Actas de las Jenui, vol. 3. 2018.
Páginas: 343-350

Uso de una herramienta de corrección automática en un curso de programación: Una experiencia docente

Miguel Ángel Rubio
Dpto. de Ciencias de la Computación e IA
Universidad de Granada
Granada
marubio@ugr.es

Francisco González del Valle

Universidad de Granada
Granada
pacogdv@correo.ugr.es

Resumen

Ser capaz de programar se ha convertido en una competencia necesaria en un número importante de profesiones. Como consecuencia ha aumentado la demanda de cursos de programación en todos los niveles de enseñanza.
Estos cursos suponen un desafío para los profesores involucrados ya que muchos estudiantes encuentran bastantes dificultades cuando empiezan a programar. Esto provoca un alto nivel de abandono en estas asignaturas. Para los profesores encargados de estas asignaturas suponen una alta carga de trabajo ya que deben corregir un gran número de ejercicios de programación.
Las herramientas de corrección automática son una solución para estos problemas. Por un lado, los estudiantes disponen de una retroalimentación inmediata de su trabajo. Esto facilita el aprendizaje y aumenta su motivación. Por otro lado, la carga docente del profesor disminuye en gran medida.
En este estudio describimos el proceso de implantación de una herramienta de corrección automática - CodeRunner - en un curso de programación. Los resultados han sido bastante buenos: ha aumentado la participación y ha bajado significativamente la tasa de abandono de la asignatura. Además, los estudiantes expresan una alta satisfacción con el uso de esta herramienta.

Abstract

A growing number of scientific and technological professions require the use of basic programming skills. In response to this need, educational institutions have begun to offer programming courses at all education levels.
These courses pose a challenge for the teachers involved. Many students find learning to program difficult and the failure rates tend to be high. Additionally, these courses place significant demands on the instructors' time as they must assess many programming exercises.
Automatic assessment tools offer a solution to these problems. On one hand, students using these tools have immediate feedback on their work. This facilitates learning and increases their motivation. On the other hand, teachers devote less time to assess programming exercises, time that can be used in other teaching tasks.
In this study we describe the use of an automatic assessment tool -CodeRunner- in an introductory programming course. The results have been quite good: students' participation in lab sessions increased, and the drop-out rate decreased significantly. Additionally, students express a high degree of satisfaction when using this tool.

Palabras clave

Evaluación automática, Retroalimentación automática, Moodle, Introducción a la Programación, CodeRunner.

1. Introducción

Un creciente número de profesiones requieren para su desempeño ser capaz de programar. En respuesta a esta necesidad han aparecido una gran cantidad de cursos de programación en la enseñanza primaria, secundaria y universitaria.

Estos cursos suponen un desafío para los profesores involucrados ya que muchos estudiantes encuentran bastantes dificultades cuando empiezan a programar. Se han realizado bastantes estudios pero sigue siendo cierta la afirmación de Agustín Cernuda del Río: *"Todavía no sabemos enseñar programación"* [5].

Una consecuencia es que estas asignaturas presentan una alta tasa de abandono [24]. Para un estudiante la decisión de abandonar una asignatura es compleja [17], siendo los principales desencadenantes la falta de tiempo y la falta de motivación ante la

adversidad [4]. Una manera de reducir el abandono es mediante el uso de metodologías de evaluación continua, Sorva [20] afirma que el uso de la evalución contínua facilita a los estudiantes alcanzar los objetivos de aprendizaje. Aun siendo efectiva, esta metodología supone una alta carga de trabajo para el profesor responsable de corregir los ejercicios de programación ya que debe entregar sus comentarios en un periodo de tiempo muy corto.

Las herramientas de corrección automática suponen una solución para estas dificultades. Los estudiantes se benefician de ellas al disponer de manera inmediata información sobre la corrección de los ejercicios. Esto aumenta la motivación ya que y al contestar preguntas y ver los fallos y aciertos, el estudiante desea continuar con los ejercicios [16].

Pero no sólo los alumnos se benefician de su uso, también los profesores. Al ser la corrección automática, su carga de trabajo se aligera, especialmente si tienen muchos estudiantes o si pretenden reutilizar los materiales en diferentes cursos [7].

La utilidad de los sistemas de corrección automática ha provocado un gran número de estudios sobre su uso en asignaturas de programación. Estos estudios comenzaron ya en la década de los 60 [7] y continúan en la actualidad tanto a nivel nacional como internacional.

A nivel nacional podemos destacar el estudio de de-la-Fuente-Valentín et al [12] que diseñaron e implementaron un sistema de evaluación automática en la Universidad Carlos III. Rodriguez del Pino et al [19] desarrollaron una extensión de Moodle para la realización de pruebas de caja negra. La facilidad de instalación y su integración con Moodle ha facilitado su adopción por múltiples instituciones [21]. El sistema CAC++ desarrollado por Delgado et al [6] destaca por su capacidad de trabajar a nivel del árbol de sintaxis abstracta pudiendo evaluar la estructura del código.

A nivel internacional destacan entre las aplicaciones comerciales Turingscraft Codelab [3] una aplicación que lleva en funcionamiento desde 2002 y que han utilizado más de trescientos mil estudiantes de veinte países distintos. Entre las aplicaciones de código libre destacan Codingbat (http://codingbat.com) un sistema con una arquitectura bastante sencilla pero que ha servido de inspiración a varios sistemas posteriores. Un ejemplo de sistema inspirado por Codingbat es Cloudcoder [16] diseñado específicamente para facilitar la investigación en *Learning Analytics.* Por último nbgrader [8] destaca por ser un sistema basado en los populares Jupyter notebooks.

Los lectores interesados pueden consultar algunas de las revisiones de la literatura realizada en los últimos años. Destacan las de Ihantola et al [10] por la gran cantidad de artículos analizados y la de Keuning et al [11] por la metodología propuesta para la clasificación de este tipo de herramientas.

A pesar de la proliferación de herramientas de este tipo todavía falta mucho para que su uso sea habitual en los cursos de introducción a la programación. Quian y Lehman afirman que uno de los principales problemas en el ámbito de la investigación en la enseñanza de la programación es que hay una gran variedad de herramientas distintas pero se realizan pocos estudios destinados a validarlas y diseminarlas [18]. Ihantola et al [9] se pregunta por qué se dedican tantos esfuerzos al diseño e implementación de nuevas herramientas con un ciclo de vida bastante corto en vez de intentar validar algunas de las herramientas ya existentes.

Estos autores coinciden en la necesidad de realizar estudios para evaluar el impacto que tienen estas herramientas y el esfuerzo necesario para introducirlas en condiciones docentes reales. Un estudio que destaca en este sentido es el realizado en la Universidad de Extremadura por Arévalo et al [2] con la herramienta VPL. En este estudio describen cual fue el proceso de implantación de la herramienta en distintas asignaturas y cuales fueron los resultados obtenidos.

El objetivo del estudio que presentamos es similar: comprobar si el uso de una herramienta de corrección automática en una asignatura de introducción a la programación supone una mejora para los estudiantes y los profesores involucrados en dicha asignatura.

A continuación describiremos en detalle la herramienta de corrección automática que hemos utilizado: CodeRunner [14]. En la sección tres describimos la metodología seguida en el estudio. En la sección cuatro mostramos los principales resultados obtenidos que son analizados en la sección cinco.

2. CodeRunner

CodeRunner (http://coderunner.org.nz/) es un entorno de corrección automática de ejercicios de programación desarrollado en la Universidad de Canterbury. Es una aplicación de código abierto que se implementa como una extensión de Moodle (http://moodle.org).

En la actualidad CodeRunner permite la evaluación de código en una gran cantidad de lenguajes de programación: Python, C++, Java, PHP, Octave… Además, es fácilmente extensible a nuevos lenguajes. Los autores de esta comunicación han extendido su implementación de CodeRunner para que permita la corrección de programas escritos en Fortran 90. El proceso completo no llevó más de cinco horas.

Algunas de las ventajas de CodeRunner son:

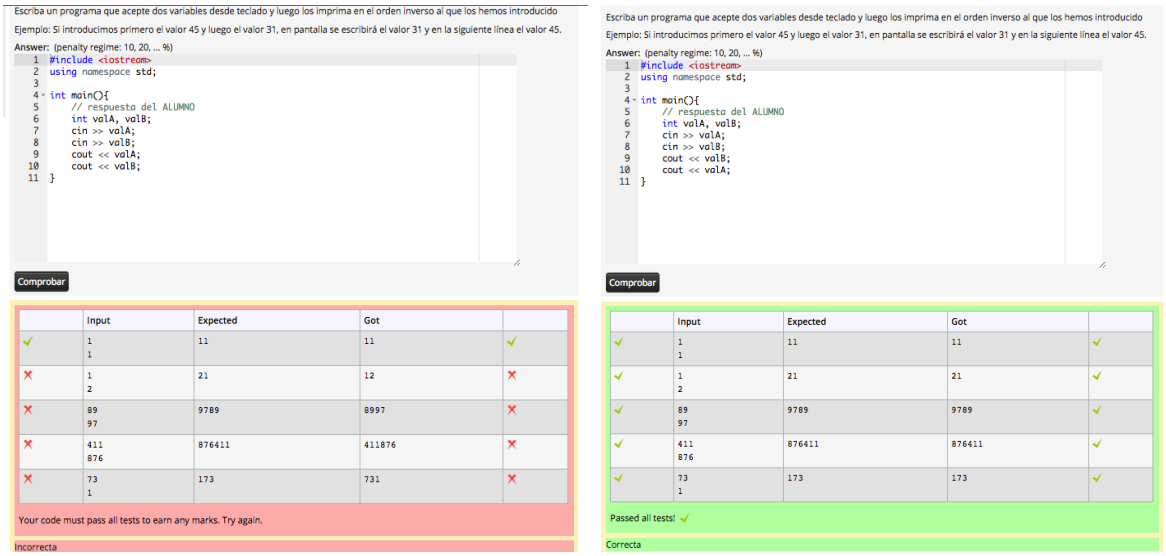

Figura 1: Retroalimentación de un ejercicio en CodeRunner: respuesta incorrecta (izquierda), respuesta correcta (derecha).

- Permitir el uso de expresiones regulares para hacer más flexibles tanto las entradas como las salidas de los casos de prueba.
- Puede realizar análisis tanto estáticos como dinámicos del código entregado por el estudiante.
- Dispone de una interfaz fácil de utilizar que no requiere programación en los casos más sencillos.

Pero no todo son ventajas, existen una serie de inconvenientes y limitaciones que conviene no perder de vista:

- No es posible evaluar de manera automática la calidad de los comentarios introducidos
- La respuesta en código debe de ser simple, en el sentido que no podemos evaluar cuando la respuesta debe estar incluida en varios ficheros
- Evaluar salidas de código que tengan elementos gráficos es complicado, aunque se están haciendo avances en este terreno.

El lector interesado puede consultar una descripción más completa de la herramienta en [14].

A continuación, pasamos a describir los aspectos más relevantes del trabajo con CodeRunner desde el punto de vista del estudiante, el profesor y el administrador de sistemas.

2.1. Desde el punto de vista del estudiante

Para nuestros estudiantes trabajar con CodeRunner no supone una gran novedad. El sistema está integrado totalmente en Moodle y nuestros estudiantes están acostumbrados a trabajar con este sistema.

El alumno comienza accediendo al guion de ejercicios y ve el enunciado de las distintas preguntas. Normalmente proporcionamos en los primeros ejercicios un pequeño esquema de código en la ventana de respuesta. Esto ayuda al alumno a concentrarse sólo en la cuestión y le facilita empezar a responderla.

Cuando el alumno concluye el ejercicio envía la solución al sistema para que compruebe su validez. En pocos segundos el sistema le indica si es correcta (Figura 1 a la derecha) o incorrecta (Figura 1 a la izquierda). Si la respuesta es incorrecta debido a un fallo de compilación el sistema muestra los mensajes de error proporcionados por el compilador. Cuando el código compila, pero no pasa algunos de los casos de prueba CodeRunner muestra qué casos de prueba se han pasado y cuáles no. El sistema permite especificar casos de prueba invisibles para el alumno para evitar estrategias de programación indeseables.

El diseño y los colores de las respuestas de CodeRunner se han escogido cuidadosamente y nuestra experiencia es que motiva enormemente a los estudiantes.

2.2. Desde el punto de vista del profesor

El profesor tiene una gran variedad de preguntas disponibles en CodeRunner: desde simplemente rellenar un espacio en blanco a realizar programas complejos. Al poder combinarse con otros formatos de preguntas ya disponible en Moodle permite una gran flexibilidad en el diseño de guiones o exámenes.

CodeRunner se ha diseñado principalmente para su uso en cursos introductorios de Programación y es aquí donde destaca. Resulta extremadamente sencillo diseñar ejercicios donde el alumno conteste y desarrolle programas sencillos que enseñen las propiedades de los lenguajes utilizados, así como sus técnicas.

Aun así, también existen ejemplos de aplicación de CodeRunner para probar autómatas con una máquina de estados finito, o en inteligencia artificial e incluso en cursos de programación web para evaluar sitios web.

Para la creación de un cuestionario o un examen para el alumno, lo primero que hay que hacer es crear las preguntas que el profesor desea formularle a los estudiantes.

En el caso de los ejercicios más sencillos una vez seleccionada la categoría es necesario indicar el lenguaje de programación que se va a utilizar, el enunciado de la pregunta y los casos de prueba que indicarán si la respuesta es correcta o no (Figura 2).

Si se desea utilizar todo el potencial de CodeRunner se deben especificar opciones adicionales que permiten especificar las condiciones que debe cumplir el código fuente, el uso de expresiones regulares, código de apoyo para el estudiante, opciones específicas del compilador, ficheros de datos que se desean utilizar…

Los sistemas de corrección automática pueden alentar al alumno a buscar la solución de un ejercicio mediante la heurística de prueba y error –técnica que Spacco [1] denomina *programming by Brownian motion*. CodeRunner implementa varias opciones para evitar este tipo de estrategias. Es posible especificar varias condiciones para que se ejecute un test y es posible ocultar algunos test para que el estudiante no disponga de la batería de test completa.

Referimos al lector interesado en conocer todas las funcionalidades de CodeRunner a la documentación que podrá encontrar en la página web oficial (http://coderunner.org.nz/).

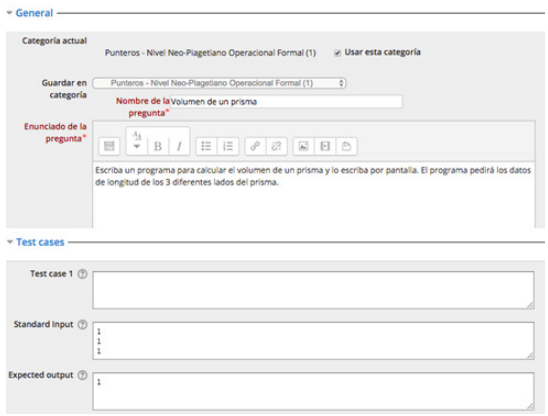

Figura 2: Enunciado de un ejercicio y casos de prueba.

2.3. Desde el punto de vista del administrador

La arquitectura de CodeRunner es bastante simple como se muestra en la Figura 3. El sistema consta de un entorno Moodle y un sandbox responsable de ejecutar el código creado por los estudiantes.

El entorno Moodle y el sandbox pueden encontrarse en servidores distintos (se recomienda hacerlo así) y la autentificación se realiza mediante tokens.

Por motivos de seguridad es recomendable restringir el acceso a internet del sandbox. La situación ideal es que el servidor con el sandbox se dedique exclusivamente a este fin.

Los requisitos de hardware no son grandes. En nuestro caso dos ordenadores con procesadores Intel Core I3 con microarquitectura Haswell (2014) y cuatro gigabytes de memoria RAM fueron más que suficiente para correr todos los procesos sin ningún retraso.

Cuando el sistema evalúa la respuesta de un estudiante los pasos que se siguen son los siguientes:

1. Para cada caso de testeo, el Motor de la Plantilla une la respuesta introducida por el estudiante con la plantilla de la pregunta y con el código para el testeo particular para tener un programa ejecutable, entendiéndose por tal un programa que puede ejecutarse tras un paso previo de compilación.
2. El programa ejecutable pasa por la Sandbox que compila el programa si fuera necesario y lo ejecuta usando las entradas especificada por los casos de test.

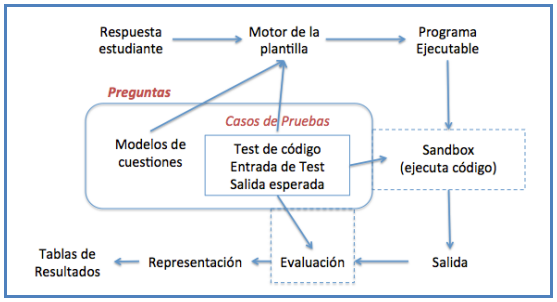

Figura 3: Arquitectura del sistema CodeRunner

3. La salida pasa a un proceso de evaluación que además espera la respuesta específica para cada caso de testeo. CodeRunner nos permite varios modelos de evaluación, aunque el más común es el conocido como "coincidencia exacta".
4. La salida del proceso de evaluación es un objeto que contiene entre otros elementos, los atributos "expected" y "got".
5. El proceso se repite para todos los test implementados, de manera que al final queda una matriz de objetos como los descritos en el punto anterior.
6. Una vez finalizados todos los test, CodeRunner ejecuta la representación para mostrar al usuario una tabla de resultados, de manera que los test evaluados tengan el formato adecuado.

3. Metodología

El estudio se realizó en una asignatura de introducción a la programación en el grado de Matemáticas de la Universidad de Granada. Dicha asignatura es la primera de dos asignaturas dedicadas al aprendizaje de la programación en C++.

La asignatura objeto del estudio consta de 6 créditos ECTS y se cubren los primeros conceptos de la programación estructurada en C++: variables, estructura secuencial, entradas y salidas, condicionales, estructuras repetitivas y arrays.

Las clases se dividen al 50% entre clases magistrales y prácticas de programación en el laboratorio. Las prácticas de laboratorio consisten en un conjunto de guiones con ejercicios de programación. En el diseño de los ejercicios de programación se ha seguido el modelo de aprendizaje Neo-Piagetiano propuesto por Lister [13]. Este modelo propone distintas categorías de ejercicios en las distintas fases de aprendizaje de los estudiantes de programación: sensorimotor, preoperacional, operacional concreto y operacional abstracto.

Cada tema tiene dos guiones asociados: uno básico y uno avanzado. Solo el guion básico se evalúa y califica. La calificación obtenida en los guiones supone un 10% de la nota final.

La intervención recogida en este estudio afecta exclusivamente a la docencia en prácticas. El grupo de control estaba formado por grupos de prácticas en el que los estudiantes realizaron los ejercicios en un entorno de desarrollo clásico. El grupo experimental estaba formado por grupos de prácticas en las que los estudiantes realizaban los mismos ejercicios en el entorno CodeRunner. Los estudiantes se asignaron por orden alfabético al grupo experimental y al grupo control.

Se recogieron datos sobre la asistencia a prácticas, participación en el examen final, notas finales y una encuesta con una única pregunta: *¿Te ha parecido útil usar CodeRunner en prácticas?*

Se calculó la tasa de abandono de la asignatura como el porcentaje de estudiantes matriculados que no se presentaron al examen final. La participación en los grupos de prácticas se calculó como la media de participación en cada uno de los módulos.

Solo se analizaron los guiones básicos ya que eran los únicos que se evaluaban. Para cada ejercicio de cada uno de los guiones se calculó el índice de dificultad: el porcentaje de estudiantes que han participado y han completado correctamente el ejercicio. También se calculó la eficiencia discriminativa: la correlación entre la calificación de la pregunta y la del guion en su conjunto.

4. Resultados

En este apartado procederemos a describir los principales resultados obtenidos en este estudio. Comenzaremos describiendo los resultados académicos obtenidos y concluiremos describiendo cual ha sido la percepción de los estudiantes y los profesores involucrados en este curso.

El resultado más destacable es la gran disminución de la tasa de abandono de la asignatura. Como se puede observar en la Figura 4 la tasa de abandono del grupo de control se sitúa en el 40% mientras que en el grupo experimental baja al 10%. Esta importante diferencia es estadísticamente significativa con un p-value de 0,006.

Si comparamos las calificaciones finales se observa que la nota media de los estudiantes que se presentan es 5,49 en el grupo de control y 5,13 en el grupo experimental.

En la Figura 5 se analiza la evolución temporal de la participación en prácticas. Podemos observar tanto el grupo de control como el grupo experimental

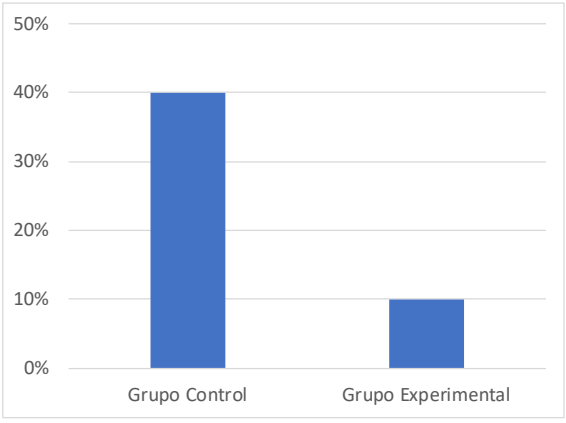

Figura 4: Tasa de abandono de la asignatura para el grupo de control y el grupo experimental.

comienzan en un nivel similar en torno al 90%. A medida que avanza el curso el grupo experimental mantiene este nivel de participación mientras que en el grupo de control la participación disminuye de manera continuada terminando el curso en torno al 60%.

A continuación, centramos nuestro análisis en el grupo experimental y analizamos los patrones de resolución de los distintos ejercicios. Se observa que en cada guion la mayoría de los estudiantes son capaces de completar los ejercicios de menor dificultad, pero sólo la mitad de los estudiantes resuelven los ejercicios de mayor dificultad. En la Figura 6 se muestran los resultados obtenidos en el guion de estructuras condicionales. Hemos de comentar que en el caso de este guion el ejercicio siete se hizo en clase como ejemplo.

Concluimos nuestro análisis con la percepción de los estudiantes respecto del uso de esta herramienta (Figura 7). A la pregunta *¿Te ha parecido útil usar CodeRunner en prácticas?* un 70% de los estudiantes responde que les ha parecido muy útil. No hay estudiantes que respondan que les ha parecido poco útil o nada útil.

La percepción de los profesores fue igualmente positiva. La dinámica de las clases de prácticas cambia radicalmente cuando se utiliza CodeRunner. Los estudiantes están mucho más motivados y realizan menos preguntas ya que disponen de la información proporcionada por el sistema. Además, la posibilidad de seguir realizando los ejercicios en casa hace que aumente considerablemente el número de ejercicios resueltos correctamente por los estudiantes. En algunos casos los estudiantes se fijan como objetivo completar todos los guiones correctamente considerándolo algo similar a los trofeos en los videojuegos.

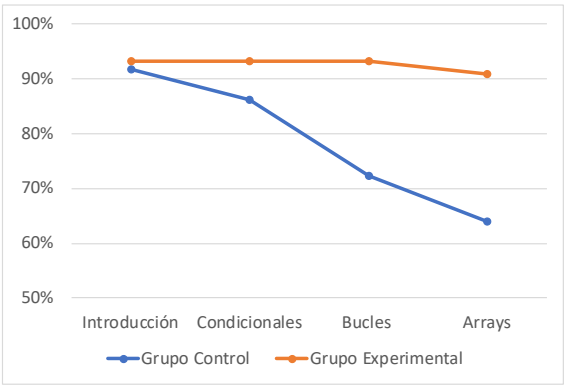

Figura 5: Evolución de la participación en los distintos módulos de prácticas para el grupo de control y el grupo experimental.

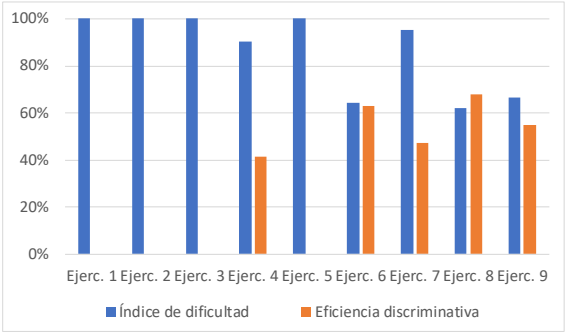

Figura 6: Índice de dificultad y eficiencia discriminativa para los ejercicios del guion de estructuras condicionales.

5. Discusión

Los resultados obtenidos son muy esperanzadores. Una reducción del 75% en la tasa de abandono es una mejora considerable en el rendimiento académico. La reducción en las calificaciones finales se puede explicar si suponemos que los estudiantes deciden perseverar en la asignatura, aun no teniendo un alto dominio de los contenidos.

Si centramos nuestra atención en la evolución de la participación en clase vemos que el uso de CodeRunner se traduce en una mayor participación de los estudiantes. En el grupo de control los estudiantes van abandonando la asignatura a medida que aumenta la complejidad del temario. En el grupo experimental la participación se mantiene prácticamente constante.

Los resultados de la encuesta de los estudiantes corroboran estos resultados. Todos los estudiantes que trabajaron con CodeRunner considera que ha resultado útil, y la mayoría muestra una gran satisfacción.

La percepción por parte del profesorado también ha

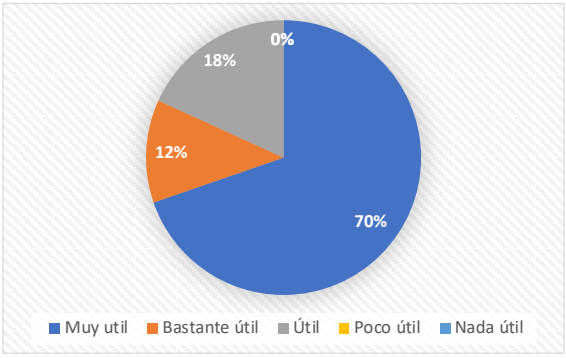

Figura 7: Respuesta de los estudiantes a la pregunta ¿Te ha parecido útil usar CodeRunner en prácticas?

sido muy positiva. La interfaz de uso y de diseño de las preguntas resulta muy intuitiva, siendo necesario únicamente un ligero conocimiento de expresiones regulares. La integración con Moodle hacer que su mantenimiento sea sencillo y fácil su integración con el trabajo diario de la asignatura.

Estos resultados concuerdan con los obtenidos por otros autores. Barr y Trytten [3] implementaron un sistema de corrección automática -CodeLab- en la universidad de Oklahoma y en el Union College obteniendo una mejora en el desempeño de los estudiantes en casi todos los casos. Arévalo et al [2] utilizaron en varias asignaturas de la Universidad de Extremadura el sistema de corrección automático VPL obteniendo resultados muy positivos: los estudiantes estaban bastantes satisfechos con el sistema y los resultados académicos mejoraron. Des et al. [22] realizaron un estudio sobre el uso de un sistema de corrección automática en una asignatura de introducción a la programación y describen una mejora en los resultados, aunque no realizaron un análisis estadístico. Cisar et al [15] realizaron un estudio cuasi-experimental sobre un sistema de corrección automático adaptativo obteniendo una mejora estadísticamente significativa.

Lo coincidencia en resultados positivos obtenidos en distintos estudios indican que este tipo de herramientas suponen una mejora significativa en la enseñanza de la programación. Estas herramientas motivan a los estudiantes, mejoran sus resultados académicos y facilitan la labor del profesorado.

Este estudio presenta una limitación: los estudiantes del grupo experimental eran conscientes de que estaban utilizando una herramienta novedosa ya que podían comparar sus prácticas con las del grupo de control. Esto podría afectar tanto a la motivación como a la persistencia en el curso debido al efecto Hawthorne [23]. Consideramos que en el caso de nuestro estudio este efecto no es relevante ya que la diferencia en los resultados es demasiado grande para deberse únicamente a la novedad del instrumento.

6. Conclusiones

Esta comunicación presenta una experiencia docente consistente en la implantación de una herramienta de corrección automática -CodeRunner- en la docencia de una asignatura de introducción a la programación.

Los resultados obtenidos son francamente positivos. Ha disminuido significativamente la tasa de abandono de la asignatura y ha aumentado la participación de los estudiantes a lo largo del curso.

La percepción de los estudiantes también es muy positiva, ningún estudiante cree que la herramienta utilizada fuese inútil.

La opinión del profesorado involucrado en la docencia es muy favorable. La interfaz de uso y de diseño de las preguntas resulta muy intuitiva. La integración con Moodle hacer que su mantenimiento sea sencillo y fácil su integración con el trabajo diario de la asignatura.

Preparar los guiones de ejercicios y las preguntas de examen no exige una gran cantidad de tiempo. La carga de trabajo utilizando esta herramienta de corrección automática es menor que la carga de trabajo siguiendo métodos tradicionales.

Diversos autores del campo de la enseñanza de la informática han llamado la atención sobre la falta de estudios que validen las herramientas de apoyo a la docencia disponibles en la actualidad. Esta experiencia docente pretende ser un pequeño paso en esta dirección.

Agradecimientos

Este trabajo ha sido financiado por el proyecto DPI2015-69585-R a través del Ministerio de Economía y Competitividad (MINECO).

Referencias

[1] Alireza Ahadi, Raymond Lister, Heikki Haapala, y Arto Vihavainen. 2015. Exploring Machine Learning Methods to Automatically Identify Students in Need of Assistance. In *Proceedings of the Eleventh Annual International Conference on International Computing Education Research* (ICER '15), 121–130.

[2] Luis Arévalo, Francisco J. Rodríguez, Rafael M. Luque-Baena, y Francisco Luna. 2017. Experiencia con una herramienta de pruebas de caja negra para el aprendizaje de asignaturas de programación en evaluación continua. *Actas de las Jornadas sobre Enseñanza Universitaria de la Informática* 2, 0.

[3] Valerie Barr y Deborah Trytten. 2016. Using Turing's Craft Codelab to Support CS1 Students As They Learn to Program. *ACM Inroads* 7, 2: 67–75.

[4] Karo Castro-Wunsch, Alireza Ahadi, y Andrew Petersen. 2017. Evaluating Neural Networks As a Method for Identifying Students in Need of Assistance. In *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education* (SIGCSE '17), 111–116.

[5] Agustín Cernuda del Río. 2017. Todavía no sabemos enseñar programación. *ReVisión* 10, 1.

[6] Pedro Delgado-Pérez, Inmaculada Medina-Bulo, y Daniel Pérez-Caro. 2017. Biblioteca CAC++ para la corrección automática de prácticas de programación en C++. *Actas de las*

Jornadas sobre Enseñanza Universitaria de la Informática 2, 0.

[7] Christopher Douce, David Livingstone, y James Orwell. 2005. Automatic test-based assessment of programming: A review. *Journal on Educational Resources in Computing (JERIC)* 5, 3: 4.

[8] Jessica B. Hamrick. 2016. Creating y Grading IPython/Jupyter Notebook Assignments with NbGrader. In *Proceedings of the 47th ACM Technical Symposium on Computing Science Education* (SIGCSE '16), 242–242.

[9] Petri Ihantola, Tuukka Ahoniemi, Ville Karavirta, y Otto Seppälä. 2010. Review of Recent Systems for Automatic Assessment of Programming Assignments. In *Proceedings of the 10th Koli Calling International Conference on Computing Education Research* (Koli Calling '10), 86–93.

[10] Petri Ihantola, Arto Vihavainen, Alireza Ahadi, Matthew Butler, Jürgen Börstler, Stephen H. Edwards, Essi Isohanni, Ari Korhonen, Andrew Petersen, Kelly Rivers, Miguel Ángel Rubio, Judy Sheard, Bronius Skupas, Jaime Spacco, Claudia Szabo, y Daniel Toll. 2015. Educational Data Mining and Learning Analytics in Programming: Literature Review and Case Studies. In *Proceedings of the 2015 ITiCSE on Working Group Reports* (ITICSE-WGR '15), 41–63.

[11] Hieke Keuning, Johan Jeuring, y Bastiaan Heeren. 2016. Towards a Systematic Review of Automated Feedback Generation for Programming Exercises. In *Proceedings of the 2016 ACM Conference on Innovation and Technology in Computer Science Education* (ITiCSE '16), 41–46.

[12] Luis De-La-Fuente-Valentín, Abelardo Pardo, y Carlos Delgado Kloos. 2013. Addressing drop-out and sustained effort issues with large practical groups using an automated delivery and assessment system. *Computers & Education* 61: 33–42.

[13] Raymond Lister. 2011. Concrete and other neo-Piagetian forms of reasoning in the novice programmer. In *Proceedings of the Thirteenth Australasian Computing Education Conference-Volume 114*, 9–18.

[14] Richard Lobb y Jenny Harlow. 2016. Coderunner: A Tool for Assessing Computer Programming Skills. *ACM Inroads* 7, 1: 47–51.

[15] Sanja Maravić Čisar, Petar Čisar, y Robert Pinter. 2016. Evaluation of knowledge in Object Oriented Programming course with computer adaptive tests. *Computers & Education* 92–93: 142 – 160.

[16] Andrei Papancea, Jaime Spacco, y David Hovemeyer. 2013. An Open Platform for Managing Short Programming Exercises. In *Proceedings of the Ninth Annual International ACM Conference on International Computing Education Research* (ICER '13), 47–52.

[17] Leo Porter y Beth Simon. 2013. Retaining Nearly One-third More Majors with a Trio of Instructional Best Practices in CS1. In *Proceeding of the 44th ACM Technical Symposium on Computer Science Education* (SIGCSE '13), 165–170.

[18] Yizhou Qian y James Lehman. 2017. Students' Misconceptions and Other Difficulties in Introductory Programming: A Literature Review. *ACM Trans. Comput. Educ.* 18, 1: 1:1–1:24.

[19] JC Rodriguez del Pino, E Royo Rubio, y Z. Hernandez Figueroa. 2010. VPL: Laboratorio virtual de programación para Moodle. *Actas de las XVI Jornadas de Enseñanza Universitaria de Informática, Jenui*: 429–435.

[20] Juha Sorva. 2012. Visual program simulation in introductory programming education.

[21] Dominique Thiébaut. 2015. Automatic Evaluation of Computer Programs Using Moodle's Virtual Programming Lab (VPL) Plug-in. *J. Comput. Sci. Coll.* 30, 6: 145–151.

[22] Des Traynor, Susan Bergin, y J Paul Gibson. 2006. Automated assessment in CS1. In *Proceedings of the 8th Australasian Conference on Computing Education-Volume 52*, 223–228.

[23] Arto Vihavainen, Jonne Airaksinen, y Christopher Watson. 2014. A Systematic Review of Approaches for Teaching Introductory Programming and Their Influence on Success. In *Proceedings of the Tenth Annual Conference on International Computing Education Research* (ICER '14), 19–26.

[24] Christopher Watson y Frederick W. B. Li. 2014. Failure Rates in Introductory Programming Revisited. In *Proceedings of the 2014 Conference on Innovation & Technology in Computer Science Education* (ITiCSE '14), 39–44.