Actas de las Jenui, vol. 4. 2019.
Páginas: 151-157

Extensión de Moodle para la evaluación automática de ejercicios de programación en Java

Guillermo Quintana, Héctor Pérez, Mario Aldea, Julio Medina y Rubén Sebrango
Departamento de Ingeniería Informática y Electrónica
Grupo de Ingeniería Software y Tiempo Real
Universidad de Cantabria
gqp904@alumnos.unican.es, hector.perez@unican.es, mario.aldea@unican.es, julio.medina@unican.es , ruben.sebrango@alumnos.unican.es

Resumen

El uso de sistemas de gestión y aprendizaje online como Moodle está en constante auge y, cada vez más, se utiliza como medio principal de comunicación con los alumnos, ya sea para proporcionarles material de estudio o para la entrega de trabajos y prácticas.
Este artículo presenta una herramienta de evaluación automática de ejercicios de programación integrada en la plataforma Moodle. Esta herramienta permite evaluar las entregas de los alumnos desde su subida a la plataforma, y generar datos de realimentación tanto para alumnos como profesores. La utilización de un sistema como éste ayuda a los docentes a reducir en parte el trabajo asociado a la evaluación continuada y, a su vez, es capaz de proporcionar a los alumnos una idea inmediata de las deficiencias de su ejercicio. En su versión actual, la herramienta es capaz de evaluar tanto el comportamiento funcional como la calidad del código escrito en lenguaje Java.
Este sistema de evaluación automática se ha utilizado con éxito en algunas de las prácticas de la asignatura de Estructuras de Datos de un Grado en Ingeniería Informática.

Abstract

The use of online learning and management systems such as Moodle is constantly growing and it is considered one of the preferred ways to communicate teachers and students. This article presents a tool for the automatic evaluation of programming exercises which is integrated in the Moodle platform. This tool is able to evaluate the students' submissions and generate feedback data for both students and teachers. The use of this kind of systems helps teachers to reduce the work associated with continuous assessment and it is also able to provide students with an immediate idea of the deficiencies found in their exercise. In its current version, the tool is capable of evaluating both the functional behavior and the quality of source code written in Java. This automatic assessment system has been successfully used in some lab exercises from the Data Structures course (Degree in Computer Engineering).

Palabras clave

Moodle, aprendizaje autónomo, evaluación automática.

1. Introducción

A lo largo del proceso de enseñanza-aprendizaje de un lenguaje de programación resulta imprescindible que el profesor lleve un control continuo del trabajo de cada alumno, y que el alumno vaya identificando progresivamente las deficiencias de sus soluciones. Estas deficiencias no deben estar solo enfocadas al resultado funcional del problema presentado, sino que debe incluir aspectos de calidad (legibilidad, complejidad, eficiencia, etc). La evaluación de todos estos aspectos constituye un proceso largo y complejo, especialmente en cursos introductorios donde el número de alumnos es generalmente alto en relación al profesorado disponible.

En este contexto, resulta interesante disponer de un sistema capaz de evaluar tanto el resultado funcional como la calidad del código de manera automatizada y coherente. Este tipo de planteamiento fomenta el aprendizaje autónomo del estudiante [5], el cual pasa a obtener información inmediata sobre las deficiencias detectadas en sus ejercicios, y en consecuencia favorece la supresión progresiva de malos hábitos en la programación. En el caso de los profesores, se reduciría en parte el trabajo asociado a las actividades de evaluación continuada.

Por otro lado, los Sistemas de Gestión de Aprendizaje (*SGA*) como Moodle facilitan el seguimiento del proceso de aprendizaje, así como el desarrollo de actividades de evaluación que generalmente permiten vincular calificaciones con comentarios de realimentación. En la actualidad, el uso de SGA está totalmente integrado en el modelo educativo de multitud de universidades, por lo que extender este tipo de plataformas con nuevas herramientas garantizan una adaptación rápida de estudiantes y profesores [1].

Por tanto, el objetivo fundamental de este artículo reside en el desarrollo y configuración de un sistema de evaluación automática de ejercicios de programación informática, así como su integración en Moodle por ser una de las plataformas SGA más utilizadas en la actualidad. En líneas generales, el sistema propuesto debe ser capaz de detectar deficiencias y generar comentarios sobre los ejercicios de programación entregados por los alumnos, así como establecer una calificación en función de unos parámetros preestablecidos por el profesor.

El documento está organizado de la siguiente manera. En el apartado 2 se describe la arquitectura de la herramienta y se proporcionan detalles de su implementación, mientras que la metodología de uso se describe en el apartado 3. El apartado 4 analiza los resultados obtenidos en las pruebas realizadas durante las prácticas de la asignatura de Estructuras de Datos de un Grado en Ingeniería Informática. El apartado 5 está dedicado a presentar el trabajo relacionado y los principales antecedentes de la herramienta desarrollada. Finalmente, el apartado 6 plantea las conclusiones y el trabajo futuro.

2. La extensión EvalCode

2.1. Visión general

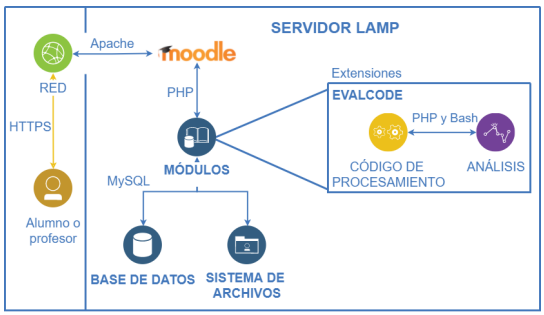

Figura 1: Diagrama de arquitectura

El sistema de evaluación automática se ha desarrollado como una extensión de Moodle denominada *EvalCode.* Esta extensión está integrada en Moodle junto con el resto de recursos y actividades proporcionados por la plataforma (por ejemplo, el chat, el cuestionario, la encuesta, el foro, la tarea, etc.).

En la Figura Figura 1 puede observarse la relación entre la extensión propuesta y las distintas tecnologías sobre las que se apoya el servidor Moodle, que suelen identificarse con las siglas LAMP (Linux, Apache, MySQL y PHP). Según se puede observar en esta Figura, los usuarios de la plataforma (p.ej., alumnos o profesores) acceden a través de la red al servidor Apache en el que se encuentra alojado Moodle. La comunicación entre éste y los módulos disponibles se hace mediante clases y funciones PHP que, a su vez, ejecutan sentencias MySQL para realizar consultas e inserciones en la base de datos, y rutinas de *Bash* para la gestión de los archivos en el sistema. La extensión *EvalCode* se integra como un nuevo módulo en Moodle.

Por último, el proceso de análisis y evaluación automática se sirve de los módulos existentes para la interconexión con el resto de componentes de Moodle, tal y como se detalla a continuación.

2.2. Implementación

El desarrollo de *EvalCode* está basado en el módulo "Tarea" de Moodle, ya que este módulo proporciona una funcionalidad para la configuración y entrega de ejercicios similar a la requerida por nuestro sistema de autoevaluación. Entre otros aspectos, ha sido necesario modificar la lógica encargada de crear tareas y de procesar las entregas para incorporar el sistema de evaluación automática.

Por otro lado, Moodle está diseñado de acuerdo a una estructura modular que facilita el desarrollo de nuevas extensiones. En nuestro caso, la extensión *EvalCode* hace uso de los siguientes módulos proporcionados por la plataforma:

- *Access*: Módulo responsable del acceso a los contenidos en función de los permisos que disponga el usuario. De forma similar a lo que ocurre en otras actividades disponibles en un curso, las operaciones disponibles o la información que se visualiza en una tarea *EvalCode* varían en función del tipo de usuario (alumno, profesor o administrador).
- *Moodle_database*: Módulo que se encarga de la comunicación con la base de datos. La extensión hace uso de la base de datos MySQL integrada en Moodle para almacenar las entregas y los *tests* con los que debe hacer el proceso de calificación. Concretamente, se han definido nuevas tablas de datos para almacenar la información relacionada con la configuración de la actividad por parte del profesor, las entregas de los alumnos o la evaluación.
- *File*: Módulo responsable de la gestión de ficheros requeridos por la extensión. Por ejemplo, el

enunciado de los problemas, las clases de prueba, las entregas o la realimentación.

- *Navigation*: Módulo que gestiona la información contextual del usuario. Este módulo se utiliza principalmente para obtener referencias que nos permitan acceder a información más detallada a través de otros módulos, como por ejemplo el identificador del usuario, el identificador del curso actual o el módulo que se está visualizando.
- *GradeBook*: Este módulo se encarga de gestionar la evaluación de los alumnos. En nuestro caso, además de definir una nueva tabla de datos para almacenar las calificaciones, se ha desarrollado la lógica para permitir la recalificación de los alumnos.
- *Form*: Módulo responsable de los elementos que conformarán la vista de la aplicación. Este módulo posibilita que todas las actividades de la plataforma presenten un diseño gráfico homogéneo.

2.3. El proceso de la evaluación automática

Este apartado revisa los detalles internos de la extensión *EvalCode* para completar el proceso de análisis y evaluación automática.

Una vez que el alumno realiza la entrega de la tarea, ésta se sube a la base de datos MySQL. Posteriormente, la extensión descarga los archivos en un directorio temporal con un nombre único construido a partir del número de usuario (proporcionado por Moodle) y una marca temporal. Para un mayor control de seguridad, en las últimas versiones de Moodle todos los procesos lanzados por el servidor son ejecutados por un usuario con permisos limitados que solo puede escribir en un determinado directorio.

Una vez almacenados los ficheros en el directorio temporal, el sistema comprueba la validez de la entrega. Si el código proporcionado por el alumno no cumple los requisitos para ser evaluada (por ejemplo, poseer una estructura de paquetes adecuada), el sistema notifica al alumno del error que ha cometido.

En caso de que la entrega cumpla los requisitos establecidos, *EvalCode* añadirá una clase Java predefinida que actuará como el método *main* de la aplicación. En esta clase nueva se añaden los nombres de las clases requeridas por las pruebas JUnit.

A continuación, se procede a la compilación del código, tras la cual se pasa a ejecutar el programa y se almacena en un archivo de texto la salida generada por JUnit con toda la información referente a las pruebas que se hayan especificado. Si se produce algún error durante la compilación del programa, se le notifica al alumno.

Una vez finalizadas las pruebas funcionales, el sistema realiza la comprobación de estilo mediante la herramienta Checkstyle, cuya salida también es almacenada en un fichero de texto para su procesado posterior.

Por último, el proceso de calificación involucra dos partes: los datos generados por JUnit, de donde se extrae el número de pruebas exitosas y fallidas, así como los errores de estas; y el número de *warnings* o errores detectados durante la verificación de estilo. De acuerdo a la configuración especificada por el profesor para el cálculo de la nota, se asigna un valor a la entrega y se almacena en la base de datos. Además, se adjuntan ficheros como realimentación para el alumno con los resultados de la ejecución de JUnit y de Checkstyle, así como un resumen del cálculo de su nota dentro del apartado de *feedback*. Tras esto, el proceso de evaluación finaliza y se le muestran todos los resultados al alumno.

3. Metodología de uso

3.1. Descripción general

El primer prototipo funcional de la extensión *EvalCode* permite evaluar ejercicios de programación en lenguaje Java. La evaluación automática del código entregado por los alumnos se realiza en base a dos criterios: (a) corrección funcional, evaluado mediante el uso de clases de prueba de JUnit; y (b) estilo, evaluado con la herramienta Checkstyle.

Dentro de la metodología propuesta para el uso de nuestra extensión de Moodle pueden identificarse las fases mostradas en la Figura Figura 2.

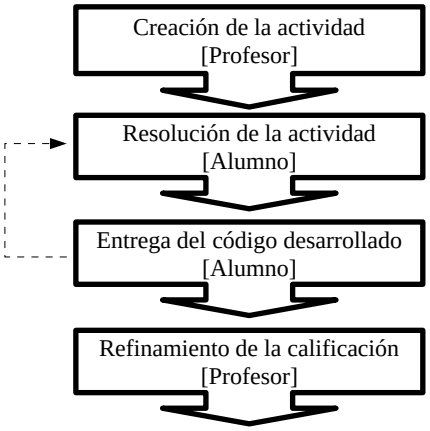

Figura 2: Metodología de uso de *EvalCode*.

En los siguientes apartados se procede a describir en más detalle cada una de las fases.

3.2. Creación de actividad EvalCode

Durante la creación de una actividad de tipo *EvalCode* el profesor debe configurar los aspectos requeridos por cualquier tarea Moodle: fechas de entrega, configuración de envío, grupos, etc.

También debe indicar la ponderación en la calificación final del alumno de cada herramienta de análisis

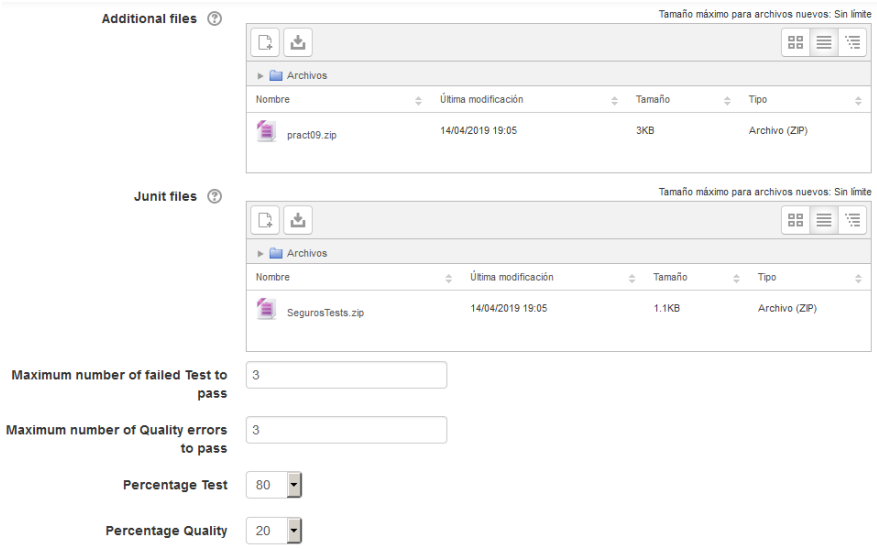

Figura 3: Cuadro de diálogo para la creación de una actividad EvalCode.

(JUnit y CheckStyle en la versión actual de nuestra herramienta). Además, para cada herramienta debe indicar el número de fallos que correspondería a una calificación de 5, de forma que las calificaciones de los alumnos se calcularán proporcionalmente a dicho valor. La Figura Figura 3 muestra un detalle del cuadro de diálogo correspondiente a la creación de una actividad EvalCode.

Por último, el profesor debe incluir dos ficheros comprimidos con paquetes, clases e interfaces Java:

- Fichero comprimido con el código inicial (campo *Additional files* de la Figura Figura 3) : conjunto de paquetes con clases e interfaces Java, parcial o completamente implementadas, que constituyen el material proporcionado al alumno. Deberá incluir las librerías requeridas para la realización de la práctica (en el caso de que las hubiera). Entre ellas se podría incluir una clase probadora de JUnit para facilitar al alumno la prueba de su código antes de realizar la entrega.
- Fichero comprimido con el código final (campo *JUnit files* de la Figura Figura 3): incluye aquellas clases e interfaces del código inicial que el alumno no debería haber modificado. También incluye la clase probadora con la que se realizará la evaluación (que puede ser igual o diferente de la proporcionada en el código inicial).

Las reglas utilizadas por CheckStyle para la evaluación del estilo del código son configurables por el profesor. En el estado actual de nuestra extensión *EvalCode*, dichas reglas se definen a nivel de servidor Moodle, lo que significa que son comunes para todas las tareas y cursos en dicho servidor.

3.3. Realización de la actividad

El alumno realiza la actividad partiendo del código inicial proporcionado por el profesor. Dependiendo del ejercicio propuesto, la solución podría ser arbitrariamente compleja, pudiendo requerir la implementación de nuevos paquetes, clases y/o interfaces. Este desarrollo puede realizarse utilizando cualquier IDE (p.ej. Eclipse).

Si se desea, durante esta fase se puede permitir que el alumno ejecute en su ordenador un conjunto de pruebas para evaluar sus progresos antes de realizar la entrega. En ese caso, dicho conjunto de pruebas habría sido incluido en el código inicial proporcionado por el profesor (en la forma de una clase probadora de JUnit).

3.4. Entrega y realimentación

Una vez realizado el ejercicio, el alumno entrega un fichero comprimido con el código desarrollado. El fichero debe incluir todo el contenido del directorio de fuentes (p.ej. el directorio src/ de un proyecto Eclipse).

Al realizar la entrega, nuestra herramienta genera la realimentación para el alumno y calcula su calificación siguiendo el proceso mostrado en la Figura Figura 4. Como se aprecia en esta Figura, la herramienta descomprime en primer lugar el fichero entregado por el alumno y, a continuación, el código final proporcionado por el profesor. De esta forma las clases que no deberían haber sido modificadas y, en particular, la clase probadora, son sobreescritas con el código proporcionada por el profesor, descartando los posibles cambios realizados por el alumno. A continuación se ejecutan las herramientas de análisis (JUnit y CheckStyle). El resultado de la ejecución de las herra-

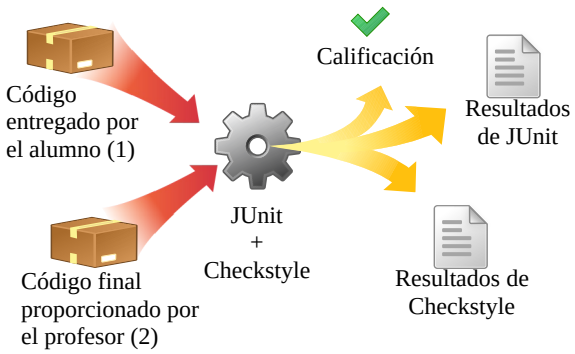

Figura 4: Procesado de una entrega.

mientas de análisis junto con la calificación obtenida constituye la realimentación proporcionada al alumno.

La calificación se registra en la tabla de calificaciones del alumno, como la de cualquier otra actividad calificable de un curso Moodle. El alumno puede iterar entre las fases de realización y entrega tantas veces como desee salvo que el máximo número de entregas haya sido limitado en los parámetros de configuración de la tarea.

3.5. Refinamiento de la calificación

La calificación asignada automáticamente al alumno puede ser posteriormente modificada por el profesor. Así, el profesor puede realizar una revisión del código para evaluar aspectos difícilmente detectables por las herramientas automáticas, tales como la estructura del código o su eficiencia.

Además, nuestra herramienta incluye una opción que permite recalificar las entregas de algunos o todos los alumnos utilizando un nuevo conjunto de pruebas.

4. Pruebas experimentales

A fin de poner *EvalCode* en explotación con garantías de receptividad por parte del alumnado y de la administración, se han realizado diversas pruebas. Las pruebas unitarias iniciales se hicieron en la máquina de desarrollo pero para ponerlo a disposición de los alumnos se hizo necesario contar con un servidor de explotación temporal que sea accesible desde casa y desde los laboratorios docentes, tal como ocurre con el servidor moodle institucional. Así pues gracias a los gestores del moodle institucional pudimos configurar este servidor usando una copia de la máquina virtual que actualmente tiene la universidad en explotación. Así, tanto el paso al servidor de explotación final como las futuras actualizaciones serán más efectivas. Co ello las pruebas de campo realizadas resultan más confiables para todos los agentes involucrados pues las versiones del sistema operativo, del servidor moodle y todos los paquetes necesarios son los mismos en ambas plataformas.

Para abrir los puertos en el firewall de la universidad para este nuevo servidor de explotación temporal se hizo necesario también contar con el apoyo de los vicerrectorados correspondientes. El hecho de que usáramos una copia del entorno de explotación final, al que además tienen acceso remoto los gestores de la red institucional, facilitó el conseguir los permisos preceptivos a nivel interno.

Las primeras pruebas en el servidor de desarrollo, de carácter funcional, sirvieron para depurar fallos en la codificación de los scripts, y fueron realizadas a la par que el código durante el proceso de desarrollo. Pero una vez operativo las pruebas de campo se plantaron con entregas reales. La asignatura empleada para estas pruebas ha sido “Estructuras de Datos” de segundo curso de Ingeniería Informática, que realiza prácticas de programación semanales y para las que los alumnos están ya bastante familiarizados con las herramientas de programación que utilizan. Así, al final del primer cuatrimestre, en el curso 2018-2019 conseguimos tener operativa toda la infraestructura necesaria.

Las pruebas de campo consistieron en pedir a los alumnos que además de subir sus prácticas al servidor moodle que han venido empleando de forma habitual, lo hicieran, al finalizar la práctica en el aula, en el servidor de pruebas que tenemos montado. Los aspectos que las pruebas de campo nos habrían de revelar eran por una parte la receptividad de los alumnos, y por otra la contención real y tiempos de espera para dimensionar los recursos que serían necesarios en condiciones reales de explotación.

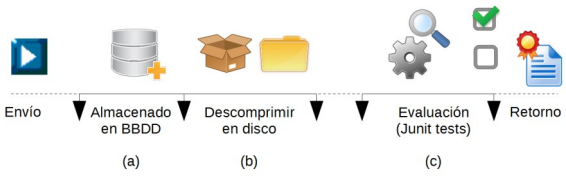

Figura 5: Especificación de tiempos de una entrega

La Figura Figura 5 muestra las partes del proceso de entrega cuyos tiempos que se han medido instrumentando convenientemente el código de la herramienta. El tiempo de almacenado en la base de datos(a), el de extracción al sistema de ficheros y descompresión (b) y finalmente el de Ejecución de la evaluación propiamente dicha (c). Los tiempos de envío por la red en ambos sentidos así como el de compilación de la unidades a probar no han sido monitorizados.

Se efectuaron pruebas con las dos prácticas finales de la asignatura, en la primera se pidió que subieran sus versiones finales y en la segunda se les pidió que utilizaran *EvalCode* como forma de autoevaluación.

Hemos utilizado la versión 3.3.7 de Moodle y ejecuta sobre una máquina virtual con un kernel Linux 4.15. Empleamos así la misma infraestructura que usa el servidor moodle institucional, de modo que una vez validado sea más fácil la migración al servidor de explotación. El procesador que aloja la máquina virtual usado en las pruebas es un Pentium(R) Dual-Core E6300 @ 2.80GHz con 8 Gb de RAM.

Los resultados obtenidos para la primera práctica se muestran en la Figura Figura 6.

Como se puede apreciar el grueso del tiempo empleado se dedica a la evaluación de los tests y se agolpa cuando las entregas se hacen muy juntas. La nota media obtenida por los alumnos que entregaron vía EvalCode esta práctica fue de 5.41.

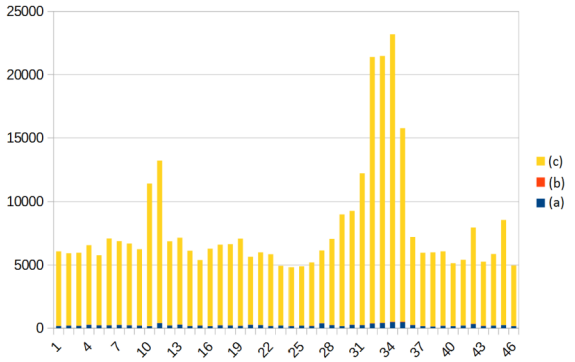

Figura 6: Tiempos observados en las entregas.

En la segunda práctica se recibieron muchas más entregas pues se pidió a los alumnos que las hicieran como forma de autoevaluación hasta conseguir los mejores resultados posibles. En este caso la nota media de los alumnos fue 6.3.

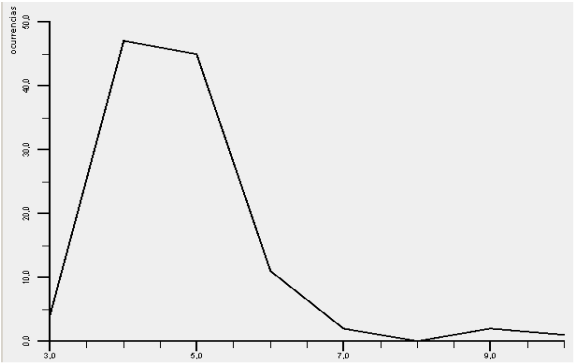

Figura 7: Distribución de tiempos de la segunda práctica.

La Figura Figura 7 muestra la distribución de tiempos de evaluación para las entregas recibidas en la segunda práctica; el eje de abscisas expresa el tiempo en segundos y el de ordenadas las entregas acumuladas en cada tramo de tiempos considerado.

La tabla de la Figura Figura 8 muestra los valores promedio calculados para los tiempos observados en las entregas de las dos practicas recibidas que se muestran gráficamente en las figuras Figura 6 y Figura 7. Estos valores son razonables de cara a su uso en explotación en el servidor institucional. El tiempo empleado para el procesado de ficheros en la practica 2 resulta notoriamente mayor debido a que en esa práctica se proporcionaba una librería adicional.

Práctica	Guardado (ms)	Preparar ficheros (ms)	JUnit (ms)
1	238,01	17,44	7688,06
2	200,74	116,85	6066,75

Figura 8: Medias en los tiempos observados en las entregas de las dos prácticas.

5. Trabajo relacionado

El uso de herramientas de apoyo en el proceso enseñanza-aprendizaje de los conceptos fundamentales de programación es un tema que ha suscitado gran interés en las últimas décadas [2]. Debido a la multitud de trabajos disponibles relacionados con la evaluación automática de ejercicios de programación [3], este artículo se va a centrar en un subconjunto representativo de las herramientas relacionadas que se encuentran integradas en la plataforma Moodle. En relación a las propuestas descritas en [6] para la evaluación automática de código fuente, EvalCode contribuye automatizando las etapas de verificación de estilo (mediante análisis estático), la fase de pruebas funcionales se implementa mediante tests de JUnit, como en otros trabajos de esta comunidad [8, 9], y el Moodle facilita el reporte de feedback al alumno. El cálculo de métricas más elaboradas mediante herramientas como Sonar por ejemplo está contemplado en el proyecto para asignaturas más avanzadas pero no está implementada en al estado actual del mismo.

Dentro de las herramientas integradas en Moodle se encuentra la extensión JavaUnitTest[1] que permite la evaluación de pequeños fragmentos de código Java utilizando test JUnit. Un enfoque similar propone CodeRunner [4], aunque este último soporta distintos lenguajes de programación. Para la evaluación de códigos más complejos existe la extensión VPL [9]. La funcionalidad principal de esta extensión es similar a la proporcionada por nuestra herramienta, aunque VPL presenta además prestaciones adicionales como soporte para distintos lenguajes de programación o detección de plagios en las entregas realizadas. Sin

[1]https://moodle.org/plugins/qtype_javaunittest

embargo, existen algunas diferencias fundamentales en el diseño de ambas herramientas.

- VPL incorpora el entorno de desarrollo (editor de código, ejecución y depuración) en Moodle, por lo que el proceso de desarrollo del alumno se realiza desde el navegador. En el caso de EvalCode, cada asignatura puede elegir su entorno de desarrollo (p.ej. Eclipse o BlueJ), ya que la extensión se centra exclusivamente en la evaluación automática del código entregado.
- VPL requiere que el profesor genere un conjunto de scripts para configurar el entorno de ejecución y la evaluación de las entregas. En este último caso, la evaluación por defecto se realiza definiendo las entradas/salidas del programa a evaluar. Para otro tipo de evaluación, se requiere modificar los scripts proporcionados [1]. En el caso de *EvalCode*, el esfuerzo del profesor se centra exclusivamente en diseñar los tests JUnit y establecer los parámetros de configuración de la tarea en Moodle.

6. Conclusiones y trabajo futuro

Este artículo describe la herramienta *EvalCode*, una extensión de Moodle para la evaluación automática de código. *EvalCode* permite analizar el código entregado por el alumno desde diferentes puntos de vista utilizando distintas herramientas. El resultado del análisis realizado por cada herramienta junto con la calificación obtenida constituyen la realimentación proporcionada al alumno.

La versión actual de nuestra herramienta permite evaluar ejercicios de programación en lenguaje Java y utiliza las herramientas de análisis JUnit y Checkstyle. La estructura interna de *EvalCode* está pensada para facilitar la integración de nuevas herramientas.

A diferencia de otras herramientas existentes, *EvalCode* permite analizar código Java arbitrariamente complejo (constituido por un conjunto de paquetes con clases y/o interfaces), el cuál puede haber sido desarrollado utilizando cualquier IDE (p.ej., Eclipse).

Otra ventaja de *EvalCode* respecto a otras herramientas similares es la simplificación de la tarea del profesor, que fundamentalmente se centra en el diseño de la clase de prueba de JUnit.

Las pruebas de campo de nuestra herramienta han resultado satisfactorias en todos los aspectos contemplados: (a) los alumnos se han adaptado fácilmente a su uso; (b) el tiempo requerido por el servidor para la evaluación de las entregas ha sido aceptable y (c) el uso de la realimentación (permitido en la segunda de las prácticas) se ha visto reflejado en una mejora de la nota obtenida.

Existen varios aspectos a ampliar o mejorar en nuestra herramienta, entre ellos los más importantes podrían ser la integración de nuevas herramientas de análisis y de control de plagio y el soporte para otros lenguajes de programación. En particular es importante añadir la posibilidad de invocar herramientas de análisis de calidad y auditoría, como Sonar por ejemplo, de modo que en cursos más avanzados los alumnos reciban retroalimentación más exhaustiva de la deuda técnica y otras figuras de mérito relacionadas con la calidad global de su trabajo.

EvalCode es una herramienta de código abierto que se encuentra en continuo desarrollo. Se invita a la comunidad docente a descargarla y contribuir a la misma. Esta herramienta se puede descargar desde https://github.com/aldeam/evalcode

Referencias

[1] Luis Arévalo, Francisco J. Rodríguez, Rafael M. Luque-Baena, Francisco Luna: Experiencia con una herramienta de pruebas de caja negra para el aprendizaje de asignaturas de programación en evaluación continua. JENUI 2017. Cáceres, pp. 61-68.

[2] Mónica Guerrero, Danny S. Guamán y Julio C. Caiza: Revisión de Herramientas de Apoyo en el Proceso de Enseñanza-Aprendizaje de Programación. Revista Politécnica 2015, Vol. 35, No. 1., pp. 82-90.

[3] Sugandha Gupta y Anamika Gupta: E-Assessment Tools for Programming Languages: A Review. ICITKM 2018, India, pp. 65-70. DOI: 10.15439/2018KM31

[4] Richard Lobb y Jenny Harlow: Coderunner: a tool for assessing computer programming skills. ACM Inroads 2016, Vol. 7, No. 1, pp. 47-51. DOI: 10.1145/2810041

[5] Ricardo Alexandre Peixoto Queirós y José Paulo Leal: PETCHA: a programming exercises teaching assistant. ACM ITiCSE 2012, New York, USA, pp. 192-197. DOI: 10.1145/2325296.2325344

[6] M. A. Pinto: Web-based system for automatic evaluation of java algorithms. Eurocon 2013, Zagreb, pp. 2123-2128. DOI: 10.1109/EUROCON.2013.6731010

[7] Dominique Thiébaut: Automatic evaluation of computer programs using Moodle's virtual programming lab (VPL) plug-in. J. Comput. Sci. Coll. 2015, Vol 30, No. 6, pp. 145-151.

[8] Agustín Cernuda del Río, Miguel García Rodríguez, Néstor García Fernández, Martín González Rodríguez. Actas de las XX JENUI. Oviedo, 9-11 de julio 2014 ISBN: 978-84-697-0774-6. Páginas: 237-243

[9] Carlos López, Raúl Marticorena, David H. Martín: Pruebas de caja negra: una experiencia real en laboratorio. JENUI 2005. Madrid, pp. 189–196.