaciones. En general, la valoración de los alumnos del Grado en Ingeniería en Diseño y Desarrollo de Productos es más alta, en los distintos apartados, que la de los alumnos de los Grados de Informática y Telemática, lo que puede indicar que aquellos alumnos de disciplinas más alejadas de la programación para los que dicha materia puede resultar más complicada, valoran más positivamente la ayuda que dicha herramienta les proporciona para el aprendizaje de la programación.

En cuanto a las respuestas a las preguntas reflexivas planteadas a los alumnos en las encuestas, a continuación destacamos los aspectos más comentados por los alumnos en dichas respuestas:

- Los alumnos destacan la posibilidad de ser evaluado en cualquier momento con retroalimentación inmediata sin esperar a la respuesta del profesorado. Indicar que se habilitaron otros tantos ejercicios no evaluables en asignaturas como EDI y MDP, permitiendo avanzar a los estudiantes en el dominio de la programación a través de una herramienta ya conocida por ellos.
- El principal aspecto negativo recogido es la sensibilidad del sistema en la evaluación en la salida: cualquier carácter distinto al esperado provoca una evaluación errónea así como la poca información generada por el sistema en estas situaciones.

4.2. Análisis

El objetivo de esta sección es analizar el impacto de la herramienta de evaluación automática de tareas en las propias calificaciones de los alumnos. Para ello, disponemos de las calificaciones de las mismas tareas en cursos académicos en los que no se ha utilizado dicha herramienta y cursos académicos en los que sí se ha hecho uso de ella. En concreto disponemos de los siguientes datos:

- En la asignatura de EDI, se tiene calificaciones de las mismas cuatro tareas evaluables para el curso 15/16, sin VPL, y el curso 16/17 utilizando VPL.
- Para la asignatura de INF, disponemos de la calificación de las mismas seis tareas evaluables para el curso 14/15, sin VPL, y el curso 16/17, utilizando ya VPL. La tercera tarea no es evaluable mediante VPL.
- En la asignatura de MDP, tenemos las calificaciones para el curso 15/16, sin utilizar VPL, y el curso 16/17, con VPL. En ambos cursos se ha pedido la entrega de las mismas 4 tareas calificables.

En las Figuras 4, 5 y 6 podemos ver gráficamente la nota media de cada una de las tareas evaluables de las

asignaturas, EDI, MDP e INF, respectivamente, para cada uno de los cursos analizados e indicando si en dicho curso se ha utilizado VPL.

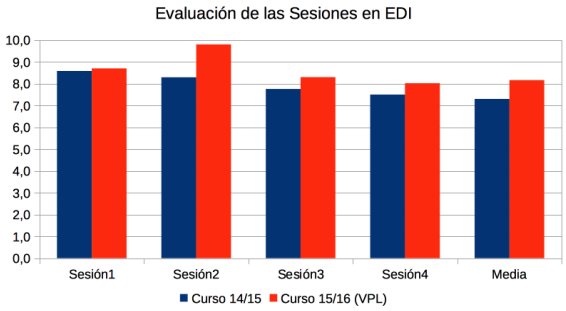

Figura 4: Nota media por curso académico de EDI

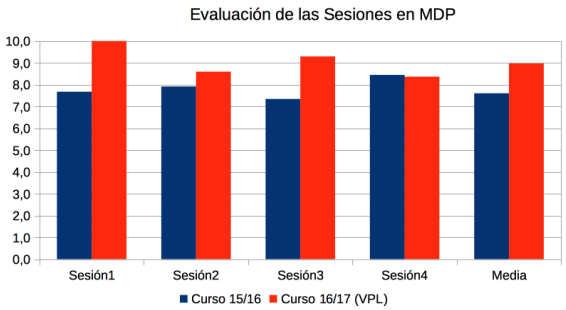

Figura 5: Nota media por curso académico de MDP

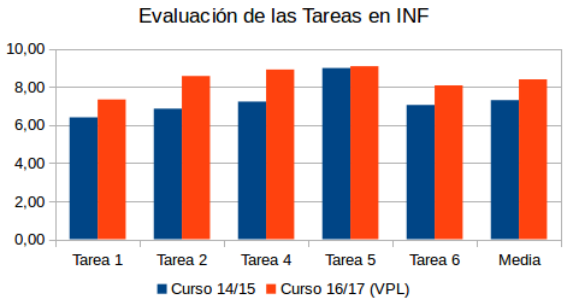

Figura 6: Nota media por curso académico de INF

Como se puede apreciar en dichas figuras, y teniendo en cuenta la limitada cantidad de datos disponible en el estudio, podríamos indicar que las calificaciones de los alumnos en dichas tareas han mejorado en cada una de las tres asignaturas analizadas con el uso de VPL como herramienta de corrección automática. Este hecho puede explicarse fácilmente dado que a través de la herramienta VPL cada tarea es sometida a una batería de pruebas, que es en la mayoría de los casos mucho más amplia que la que cualquier alumno puede aplicar por propia iniciativa, y por lo tanto el alumnado puede detectar un mayor número de errores.

Los beneficios esperados de la utilización de VPL por parte de los alumnos deberían incrementarse según lo explicado anteriormente a medida que aumente la complejidad de las tareas. Para estudiar este efecto, vamos a comparar el número de veces (intentos) que el alumno envía una misma tarea para su evaluación. Es importante destacar que no tienen la misma dificultad las tareas de INF, que está dirigida a estudiantes que se están iniciando en la materia de programación, que las tareas dirigidas a estudiantes de los títulos afines.

En la Figura 7, se muestra el número medio de intentos por alumno y tarea en las tres asignaturas. Este valor se obtiene como la media de los intentos realizados por todos los estudiantes en esa tarea o sesión y asignatura. Como se puede apreciar, efectivamente el número de intentos es más alto en MDP que en EDI y a su vez en está última que en INF. Esto es lógico si tenemos en cuenta que MDP, como se ha visto en la Sección 3.3, su batería de pruebas es mucho más compleja que en el resto.

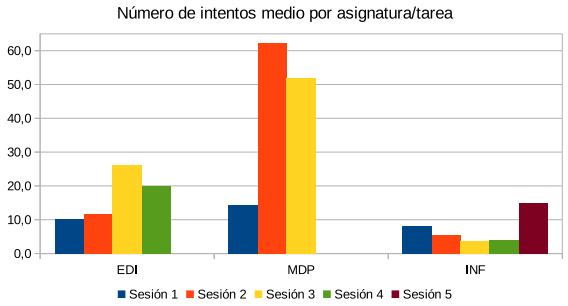

Figura 7: Número de Intentos

Finalmente, destacar que, si bien se ha comprobado un aumento en las calificaciones medias con el uso de VPL en las tareas de evaluación continua, no se ha apreciado ninguna correlación con una mejora en la evaluación final de los estudiantes que han usado este sistema, lo cual corrobora lo expuesto en [11].

4.3. Discusión

Esta herramienta lleva usándose casi dos cursos académicos, con más de 30 tareas evaluables y otras tantas no evaluables en distintas asignaturas, considerando muy positivos sus resultados tanto para el profesorado como para el alumnado.

Desde el punto de vista del profesor nos gustaría resaltar la posibilidad de proporcionar a los estudiantes una evaluación continua realista sin necesidad de una sobrecarga excesiva de trabajo, solo siendo necesario preparar las baterías de pruebas para los ejercicios propuestos. Además, dispone de un mecanismo para detectar plagio en las entregas que en clases numerosas es muy difícil llevarse a cabo. Y finalmente, destacar

que puede usarse como una herramienta única a lo largo de toda una titulación, a aplicar a distintos cursos y lenguajes de programación sin ningún coste adicional de aprendizaje para el alumnado. Con respecto a la funcionalidad de detección de plagio, comentar que en las primeras sesiones se muestra la potencialidad de dicha función a los estudiantes como factor disuasorio y solamente ha tenido efectos en la evaluación en tres estudiantes, cumpliendo su objetivo en nuestra opinión.

Desde el punto de vista de los estudiantes, les proporciona una herramienta para conseguir el deseado propósito de *"A programar se aprende programando"*, pueden hacer muchas tareas y reciben una retroalimentación instantánea. Este aspecto ha sido valorado positivamente por ellos. Además, el estudiante toma consciencia de la necesidad de realizar pruebas exhaustivas en sus programas.

Como puntos negativos de esta metodología, está el hecho de que los alumnos se puedan enfocar exclusivamente en los resultados y descuiden la calidad del código y una adecuada comprensión del mismo. Esta dificultad entronca directamente con la propia naturaleza de la metodología de caja negra. Además, con respecto a la propia herramienta, los estudiantes manifiestan la poca información proporcionada por los resultados de error y la inflexibilidad en las entradas y salidas de los algoritmos, pues la comparación con la salida ideal se realiza carácter a carácter.

5. Conclusiones y Trabajo Futuro

El profesorado involucrado en la enseñanza de programación es consciente de que los estudiantes deben adquirir habilidades que solo pueden conseguir practicando. No obstante, la situación ideal en la que se planifica un gran número de prácticas de programación, conlleva una ingente cantidad de tiempo y esfuerzo para el profesor, impidiendo una retroalimentación en tiempo y forma.

Por ello, en este trabajo se ha mostrado la experiencia realizada con una herramienta de caja negra existente para realizar la evaluación automática y así facilitar la adquisición por parte de los estudiantes de las destrezas necesarias y, al mismo tiempo, aliviar la carga de trabajo del profesorado.

A partir de los datos obtenidos, se observa que existe una gran aceptación por parte de los estudiantes y que mejoran sus calificaciones en las tareas evaluables. Como principal punto a favor esta la posibilidad de comprobar en cualquier instante si el programa funciona correctamente y en consecuencia él adquiera la consciencia de una ejecución correcta y sin errores.

Como trabajo futuro, se propone: 1) mejorar la flexibilidad de las entradas y salidas mediante el uso de expresiones regulares, 2) incluir evaluaciones de caja blanca, 3) aplicación en otras asignaturas y 4) realizar un análisis estadístico más profundo, disponiendo de una muestra más amplia en años sucesivos.

Referencias

[1] A. M. Bejarano, L. E. García, and E. E. Zurek. Detection of source code similitude in academic environments. *Comput Appl Eng Educ*, 23:13–22, 2015.

[2] J.C. Caiza and J.M. Del Alamo. Programming assigments automatic grading: Review of tools and implementations. In *INTED2013 Proceedings*, 7th International Technology, Education and Development Conference, pages 5691–5700. IATED, 4-5 March, 2013 2013.

[3] J. Carter, C. English, K. Ala-Mutka, M. Dick, W. Fone, U. Fuller, and J. Sheard. How shall we assess this? In *Proc. 8th Annual Conf. Innovation and Technology in Computer Science Education (ITiCSE 2003)*, page 17, 2003.

[4] C. Douce, D. Livingstone, and J. Orwell. Automatic test-based assessment of programming: A review. *Journal on Educational Resources in Computing*, 5(3), 2005.

[5] M. Farrow and P. J. B. King. Experiences with online programming examinations. *IEEE Transactions on Education*, 51(2):251–255, 2008.

[6] C.A. Higgins, G. Gray, P. Symeonidis, and A. Tsintsifas. Automated assessment and experiences of teaching programming. *ACM Journal on Educational Resources in Computing*, 5(3):Article No. 5, 2005.

[7] M. Joy and M. Luck. Plagiarism in programming assignments. *IEEE Transactions on Education*, 42(2):129–133, 1999.

[8] H. Keuning, J. Jeuring, and B. Heeren. Towards a systematic review of automated feedback generation for programming exercises. In *Annual Conference on Innovation and Technology in Computer Science Education, ITiCSE*, pages 41–46, 2016.

[9] Juan Carlos Rodríguez del Pino, Enrique Rubio Royo, and Zenón J Hernández Figueroa. Vpl: Laboratorio virtual de programación para moodle. In *XVI Jornadas de Enseñanza Universitaria de la Informática*, pages 429–435, 2010.

[10] V. J. Shute. Focus on formative feedback. *Review of Educational Research*, 78(1):153–189, 2008.

[11] Belle Selene Xia and Elia Liitiäinen. Student performance in computing education: an empirical analysis of online learning in programming education environments. *European Journal of Engineering Education*, pages 1–13, 2016.