INVESTIGACIÓN e INNOVACIÓN en INGENERÍAS
ISSN2344-8652

Estrategia de enseñanza basada en la colaboración y la evaluación automática de código fuente en un curso de programación CS1

Strategy based on collaboration and the automatic code assessment in a programming CS1 course

Carlos Giovanny Hidalgo Suarez **Víctor Andrés Bucheli Guerrero**

Universidad del Valle, Colombia

Felipe Restrepo Calle **Fabio Augusto González Osorio**

Universidad Nacional de Colombia, Colombia

OPEN ACCESS

Recibido: 9/09/2020
Aceptado: 27/11/2020
Publicado: 04/01/2021

Correspondencia de autores:
carlos.hidalgo@correounivalle.edu.co

Resumen

Objetivo: Proponer una estrategia de colaboración respaldada por una herramienta de software de evaluación automática que permita mejorar las habilidades, las calificaciones y los intentos de tiempo para resolver un problema de programación. **Metodología:** Se integró la colaboración y la evaluación automática de código fuente para una actividad de programación que permitió responder las siguientes cuestiones: ¿cuánto tiempo en promedio, se reduce la solución de una tarea de programación mediante una estrategia colaborativa apoyada por una herramienta de evaluación automática de código?, y ¿cuánto se incrementa en la calificación promedio de una tarea de programación utilizando una estrategia colaborativa apoyada por una herramienta de evaluación automática de código? **Resultados:** Los experimentos de este trabajo, demuestran que el uso de la colaboración y la evaluación de código fuente automático, mejora las calificaciones en un 50% y afianzan las habilidades de programación, permitiendo intercambiar conocimientos para resolver una tarea de programación en menor tiempo. **Conclusiones:** El desarrollo de estrategias que integran la colaboración y la evaluación automática de código impactan positivamente en el proceso de aprendizaje de programación, mejorando significativamente las calificaciones d el e studiante, a demás de habilidades interpersonales que incentivan a mejorar los cursos de programación.

Palabras clave: Educación tecnológica, evaluación automática de código fuente, programación de computadores, CS1, aprendizaje colaborativo.

Abstract

Objective: Propose a collaboration strategy supported by an automated assessment software tool that improves skills, qualifications, and time attempts to solve a programming problem. **Methodology:** A study was designed with two groups of students with the goal of answering the following questions. How much time on average, is the time required for solving a programming task reduced using a collaborative strategy supported by an automatic code evaluation tool? And how much does the average grade for a programming task increase on average, using a collaborative strategy supported by an automatic code evaluation tool? **Results:** The results of the study demonstrate that the use of collaboration strategies in conjunction with and automatic evaluation tool, improves the grades by 50% and improves the student programming skills, by supporting knowledge sharing to solve a programming task in less time. **Conclusions:** The development of strategies that integrate collaboration and the automatic code assessment positively impact the programming learning process of programming, significantly improving the student's qualification, as well as interpersonal skills that encourage the improvement of programming courses.

Keywords: Technological education, automatic code assessment tool, introductory programming course.

Como citar (IEEE): C. Hidalgo-suarez., V. Bucheli-Guerrero., F. Restrepo-Calle., y F. González-Osorio "Estrategia de enseñanza basada en la colaboración y la evaluación automática de código fuente en un curso de programación CS1", *Investigación e Innovación en Ingenierías*, vol. 9, n°1, 50-60, 2021. DOI:https://doi.org/10.17081/invinno.9.1.4185

Introducción

El curso de Introducción a la Programación CS1, es un requisito que deben cumplir los estudiantes de Ingeniería de Sistemas. En este curso los estudiantes deben adquirir habilidades básicas para aplicar un paradigma de programación en la solución de problemas lógicos y de abstracción, mediante el uso de un lenguaje de programación [1, 2]. El diseño del curso se compone de competencias, logros de aprendizaje que cada estudiante debe alcanzar a través de criterios de evaluación sumativa [3, 4]; propuestos en actividades, talleres y exámenes, que permiten acumular una nota para conocer si el estudiante aprueba o no el curso.

La metodología tradicional de enseñanza-aprendizaje del curso CS1 consta de clases presenciales donde el profesor dirige el curso de forma práctica-teórica, según los contenidos del programa académico del programa de Ingeniería de la Universidad del Valle. Así, algunos estudiantes adquieren las bases necesarias para aprobar la asignatura y continuar con el curso CS2. Sin embargo, en el curso de programación se observan las mayores tasas de deserción, absentismo y bajas calificaciones [5, 6, 7, 8]. Lo que podría estar causado por el método de aprendizaje actual, que no cumple su objetivo. En este sentido, es necesario contar con nuevas estrategias de aprendizaje que permitan a los estudiantes, mejorar calificaciones, incrementar la motivación hacia la práctica y explotar sus habilidades [9].

La colaboración, como estrategia de aprendizaje en el aula, ha tenido buenos resultados en cursos de programación [10, 11, 12, 13], mejorando desde aspectos académicos hasta habilidades a nivel personal. Sin embargo, identificar en una actividad de programación el trabajo individual que aporta cada estudiante sobre el grupo, es una tarea complicada para el profesor. Por esta razón, es necesario soportar este proceso colaborativo a través de herramientas que permitan evaluar y retroalimentar automáticamente de forma integral, considerando no sólo la evaluación sumativa, sino la evaluación formativa también. Se deben considerar características propias del código fuente, como la validación sintáctica y semántica de un programa de computador, el trabajo individual, el tiempo en la solución de una tarea de programación, entre otras características para evaluar y valorar el trabajo de cada estudiante [14,15].

En este artículo se propone una estrategia para estimular el desarrollo de las habilidades programación y en la obtención de mejores calificaciones académicas por parte de los estudiantes del curso CS1, a través de una actividad colaborativa soportada por una herramienta de evaluación automática de código fuente. Se realizó un experimento con dos grupos del curso CS1. Durante el proceso los estudiantes tuvieron que resolver dos actividades: en la primera actividad, todos los estudiantes resolvieron una tarea de programación de la forma tradicional, es decir cada estudiante trabajo de forma individual en un computador y realizo la entrega del ejercicio a través de la plataforma Moodle. La segunda actividad consistió, en que los dos grupos CS1 usaron una herramienta de evaluación de código automática, pero en uno de los grupos se usó la colaboración como estrategia de apoyo adicional.

Los resultados encontrados con los experimentos demuestran que el uso de la colaboración y la evaluación de código fuente automático, mejoran las notas y afianzan las habilidades de programación, además, permiten intercambiar conocimientos para resolver una tarea de programación, y mejoran el tiempo de entrega de una tarea de programación.

Trabajos relacionados

Las herramientas creadas para la evaluación automática usualmente adoptan un mecanismo de prueba dinámica que ejecuta los programas de los alumnos a través de un conjunto de datos de prueba [16,18]. Por ejemplo, en la plataforma INGInious [19], que proporciona retroalimentación inmediata al estudiante, el programa es probado comparando símbolo por símbolo los resultados generados con los resultados esperados. INGInious permite al estudiante realizar varios intentos de respuesta, pero sólo los restringe a un número máximo para forzarlo a pensar exhaustivamente en la operación de su programa antes de someter su respuesta a evaluación. En la revisión de herramientas que evalúa automáticamente el código fuente, se encontró:

TRY-platform [20], es un sistema pensado para incluir la adecuación de los conceptos de prueba de programas representados por el acrónimo SPRAE (Specification, Premeditation, Repeatability, Accountability, Economy), para especificar un ciclo de vida de tareas de programación que son susceptibles de ser calificadas automáticamente. El ciclo obtenido incluye tres etapas: (1) especificación de tareas (qué probar, cómo probar, plan de calificación); (2) distribución de tareas (publicar en repositorio público, los requisitos de aprobación, definir los probadores, definir las entradas y capturar las salidas, calcular los puntajes); (3) probar los programas de acuerdo con el plan establecido.

CourseMaker [21], desarrollada en Java, usando todas las herramientas de diseño orientado a objetos y patrones para obtener modularidad y flexibilidad. Realiza pruebas dinámicas de funcionalidad y de eficiencia y pruebas estáticas que incluyen análisis de estilo y detección de plagio. Permite que el número de intentos y el nivel de retroalimentación sean definidos por el autor del problema. Este sistema registra la estadística de errores de compilación por alumno en una base de datos MySQL. Este registro es usado posteriormente para determinar los contenidos del curso que deben ser reforzados.

TRAKLA [22], es una herramienta de evaluación automática aplicada a la simulación gráfica de algoritmos. El alumno simula un algoritmo arrastrando y soltando en una representación gráfica de estructuras de datos. Esta herramienta permite trabajar en grupos, pero no permite evaluar el código fuente.

I-MINDS [23], es un software para la gestión inteligente de aulas o grupos en línea, le permite revisar las actividades de programación en tiempo real y fuera de línea, lo que facilita la práctica del estudiante. Su tecnología se basa en agentes de software inteligentes que interactúan con los usuarios de forma autónoma en un chatbot. Para evaluar el código fuente utilizando el juez virtual DOMJudge [24], permite a los estudiantes presentar sus soluciones para los problemas de programación planteados y evaluarlos de inmediato.

La plataforma UNCode [25], ofrece la posibilidad de evaluar el código a través del método de comparación de entradas y salidas de compilación de código fuente. El entorno tiene diferentes utilidades para hacer comentarios sumativos a través de los veredictos del programa, que indican si la sintaxis, la semántica y la eficiencia del programa son correctas o presentan algún tipo de error. Con base en estos veredictos, se asigna una calificación a las soluciones propuestas por los estudiantes. Este entorno incluye mecanismos como el análisis estático y la retroalimentación con cuestionarios (minitest) que ayudan a los estudiantes a mejorar las soluciones de programación propuestas.

Metodología

En esta sección, se identifican: las preguntas de interés que motivaron a integrar la colaboración y la evaluación de código para una actividad de programación. La selección de los grupos de programación que permitieron realizar experimentos que arrojaron los resultados. Por último, las fases de experimentación que se realizaron con los grupos seleccionados.

A. Preguntas de interés

Este trabajo se fundamenta a partir de que el curso CS1 es uno de los cursos con mayor tasa de mortalidad académica y absentismo, entonces, es importante que se integren nuevas estrategias de aprendizaje basadas en la colaboración, y herramientas que soporten la evaluación de código fuente automáticamente, para mejorar el rendimiento académico. Sin embargo, de implementar una estrategia que integre la colaboración y la evaluación automática, ¿cuánto tiempo en promedio, se reduce la solución de una tarea de programación mediante una estrategia colaborativa apoyada por una herramienta de evaluación automática de código?, y ¿cuánto se incrementa la calificación promedio de una tarea de programación, utilizando una estrategia colaborativa apoyada por una herramienta de evaluación automática de código? En el desarrollo del documento se da respuesta a estas preguntas.

B. Población y muestra

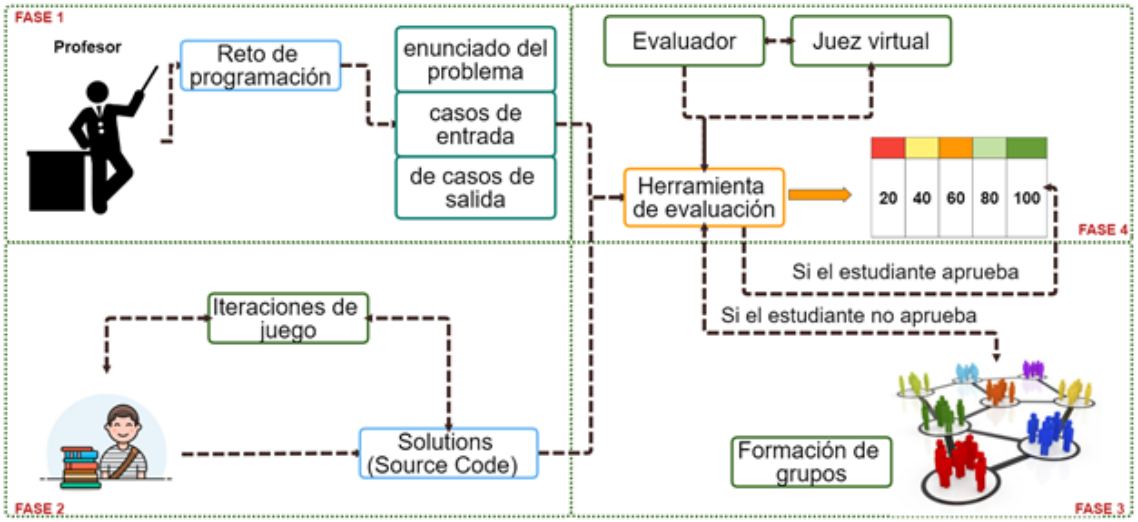
El CS1 es el primer curso de programación del plan de estudios del programa de Ingeniería de Sistemas de la Universidad del Valle, Cali-Colombia. Este curso consta de 4 horas de clase por semana (2 teóricas y 2 prácticas) y 8 horas de trabajo autónomo. En total, los estudiantes tienen que estudiar 128 horas cada semestre (16 semanas). El objetivo general de este curso es aprender e implementar estructuras de datos y estructuras de control, así como desarrollar el pensamiento algorítmico para resolver problemas computacionales. El lenguaje de programación del curso es C++.

Cada semestre, aproximadamente 50 estudiantes toman el curso, en el semestre ll del año 2018, 40 estudiantes se matricularon al curso CS1, los cuales fueron seleccionados para el experimento presentado en este artículo. Los 40 estudiantes fueron divididos en dos grupos de forma aleatoria: 20 estudiantes de control (Grupo Control = GC) y 20 estudiantes de experimento (Grupo Experimental = GE).

Estrategia de enseñanza basa en la colaboración y la evaluación automática de código fuente

Para el desarrollo de la estrategia, se realiza un diseño de talleres de programación, una actividad colaborativa cíclica que permita que los estudiantes despierten habilidades en programación y mejoren sus calificaciones, además de sus relaciones personales. Y la selección de una herramienta de evaluación automática de código fuente, que se integre a las fases actividad colaborativa. A través de fases se ha propuesto la integración entre la colaboración y la herramienta de evaluación automática, como se muestra en la Figura 1, inicialmente en la fase 1 el profesor propone una tarea de programación, que debe contar con un enunciado, casos de entrada y casos de salida. En la fase 2, el estudiante debe abstraer el contenido del enunciado para dar una solución al problema, verificando que los casos de entrada sean acordes a los de salida, en esta fase los estudiantes pueden realizar varios envíos a través de la herramienta y esta evaluara e informara si están bien el código fuente o si tienen fallos de sintaxis o semántica. En la fase 3 y 4, se divide en los que ya terminaron la tarea y los que no; los que ya terminaron y aprobaron los casos de salida, ayudaran a sus compañeros que aún no han terminado la tarea.

Figura 1. Interacción de las fases de la actividad colaborativa apoyada de la herramienta de evaluación de código



Fuente: Elaboración propia

A. Actividad Colaborativa

En el curso CS1, uno de los problemas es lograr que los estudiantes exploten sus habilidades de programación y que mejoren sus calificaciones académicas. Estas son tareas difíciles que requieren proponer estrategias innovadoras que integren habilidades y mejoren las relaciones personales. Se diseña una actividad colaborativa basada en 4 fases cíclicas (Tabla I), que permiten generar conocimiento, facilitar el aprendizaje y promover actitudes autodidactas en los estudiantes. Las fases se repiten hasta que se cumpla el tiempo final de la tarea de programación.

Tabla 1. Actividad Colaborativa para una tarea de programación

Fases	Descripción
Fase 1. Asignación	Se elabora un taller de programación y se diseñan los casos de entradas y salidas.
Fase 2. Solución	Los estudiantes que logran terminar el taller antes del tiempo límite.
Fase 3. Agrupación	Agrupación, cada estudiante que terminó el taller, se agrupan con un estudiante que aún no han terminado.
Fase 4. Interacción	El estudiante que termino debe ayudar de forma verbal a resolver el taller al estudiante que no ha terminado (según los grupos).

Fuente: Elaboración propia

B. Herramienta de evaluación automática de código

Los estudios muestran que uno de los problemas en la enseñanza y aprendizaje de la programación está relacionado con la forma en que se evalúa el código fuente, teniendo en cuenta que cada código de un estudiante es diferente [26,27]. En este sentido, es necesario contar con herramientas automáticas de evaluación de códigos que apoyen al profesor y permitan al alumno comprender si tienen errores de código a nivel sintáctico o semántico [28,29].

En la forma tradicional de evaluación, donde el docente revisa el código fuente de cada estudiante, y toma un tiempo prudente para realizar realimentación de cada código fuente para determinar si es correcta la solución o identificar las fallas. Además, revisar varios códigos fuente por cada estudiante, requiere de tiempo con el que no se cuenta. Esto impide que los estudiantes no tengan la posibilidad de identificar sus errores y esto afecta el proceso de aprendizaje. Sin embargo, en la revisión de herramientas que permitan evaluar automáticamente código fuente, encontramos algunas que se mencionan en la Sección 1-A (Trabajos relacionados). Entre las herramientas se encuentra UNCODE [25] que cuenta con una interfaz apropiada para administrar cursos de programación de computadoras. Tiene un juez virtual que permite a los estudiantes presentar soluciones de código fuente, evaluar y enviar comentarios automáticamente en tiempo real. La evaluación automática del código se basa en un método de comparación entre casos de prueba (entradas) y casos de prueba esperados (salidas). Además, la plataforma permite guardar registros sobre las calificaciones obtenidas, el número de intentos de respuesta, los tiempos promedio, el puntaje de calificación y el indicador de estado para cada tarea. En este sentido, para este trabajo se la utiliza la herramienta UNCode, para integrarla a la actividad colaborativa.

C. Experimentos

Se plantearon dos experimentos, cada uno con 2 talleres de programación (Tabla ll), el taller 1 tiene una dificultad media y el taller 2 tiene un nivel de dificultad un poco más alto que el taller 1. En ambos talleres se incluye contenido del curso, acerca de: clases, objetos, métodos, estructuras de iteración y ciclos. Cada experimento se lleva a cabo en un aula por grupo, la cual cuenta con equipos de cómputo con los requisitos necesarios para que el estudiante realice los talleres.

Obtención de tiempos: Para el taller 1 se obtienen los datos desde la plataforma Moodle. Para el taller 2, se obtienen los datos desde la plataforma UNCode.

Tabla 2. Experimentos para el grupo control (GC) y el grupo experimental (GE)

Experimento	Talleres	GC	CE
1	Taller 1	Tradicional	Tradicional
2	Taller 2	Con herramienta de evaluación	Con herramienta de evaluación y actividad colaborativa

Fuente: Elaboración propia

1. Experimento 1

Ambos grupos, GC y GE, tiene un tiempo límite de 2 horas para resolver el taller de forma individual; cada uno con un asiento y un computador (tradicional). Al terminar el taller, debe enviar el código, en un archivo comprimido a través del campus virtual Moodle.

2. Experimento 2

Ambos grupos, GC y GE, tiene un tiempo límite de 2 horas para resolver el taller de forma individual; cada uno con un asiento y un computador. A diferencia del Taller 1, en este taller la entrega se realiza a través de la herramienta UNCode, en ella se validan los casos de entrada y salidas (previamente conocidas por los estudiantes). Sin embargo, para el GE, se utiliza la actividad colaborativa.

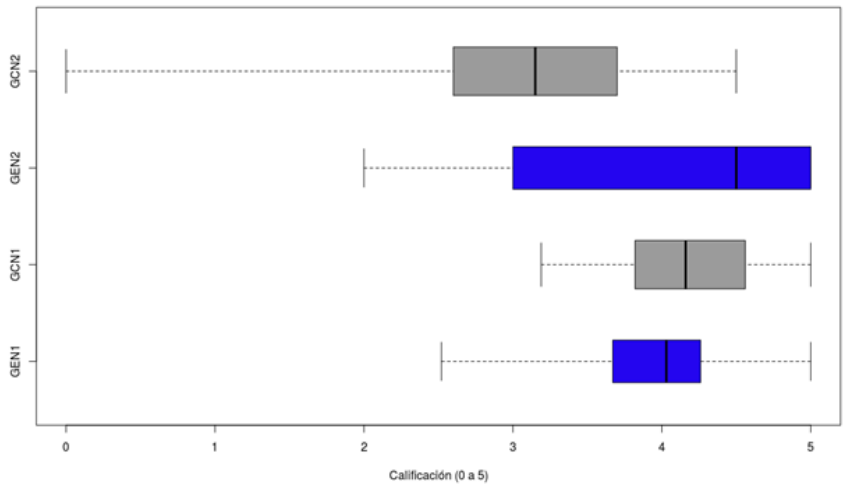
Resultados

En el experimento con el taller 1, los estudiantes entregaron sus soluciones de código a través de Moodle. Con respecto a las calificaciones en el taller 1 (N1), se obtuvo que el GC tiene un promedio de nota de 4.3 (N1) y el GE, mientras que el GE tiene un promedio de nota de 4.1 (N1), como se puede observar en la Figura 2. Los resultados son similares, los rangos se encuentran entre 3.7 y 4.7, con algunos casos de calificaciones no aprobadas con 2.0 y otros pocos casos obteniendo la máxima nota de 5.0.

En el taller 2 (N2), los estudiantes entregaron sus soluciones de código a través de la plataforma UNCode, y únicamente el GE usó la actividad de colaboración. La Figura 2, muestra que el GC tiene un promedio de nota de 3.3 (N1), y el GE tiene un promedio de nota de 4.6 (N1). Se puede observar en la figura que el caso de GC hay notas en el mínimo de 0.2 y en el máximo 3.6, mientras que, en el GE, la nota mínima fue de 2.0 y la máxima de 5.0.

En respuesta a la pregunta de interés ¿cuánto se incrementa la calificación promedio de una tarea de programación, utilizando una estrategia colaborativa apoyada por una herramienta de evaluación automática de código? En promedio, en comparación de GEN2 y GEN1, aumentaron cerca de 2.3%, 4 estudiantes mejoraron su nota. En comparación de GEN2 y GCN2, hay un incremento de 4.1%, 8 estudiantes mejoraron la nota por encima del otro grupo.

Figura 2. N1) Calificación obtenida en el taller 1, por cada grupo. N2) Calificación obtenida en el taller 2, por cada grupo



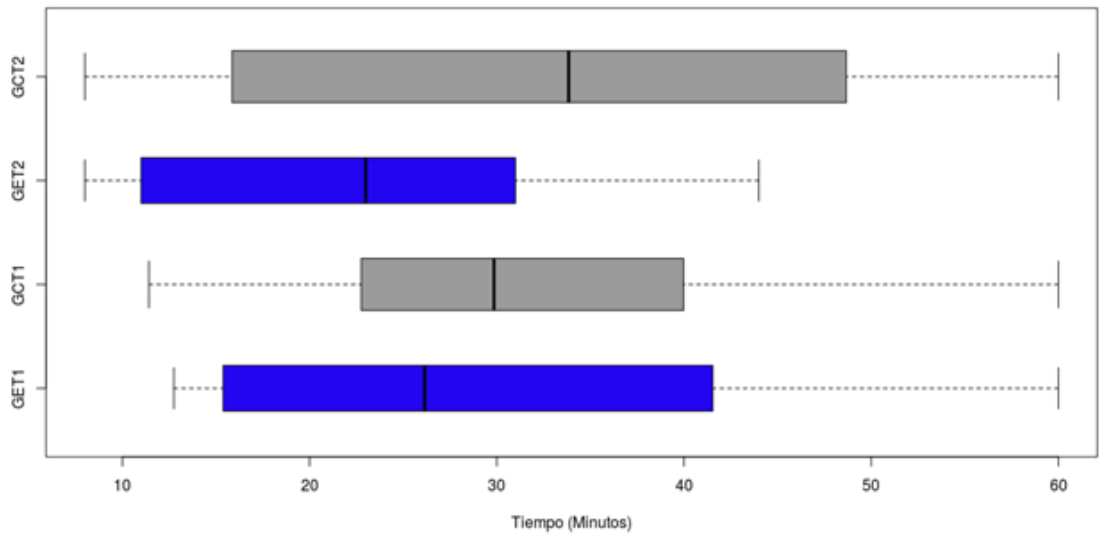
Fuente: Elaboración propia

Por otro lado, en cuanto a los tiempos de entrega de cada grupo para el taller 1, se observa en la Figura 3, que en promedio de solución para el GC fue de 30 minutos (T1). Es similar al del GE, que su promedio es de 27 minutos (T1), en este caso el rango general se encuentra entre 16 minutos y 42 minutos. Ambos grupos les toma la mitad del tiempo límite de la actividad para terminar.

Los tiempos de entrega del Taller 2, donde los estudiantes entregaron sus soluciones de código a través de la plataforma UNCode, y el GE uso la actividad de colaboración, se presentan. En la Figura 3, se muestra una gran diferencia entre los grupos, mientras que GE, 19 estudiantes terminan antes de 33 minutos (T2), en el GC, en ese mismo tiempo (33 minutos) cerca de 10 estudiantes han terminado.

En respuesta a la pregunta ¿Cuánto se incrementa en promedio la calificación de una tarea de programación, utilizando una estrategia colaborativa apoyada por una herramienta de evaluación automática de código? Según los resultados obtenidos, es claro que el uso de la estrategia permite que los estudiantes terminen casi en la tercera parte del tiempo límite.

Figura 3. T1) Tiempo total de entrega utilizado para resolver el taller 1, por cada grupo. T2) Tiempo total de entrega utilizado para resolver el taller 2, por cada grupo.



Fuente: Elaboración propia

Finalmente, los resultados del Taller 2, muestran que, de 20 estudiantes, sin utilizar la colaboración, solo 10 lograrían una calificación mayor a 3.0. Mientras que usando la colaboración 20 de estudiantes logran una calificación mayor a 3.0. Esto significa que el uso de la colaboración incrementa en un 50% la calificación de los estudiantes.

Discusión

Este trabajo presenta una estrategia para estimular las habilidades en la programación y la mejora de la calificación académica de estudiantes del curso CS1, a través de la herramienta UNCode, para automatizar la calificación de una tarea de programación. Al usar esta herramienta, fue posible probar y evaluar la estrategia propuesta, permitiendo realizar seguimiento sobre el proceso de aprendizaje, identificando si aplican los conceptos de programación en el código fuente, errores en el código fuente y obtener información para desarrollar estadísticas descriptivas sobre la calificación del grupo y el tiempo de entregas por cada taller de programación.

Por otro lado, el uso de la estrategia colaborativa apoyada con la herramienta UNCode, integra la evaluación formativa y sumativa. Por un lado, la evaluación automática permite identificar identifica el nivel alcanzado en el logro de aprendizaje de una actividad y, por otro lado, los resultados permiten al profesor tomar decisiones oportunas que beneficien el aprendizaje de los estudiantes. Sin embargo, es necesario contar a futuro con el diseño de una rúbrica que permita identificar claramente los resultados esperados.

El trabajo colaborativo entre estudiantes surge como apoyo para conseguir un resultado que resulta más difícil de conseguir individualmente. Además, el trabajo colaborativo permite que haya mayor homogeneidad de los estudiantes en los logros de aprendizaje del curso. La estrategia planteada, centra el aprendizaje del estudiante a través de la experiencia de colaboración y reflexión individual, mejorando habilidades sociales de comunicación, liderazgo, confianza en los demás, capacidad de intercambio de roles, entre otros. Esto transforma el espacio académico tradicional, por un espacio lúdico y divertido.

En el aprendizaje de la programación, y especialmente en el curso CS1, es necesario que el estudiante sea constante en la práctica de la programación para lograr alcanzar las competencias necesarias del curso. Para esto la academia cuenta con profesores idóneos que enseñan las bases necesarias, sin embargo, el tiempo limitado de interacción entre el profesor y el estudiante, no permite que el estudiante reciba retroalimentación permanente sobre sus procesos de aprendizaje. En este sentido, contar con herramientas que permitan evaluar un código fuente y dar realimentación inmediata, es un insumo muy importante para la enseñanza de la programación. Esto logra acelerar el proceso de aprendizaje, facilitar la adquisición de competencias académicas e incentivar actitudes de aprendizaje autónomo en los estudiantes.

Conclusiones

El diseño de estrategias que integren la colaboración y herramientas tecnológicas que impacten en el proceso de aprendizaje de programación, mejoran significativamente las calificaciones del estudiante, además de las habilidades interpersonales que incentivan a mejorar los cursos de programación.

En el desarrollo de una estrategia de aprendizaje basada en la colaboración y la evaluación de código, es importante que haya iteraciones que permitan observar la relación entre estudiante-herramienta, estudiante-estudiante para crear nuevas estrategias de enseñanza y tomar decisiones oportunas que beneficien al estudiante.

Cuando el estudiante se encuentra con problemas de programación retadores, pocas veces todos los resuelven. La estrategia colaborativa apoyada por una herramienta de evaluación automática de código, permiten integrar a los estudiantes para encontrar una solución colaborativa, además de fomentar el aprendizaje y el apoyo mutuo.

Referencias Bibliograficas

1. J. Figueiredo y F. J. García-Peñalvo, "Building skills in introductory programming", *ACM International Conference Proceeding Series, 2018*, pp. 46–50, DOI: https://doi.org/10.1145/3284179.3284190.
2. C. Alvarado, C. B. Lee, y G. Gillespie, "New CS1 pedagogies and curriculum, the same success factors?", *SIGCSE 2014 - Proceedings of the 45th ACM Technical Symposium on Computer Science Education,* New York, New York, USA, 2014, pp. 379–384, DOI: https://doi.org/10.1145/2538862.2538897.
3. «Computer Science Curricula 2013 Curriculum Guidelines for Undergraduate Degree Programs in Computer Science The Joint Task Force on Computing Curricula Association for Computing Machinery (ACM) IEEE Computer Society A Cooperative Project of», 2013, [En línea]. Disponible en: https://www.acm.org/binaries/content/assets/education/cs2013_web_final.pdf.
4. T. Lauwers y I. Nourbakhsh, "Informing Curricular Design by Surveying CS1 Educators", *Computing,* 2015, [En línea]. Disponible en: https://www.cs.cmu.edu/ illah/PAPERS/amire08.pdf.
5. J. Figueiredo, N. Lopes, y F. J. García-Pẽalvo, "Predicting student failure in an introductory programming course with multiple back-propagation", *presentado en ACM International Conference Proceeding Series,* 2019, pp. 44-49, DOI: https://doi.org/10.1145/3362789.3362925.
6. R. Kraleva, V. Kralev, y D. Kostadinova, "A methodology for the analysis of block-based programming languages appropriate for children", *Journal of Computing Science and Engineering,* vol. 13, n.o 1, pp. 1-10, 2019, DOI: https://doi.org/10.5626/JCSE.2019.13.1.1.
7. V. T. Nguyen, R. Hite, y T. Dang, "Web-based virtual reality development in classroom: From learner's perspectives", presentado en Proceedings - 2018 IEEE International Conference on Artificial Intelligence and Virtual Reality, AIVR 2018, 2019, pp. 11-18, DOI: https://doi.org/10.1109/AIVR.2018.00010.
8. S. I. Malik, "Assessing the teaching and learning process of an introductory programming course with bloom's taxonomy and Assurance of Learning (AOL)", *International Journal of Information and Communication Technology Education,* vol. 15, n.o 2, pp. 130-145, 2019, DOI: https://doi.org/10.4018/IJICTE.2019040108.
9. Q. Sun, J. Wu, y K. Liu, "How are students' programming skills developed: an empirical study in an object-oriented course", *Proceedings of the ACM Turing Celebration Conference - China*, Chengdu, China, may 2019, pp. 1–6, DOI: https://doi.org/10.1145/3321408.3322858.
10. A. Gonzalez-Escribano, V. Lara-Mongil, E. Rodriguez-Gutiez, y Y. Torres, "Toward improving collaborative behaviour during competitive programming assignments", presentado en Proceedings of EduHPC 2019: Workshop on Education for High Performance Computing - Held in conjunction with SC 2019: The International Conference for High Performance Computing, Networking, Storage and Analysis, 2019, pp. 68-74, DOI: https://doi.org/10.1109/EduHPC49559.2019.00014.
11. G. Wang, H. Zhao, Y. Guo, y M. Li, "Integration of flipped classroom and problem based learning model and its implementation in university programming course", presentado en 14th International Conference on Computer Science and Education, ICCSE 2019, 2019, pp. 606-610, DOI: https://doi.org/10.1109/ICCSE.2019.8845525.
12. J. R. Uhlar y S. Secules, "Butting heads: Competition and posturing in a paired programming team", presentado en Proceedings - Frontiers in Education Conference, FIE, 2019, vol. 2018-October, DOI: https://doi.org/10.1109/FIE.2018.8658654.
13. M. M. Rahman y R. Paudel, "Visual programming and interactive learning based dynamic instructional approaches to teach an introductory programming course", presentado en Proceedings - Frontiers in Education Conference, FIE, 2019, vol. 2018-October, DOI: 10.1109/FIE.2018.8658581.

14. S. Rasipuram y D. B. Jayagopi, *"Automatic assessment of communication skill in interview-based interactions", Multimedia Tools and Applications*, vol. 77, n.o 14, pp. 18709-18739, 2018, DOI: https://doi.org/10.1007/s11042-018-5654-9.

15. D. Insa y J. Silva, "Automatic assessment of Java code", *Computer Languages, Systems and Structures*, vol. 53, pp. 59-72, 2018, doi: https://doi.org/10.1016/j.cl.2018.01.004.

16. K. M. Ala-Mutka, "A Survey of Automated Assessment Approaches for Programming Assignments", *Computer Science Education*, vol. 15, n.o 2, pp. 83-102, jun. 2005, DOI: https://doi.org/10.1080/08993400500150747.

17. P. Navrat y J. Tvarozek, "Online programming exercises for summative assessment in university courses", presentado en ACM International Conference Proceeding Series, 2014, vol. 883, pp. 341-348, doi: https://doi.org/10.1145/2659532.2659628.

18. C. Douce, D. Livingstone, y J. Orwell, "Automatic test-based assessment of programming: A review", *J. Educ. Resour. Comput*., vol. 5, n.o 3, pp. 4–es, sep. 2005, DOI: https://doi.org/10.1145/1163405.1163409.

19. «INGInious' documentation — INGInious 0.6.dev0 documentation». https://inginious.readthedocs.io/en/v0.6/ (accedido nov. 25, 2019).

20. Y. huei Wang y H. C. Liao, "Learning performance enhancement using computer-assisted language learning by collaborative learning groups", *Symmetry*, vol. 9, n.o 8, p. 141, ago. 2017, DOI: https://doi.org/10.3390/sym9080141.

21. D. D. Pratt, "The making of CourseMaker, a web-based shell program which can be set up by the teacher to run online courses", 2003, Accedido: feb. 14, 2020. [En línea]. Disponible en: http://openscholar.dut.ac.za/handle/10321/243.

22. L. Malmi, V. Karavirta, A. Korhonen., y J. Nikander, Experiences on automatically assessed algorithm simulation exercises with different resubmission policies, J*ournal on Educational Resources in Computing (JERIC).* DOI https://dl.acm.org/doi/10.1145/1163405.1163412

23. L.-K. Soh, N. Khandaker, X. Liu, y H. Jiang, "I-MINDS: A multiagent system for intelligent computer-supported collaborative learning and classroom management", *International Journal of Artificial Intelligence in Education,* vol. 18, n.o 2, pp. 119–151, 2008.

24. M. T. Pham y T. B. Nguyen, "The DOMJudge Based Online Judge System with Plagiarism Detection", *2019 IEEE-RIVF International Conference on Computing and Communication Technologies (RIVF),* mar. 2019, pp. 1-6, DOI: https://doi.org/10.1109/RIVF.2019.8713763.

25. F. Restrepo-Calle, J. J. Ramírez-Echeverry, y F. A. Gonzalez, "Uncode: Interactive System for Learning and Automatic Evaluation of Computer Programming Skills", *EDULEARN18 Proceedings*, 2018, vol. 1, pp. 6888–6898, DOI: https://doi.org/10.21125/edulearn.2018.1632.

26. J. P. Ignizio, "Generalized goal programming An overview", *Computers & Operations Research*, vol. 10, n.o 4, pp. 277-289, ene. 1983, DOI: https://doi.org/10.1016/0305-0548(83)90003-5.

27. B. Dunne, A. J. Blauch, y A. Sterian, "The Case For Computer Programming Instruction For All Engineering Disciplines", 2005.

28. E. Stankov, M. Jovanov, B. Kostadinov, y A. Madevska Bogdanova, "A new model for collaborative learning of programming using source code similarity detection", *IEEE Global Engineering Education Conference*, EDUCON, 2015, vol. 2015-April, pp. 709-715, DOI: https://doi.org/10.1109/EDUCON.2015.7096047.

29. M. M. Rahman, Y. Watanobe, y K. Nakamura, "Source Code assessment and classification based on estimated error probability using attentive lstm language model and its application in programming education", *Applied Sciences (Switzerland),* vol. 10, n.o 8, 2020, DOI: https://doi.org/10.3390/APP10082973.