Como citar este artículo: C. G. Hidalgo-Suarez, V. A. Bucheli and H. Ordoñez, "Automatic Assessment of Learning Outcomes as a New Paradigm in Teaching a Programming Course: Engineering in Society 5.0," in IEEE Revista Iberoamericana de Tecnologias del Aprendizaje, vol. 17, no. 4, pp. 379-385, Nov. 2022, doi: 10.1109/RITA.2022.3217193.

Evaluación Automática de Resultados de Aprendizaje como un Nuevo Paradigma en la enseñanza de un Curso de Programación: La ingeniería en la sociedad 5.0

Carlos G Hidalgo-Suarez, Victor A Bucheli, Hugo Ordoñez

***Abstract*—Programming education goes through the transition from the content model to the learning outcomes model and the integration of technology, where automatic assessment tools allow students to support them in practice; This makes learning inclusive by proposing a transversal approach so that the necessary programming skills are achieved. In this sense, this paper presents a strategy that evaluates the source code and analyzes it using software metrics to identify students' learning results in a programming course. A strategy was developed that integrates an automatic source code evaluation tool, which allowed us to identify How an evaluation-based approach supports the learning process, time, and impact in a computer programming course? The results show that the strategy helps the transition from a programming course to a learning outcomes model and reduces the evaluation time without affecting students' grades, compared to the traditional way. Finally, it is essential to highlight that the development of strategies that integrate tools that support the teaching-learning and evaluation process in programming courses has a positive impact on academic training and decision-making, seeking to improve students' weaknesses through the analysis of the outcomes obtained.**

***Index Terms*— Code Metrics, Automatic Assessment Code, Programming Course, Education, Educational technology.**

I. Introduction

La inclusión del modelo por competencias y resultados de aprendizaje en la instituciones académicas de nivel superior y la integración en la sociedad 5.0, dónde tecnologías convergentes apoyan al estudiante, de manera responsable e inclusiva; además de permitir identificar si los estudiantes logran los conocimientos, habilidades, disposiciones y conductas necesarias, a partir de actividades propuestas en el curso [1], [2]. Esta innovación busca que el would estudiante domine todos los conceptos usando contenido/material personalizado, a diferencia del tradicional, donde los estudiantes dominaban algunos conceptos y el contenido/material es generalizado. Así mismo, buscando fomentar y mantener las conexiones entre el estudiante y la tecnología para facilitar el aprendizaje continuo, se hace importante fomentar el aprendizaje para la era digital buscando dar soluciones a la educación abierta y a distancia a través de la teoría del conectivismo [3], [4].

Los estándares de un curso de programación basado en resultados de aprendizaje han sido mediados por ACM/IEEE-CS Computer Science Curricula, las cuales dan recomendación de incluir la tecnología como fundamento en la enseñanza y el aprendizaje de los futuros profesionales, tecnólogos, desarrolladores. Según [5] evaluar resultados de aprendizaje se compone de 3 requisitos: 1) definir el diseño (micro-curricular) de los cursos basado en Sensibilidades, Capacidades y Competencias (SCC). 2) definir un esquema de evaluación, que integre la evaluación cuantitativa y la evaluación de los resultados y logros de aprendizaje de forma sincrónica. 3) monitorear y analizar el desarrollo de indicadores de logros de los estudiantes a través de herramientas tecnológicas que permitan identificar patrones que intervengan en la toma de decisiones.

Diseñar un curso basado en resultados de aprendizaje en el cual el fundamento base sea la teoría, es un tanto más fácil para evaluar al estudiante por medio de actividades; cuando se trata de evaluar un curso de programación, incluso las mismas tareas de los estudiantes, aparecen características que deben ser evaluadas manualmente, esto puede generar sesgos de precisión tomando en cuenta que se encuentran relacionadas con la formulación de problemas, la organización y el análisis lógico de datos, la representación de datos a través de abstracciones, el pensamiento algorítmico, la generalización de soluciones [6], entre otros elementos que pueden ser medidos a partir del código fuente [7]–[9]; dando una nueva forma de realimentación que permite identificar específicamente los indicadores de logro, debilidades y fortalezas del estudiante, y tomar decisiones a tiempo para mejorar el proceso académico formativo.

A pesar del trabajo de investigación en esta área, no existe consenso sobre cómo desarrollar estas habilidades en los

Carlos G Hidalgo-Suarez El autor pertenece a la Universidad del Valle, Cali, Colombia, teléfono: 317-367-2893; correo electrónico: carlos.hidalgo@correounivalle.edu.co.

Victor A Bucheli Autor de la Universidad del Valle, Cali, Colombia, es profesor del Departamento de Ciencias de la Computación, e-mail: victor.bucheli@correounivalle.edu.co.

Hugo Ordoñez Autor de la Universidad del Cauca, Popayan, Colombia, es profesor del Departamento de Ciencias de la Computación, e-mail: hugoordonez@unicauca.edu.co.

estudiantes [6] y tampoco se encuentra una estrategia de evaluación que sea genérica, ya que cada curso conlleva sus propias dificultades [10]. Sin embargo, en la última década las tecnologías de apoyo al aprendizaje y la enseñanza en la programación integran teorías de aprendizaje y técnicas computacionales que en conjunto mejoran los procesos formativos y la evaluación en los cursos de programación informática. Esto ha generado que se desarrollen herramientas que permitan evaluar y analizar código fuente usando algoritmos de Inteligencia Artificial y análisis sintáctico de código a través de métricas de software [11], [12], además, de impulsar el análisis, síntesis, evaluación, comunicación, intercambio de conocimientos y pensamiento crítico entre los estudiantes [13], y reducción de tiempo de evaluación; todo esto propone que los estudiantes aprendan en un contexto social mediado por la tecnología [4], [14].

En este artículo se presenta una estrategia que apoya la evaluación en el aprendizaje (resultados de aprendizaje) de un curso de programación funcional a través de una herramienta de evaluación de código fuente, que integra las métricas de Halstead para identificar el cálculo de palabras clave (reservadas) y variables escritas en el código fuente de un estudiante [15], [16]. La estrategia se implementó usando las entregas de 30 actividades/talleres de 64 estudiantes en un curso (control y experimental) de programación funcional. Se contrastó el impacto de la estrategia propuesta, mostrando resultados de una evaluación tradicional y una evaluación automática usando métricas de software. Los resultados muestran que se mejoró la evaluación, en cuanto a tiempo usado por el profesor para calificar precisión y realimentación de las actividades de aprendizaje (talleres de programación, exámenes etc).

En particular, se examinaron algunos aspectos sobre la relación entre los estudiantes, la estrategia de evaluación y la herramienta de evaluación automática en términos de interacción. Por lo tanto, se realizó un estudio exploratorio cuantitativo con el fin de dar respuesta a ¿Cómo una estrategia basada en evaluación de resultados de aprendizaje apoya el proceso de aprendizaje, el tiempo y el impacto en un curso de programación informática? Algunos trabajos, han propuesto hipótesis similares, basadas en estrategias de evaluación para los cursos de programación; se destacan tres trabajos [17]–[19] los cuales demuestran cierta mejora del uso de un aprendizaje activo asistido por la tecnología en contraste con el aprendizaje tradicional.

Este artículo se compone de seis secciones. I. Introducción. Se presenta el contexto del trabajo. II. Trabajos relacionados. Se presentan algunos trabajos similares junto con descripciones y características de las herramientas de software. III. Método. Se describe en detalle las preguntas de interés, la selección de la muestra del curso de programación y las fases del experimento, diseño de los resultados de aprendizaje a evaluar y la integración de una herramienta de evaluación de código fuente y métricas. IV. Resultados. Aplicación de los experimentos sobre el grupo experimental y el grupo control. V. Discusión y trabajo futuro. Se expone el análisis y algunas recomendaciones para continuar el trabajo. VI. Conclusiones. Finalmente se resaltan los aportes y resultados más relevantes de esta investigación.

II. Trabajos relacionados

En la actualidad los esfuerzos por apoyar el aprendizaje y la enseñanza de la programación usando tecnologías innovadoras se centran en los aspectos cognitivos del aprendizaje y el alumno [20], [21], y otros desarrollan herramientas de evaluación automática basadas en enfoques de aprendizaje [11], [12], [22]–[24]. Herramientas que utilizan mecanismos para ejecutar y compilar tareas de programación, sometiendo a diferentes casos de prueba que apoyan al estudiante a trabajar en modo prueba y error, reduciendo la realimentación por cada vez que lo intenta. Esto se complementa con herramientas que realizan una prueba estática, que consiste en verificar el estilo y aplicación de las métricas en la estructura sintáctica de los programas fuente o comparar la similitud del código con los programas previamente construidos. Algunas herramientas se describen a continuación y la Tabla I muestra las características más importantes de cada herramienta.

I-MINDS [25] es un software para la gestión inteligente de aulas o grupos online, permite revisar las actividades de programación en tiempo real y offline, facilitando la práctica del alumno. 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 [26], permite a los estudiantes presentar sus soluciones a los problemas de programación planteados y evaluarlos de forma inmediata.

Tabla I
Comparación de herramientas del tipo de evaluación de código fuente

Herramienta	Análisis sintáctico de código fuente	Métricas de software
I-MINDS	Si	Halstead
TRY	Si	CHOI
Course Maker	Si	No
TRAKLA	Si	No
INGInious M-IDEA	Si	Halstead

La plataforma INGInious M-IDEA [11], la cual integra diversas estrategias para los procesos académicos: formación de grupos, evaluación formativa y la realimentación de contenido y código fuente. Todo esto como apoyo al diseño de actividades de aprendizaje de los cursos de programación, específicamente en el curso CS1. Además, el entorno tiene diferentes utilidades para realizar comentarios sumativos a través de veredictos del programa, que indican si la sintaxis del código fuente son correctas o presentan algún tipo de error. Con base en estos veredictos, se asigna una calificación a las soluciones propuestas por los estudiantes.

El sistema TRY (Reek) [27], proporciona realimentación inmediata sobre un programa, el programa se compara con un símbolo de los resultados generados con los resultados esperados. TRY permite varios intentos de respuesta, pero los restringe a un número máximo para obligar al estudiante a pensar detenidamente sobre la lógica semántica y el

funcionamiento de su programa antes de enviar su respuesta a la evaluación.

TRAKLA [28] es una herramienta de evaluación automática aplicada a la simulación gráfica de algoritmos. El estudiante simula un algoritmo arrastrando y soltando una representación gráfica de estructuras de datos. Esta herramienta permite trabajar en grupo, pero no permite evaluar el código fuente.

CourseMaker [29] está desarrollado en Java, utiliza patrones y herramientas de diseño orientado a objetos para obtener modularidad y flexibilidad. Realiza pruebas dinámicas de funcionalidad, eficiencia y pruebas estáticas que incluyen análisis de estilo y detección de plagio, solo funciona para programas en el lenguaje Java. Permite que el autor del problema determine el número de intentos y el nivel de realimentación. Este sistema registra las estadísticas de errores de compilación por alumno, este registro se utiliza posteriormente para determinar los contenidos del curso que deben reforzarse. posteriormente para determinar los contenidos del curso que deben reforzarse

Finalmente, después de la revisión de herramientas que permitan la evaluación y realimentación automática, la plataforma INGInious M-IDEA es la seleccionada para las pruebas en este artículo.

III. Método

En esta sección, se identifica: 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 del estudio en el curso de fundamentos de programación funcional (FDP) del plan de estudios del programa de Ingeniería de Sistemas de la Universidad del Valle, Cali-Colombia, el cual es uno de los cursos con mayor tasa de mortalidad y absentismo. Buscando soluciones oportunas, es importante integrar nuevas estrategias de evaluación basadas en el análisis y estudio de las diferentes variables que permiten a los estudiantes aprender los diferentes conceptos del curso, además, de integrar nuevas herramientas que soporten la evaluación de código fuente automáticamente, para medir cuáles son las falencias en el rendimiento académico. En este sentido, buscando implementar una estrategia y descubrir si es un aporte, se responde ¿Cómo es la distribución del período de tiempo entre la calificación tradicional y la calificación automática? ¿Cuál es el impacto/ventaja en el aprendizaje que podría proporcionar a los estudiantes usar la estrategia de evaluación en un curso de programación informática? ¿Cómo una estrategia basada en evaluación de código y métricas apoya el proceso de aprendizaje en un curso de programación informática?

B. Población y muestra

Cada semestre, aproximadamente 150 estudiantes toman el curso de FDP, en el semestre II del 2020, 148 estudiantes se matricularon en 2 cursos diferentes, curso 1 (64 estudiantes) y curso 2 (84 estudiantes); cada curso cuenta con un profesor y un monitor. Se selecciona el curso 1 para hacer parte del experimento. Con el total de entregas de las actividades de los estudiantes del curso 1 (1.920 entregas aproximadamente), se analizan los mismos datos como grupo control (C = forma tradicional) y como grupo experimental (E = uso de estrategia de evaluación).

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, durante 16 semanas. Para aprobar el curso los estudiantes deben presentar diferentes actividades que son evaluadas por el profesor de forma numérica (nota de 0.0 a 5.0), entre las que se encuentran talleres, actividades colaborativas, un exámen y un proyecto. Los contenidos del curso están basados en el paradigma funcional usando el lenguaje de programación JavaScript.

C. Estrategia de evaluación

Para el desarrollo de la estrategia, se realizó un diseño del curso FDP por resultados de aprendizaje, tomando como base la reforma curricular de la Facultad de Ingeniería de la Universidad del Valle [30]. El cual se compone de una estructura jerárquica de mayor a menor:

- Resultados de Aprendizaje (RA): Se dividen en uno o varios indicadores de logro IL, para lograr evidenciar el proceso de aprendizaje de los estudiantes en el curso.
- Indicador de logro (IL): Se miden en una o varias actividades de evaluación (AE) y se enseñan en una o varias actividades de formación (AF).
- Actividades de Evaluación (AE): Las AE y las AF son propuestas por el docente; las AE, permiten observar el cumplimiento o no de los IL y asignarles una nota.
- Actividades de Formación (AF): Toda AE se basa en algo que se puede medir, para tener evidencia del logro en el aprendizaje. Las AE típicas son los exámenes, los trabajos finales, los proyectos, entre otras. Esta reforma tiene como sentido de aprendizaje una base del conectivismo, donde el aprendizaje mediado por la tecnología es un apoyo a la forma en que el estudiante decide aprender [4].

En la Tabla II, se puede observar un ejemplo usando la estructura basada en la reforma curricular [30] utilizada para este trabajo.

TABLA II
ESTRUCTURA DEL DISEÑO DEL CURSO FDP POR RESULTADOS DE APRENDIZAJE

RA	IL	AE	AE
Usa y conoce los procesos de la sintaxis del lenguaje de programación JavaScript, para implementar los algoritmos que dan solución a problemas específicos.	Implementa un algoritmo que da solución a un problema que incluye funciones y composición de funciones.	Actividad donde debe desarrollar un programa para hacer la sumatoria recursiva.	Taller 1-Recursión.

D. Proceso

En el momento que el juez de la herramienta M-IDEA tenga un veredicto de cada estudiante, se activa el módulo MetricsV que extrae el código fuente y realiza 3 tareas:

1) Usando las métricas de complejidad del software Halstead, tomando como referencia [31], el cual distingue entre el número de operadores y operandos únicos y el número total de operadores y operandos que existen en cada código fuente, se extrae y se obtiene un informe métrico de JavaScript (ver Tabla III).

TABLA III
MEDIDAS/MÉTRICAS DE HALSTEAD

Symbol	Quantity	Conversion from Gaussian and CGS EMU to SI [a]
n	Vocabulario	n1+n2
N	Media	N1+N2
V	Volumen	Longitud * Log2 Vocabulario
D	Dificultad	(n1/2) * (N1/n2)
E	Esfuerzos	Dificultad * Volumen
B	Errores	Volumen / 3000
T	Tiempo de test	Tiempo = esfuerzos / S, donde S=18 segundos.

2) A partir del diseño del curso basado en competencias y con las métricas obtenidas del código fuente, MetricsV compara e identifica automáticamente si se cumplió ese logro de aprendizaje o no. En la Tabla IV, se muestra un ejemplo, donde se pide que realice un programa para calcular el área de un rectángulo, un estudiante hace su envío y con la métrica de vocabulario, se puede identificar que escribió una función de forma correcta, eso quiere decir que el indicador de logro que se mide fue aceptado; adicional a ello, no solo se verifica si se cumple con la implementación, sino que además, se verifica que el código sea correcto, para ello el juez evaluador, prueba con los casos de entrada, donde al donde al ejecutarlos se muestran las salidas (para este caso correctas).
3) 3) Finalmente, MetricsV realiza una rúbrica de evaluación, en la cual se asigna porcentajes a cada RA, en total son 4 (cada uno con 25%). Cada RA se divide en 7 IL, que cada uno equivale a 3,57% dentro de su RA. Cada AE y AF (actividades propuestas) apuntan a IL específicos, dependiendo de lo que el estudiante logre se verá reflejada su calificación en porcentaje alcanzado.

E. Experimento

Los 64 estudiantes del semestre II de 2020 (curso 1), entregaron cada uno 30 actividades (cada una con un peso de 0.166 en la nota total) en INGInious M-IDEA, entre las que se encuentra: 2 de tipos de datos, 4 de funciones, 6 de composición de funciones, 7 de condicionales, 5 de recursividad, 3 de datos compuestos, 2 de estructuras y árboles y 1 de abstracción, encapsulación y polimorfismo.

Las entregas totales de los estudiantes se estiman en 1.920, sin embargo, no todos los estudiantes realizaron las actividades, en este caso en total fueron 1.782 entregas. Con el total de entregas, se procede a realizar tres experimentos (ver Tabla V):

TABLA V
EXPERIMENTOS PARA EVALUAR LAS ENTREGAS DE CÓDIGO FUENTE DE ESTUDIANTES DE UN CURSO DE PROGRAMACIÓN

Experimento	Talleres/ estudiantes	Entregas	Evaluador	Identificar
1	30*64-2%	1782	Profesor	Tiempo, nota,
2	30*64-2%	1782	Monitor	Tiempo, nota
3	30*64-2%	1782	MetricsV	Tiempo, nota,

1) El profesor del curso califica de forma manual (tradicional) las entregas de los estudiantes y hace el promedio para dar una calificación final (se apoya de la herramienta LibreOffice Calc y usando un cronómetro digital toma el tiempo que tarda en calificar cada entrega). Adicionalmente, de forma manual también usa el diseño del curso basado en competencias para identificar los IL que el estudiante alcanzó en cada entrega (este paso no fue tomado en cuenta dentro del tiempo de calificación)
2) El monitor del curso califica de forma manual el código usando una rubrica analítica (tradicional) de las entregas de los estudiantes y hace el promedio para dar una calificación final (se apoya de la herramienta LibreOffice Calc y un cronómetro digital para tomar el tiempo que tarda en calificar el total de entregas).
3) Con el uso de la herramienta desarrollada para este artículo *MetricsV*, se realiza la calificación de forma automática y se identifica los IL alcanzados por el estudiante en cada entrega.

IV. RESLTADOS

Respondiendo a la pregunta de interés ¿Cómo es la distribución del período de tiempo entre la calificación tradicional y la calificación automática? Se realizó una comparación entre los tiempos que le toma al profesor, al monitor y a MetricsV hacer la revisión y evaluación de las entregas de las 30 actividades de los 64 estudiantes (ver Figura 2). Primero, el profesor del curso califica las entregas,

contando con la experiencia y experticia en los temas y usando una rubrica analítica que hace parte de la Escuela de Ingeniería y Computación (EISC) de la Universidad del Valle. Se contabilizó un tiempo promedio de 72.0 minutos por actividad (64 entregas), eso quiere decir que al profesor le tomó cerca de 2.160 horas en calificar. Segundo, el monitor, quien tiene un conocimiento un poco más avanzado que el de los estudiantes del curso, se contabilizó un tiempo promedio de 133.5 minutos por actividad (64 entregas), eso quiere decir que al monitor le tomó cerca de 3.990 horas en calificar. Finalmente, a la herramienta de evaluación MetricsV le tomó calificar automáticamente las entregas en un tiempo promedio de 0.475 segundos, eso quiere decir, que le tomó 14 minutos. Adicionalmente MetricsV evaluó los IL en 0.02 segundos, en total le tomó 56 segundos, muy por debajo de la evaluación manual del profesor y el monitor.

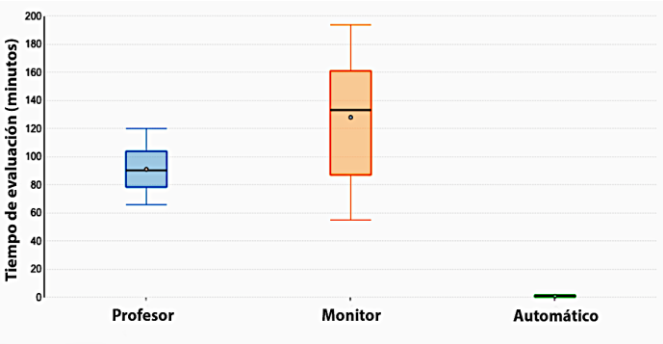

Figura 2. Comparación de tiempo de calificación de entregas de estudiantes entre profesor, monitor y la herramienta MetricsV.

Para validar la confiabilidad de los evaluadores (profesor, monitor y herramienta) se utilizó la medida kappa de Fleiss [32] una medida de acuerdo de evaluadores. Radica en el hecho de que representa hasta qué punto los datos recopilados en el estudio son representaciones correctas de las variables medidas. Basado en la interpretación de [33], nuestra Kappa=0,8818815 como el intervalo de confianza para el 95% (0,8735369-0,8902261), es concordancia muy buena, con Kappa estadísticamente significativo Z = 4,3726368.

El siguiente resultado, hace referencia a responder la pregunta de interés ¿Cuál es el impacto/ventaja en el aprendizaje que podría proporcionar a los estudiantes usar la estrategia de evaluación en un curso de programación informática? Después de que el profesor calificará las entregas, asigna una nota total del curso, usando el promedio (Calificación total por actividad / 30), dando como resultado que los estudiantes están en un promedio de 4.4 con algunos casos entre 3.2 y 3.7. Mientras que usando MetricsV el promedio es de 4.3 y se nota una distribución de notas uniforme entre 3.0 y 3.6. En la Figura 3, donde el grupo control, es la evaluación de forma manual por parte del profesor, y el grupo experimental, es la evaluación de forma automática usando MetricsV. Se observa un impacto positivo de la herramienta realizando evaluaciones de manera similar a la del profesor (media 4.3 y 4,4 respectivamente), además, evita los efectos halo y horn [34], evitando sesgos en las calificaciones del estudiante.

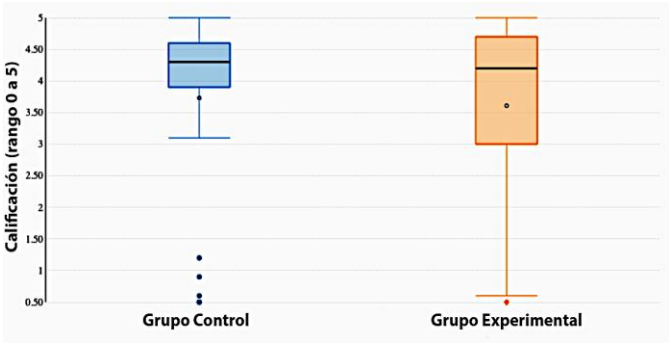

Figura 3. Comparación de notas totales entre la calificación del profesor y la calificación con la herramienta MetricsV

Finalmente, para responder a la pregunta de interés ¿Cómo una estrategia basada en evaluación de código y métricas apoya el proceso de aprendizaje en un curso de programación informática? Es importante mencionar que la estrategia es desarrollada con el propósito de apoyar al profesor, en ningún caso puede reemplazarlo. Sin embargo, tanto la estrategia como el módulo MetricsV, según como se evidenció en los anteriores resultados, es capaz de calificar de una forma similar a la del profesor y ahorrar tiempo en esta tarea. Además, permite identificar los logros de aprendizaje que se realizaron en el diseño del curso por resultados de aprendizaje, obteniendo de los 64 estudiantes lo siguiente:

1) Un 30% no implementa o implementa mal la composición de funciones en las actividades.
2) Los temas en el que menos logros de aprendizaje obtuvieron fueron: recursividad con 19 estudiantes que obtuvieron menos del 50% de los IL y condicionales con 1 estudiante que obtuvo menos del 50% de los IL.
3) En general a los estudiantes les fue bien, sin embargo 6 de ellos, no alcanzaron los resultados esperados, no llegaron al 50% de los IL y eso se vio reflejado en su nota numérica con calificación final entre 2.7 y 2.9.
4) El impacto porcentual puede que no sea significativo, ya que en las calificaciones no hay mucha varianza, sin embargo, el estudiante fue capaz de reconocer la herramienta como una forma de practica y un aprendizaje activo que puede tener cierto grado de significancia, pero por ahora nos quedamos con el logro de apoyar al profesor en avaluar y obtener datos para poder tomar decisiones.

V. Conclusiones

El desarrollo de estrategias que integren herramientas de evaluación automática para mejorar la enseñanza de la programación, puede que apoyen el aprendizaje del estudiante, pero es evidente que, si mejoran significativamente los tiempos de calificación del estudiante, además de identificar los logros de aprendizaje por una actividad especifica.

Las herramientas que poyan los cursos de programación más la evaluación automática, permiten a los estudiantes practicar y aprender de diferentes maneras, considerando tiempo y espacio ideal, donde el estudiante se sienta cómodo; esto logra que la programación no solo sea para ingeniería de sistemas, sino sea general y permita ser aplicada en diferentes

contextos.

La evaluación automática a diferencia de la forma manual/tradicional no tiene sesgo sobre las calificaciones, lee un programa y es evaluado según los parámetros que se le indiquen, además toma menos tiempo en evaluar, dando realimentación de inmediato y así el estudiante es capaz de identificar su debilidades y fortalezas

En el desarrollo de una estrategia basada en la evaluación de código, es importante que haya actividades dinámicas que integren diferentes temas vistos en clase, así permitan observar la relación entre estudiante-herramienta, estudiante-estudiante y conocimiento es general. Para crear nuevas estrategias de aprendizaje-enseñanza y tomar decisiones oportunas que beneficien al estudiante.

Cuando el estudiante no alcance los objetivos necesarios en el curso, es importante tener medidas alternas que permitan mejorar el proceso académico del estudiante.

AGRADECIMIENTOS

Queremos agradecer a la Fundación CEIBA-Nariño, por ser el patrocinador de los estudios doctorales de Carlos G Hidalgo. También agradecer la contribución de la Universidad del Cauca al profesor Hugo Ordoñez y la Universidad del Valle al profesor Víctor Bucheli, por el apoyo y el tiempo asignado para realizar esta investigación.

REFERENCES

[1] M. Panizzon y P. F. P. Barcellos, «Critical Success Factors of the University of the Future in a Society 5.0: A Maturity Model», *World Futur. Rev.*, vol. 12, n.º 4, pp. 410-426, dic. 2020, doi: 10.1177/1946756720976711.

[2] P. Sancho-Thomas, R. Fuentes-Fernández, y B. Fernández-Manjón, «Learning teamwork skills in university programming courses», *Comput. Educ.*, vol. 53, n.º 2, pp. 517-531, sep. 2009, doi: 10.1016/j.compedu.2009.03.010.

[3] M. Zapata-Ros, «Teorías y modelos sobre el aprendizaje en entornos conectados y ubicuos : bases para un nuevo modelo teórico a partir de una visión crítica del conectivismo = Theories and models about learning in connected and ubiquitous environments : bases for a new theoretical model from a critical vision of connectivism», *Teorías Model. Sobre El Aprendiz. En Entornos Conectados Ubicuos Bases Para Un Nuevo Modelo Teórico Partir Una Visión Crítica Conectivismo Theor. Models Learn. Connect. Ubiquitous Environ. Bases New Theor. Model Crit. Vis. Connect.*, pp. 69-102, 2015.

[4] «Connectivism and leadership: harnessing a learning theory for the digital age to redefine leadership in the twenty-first century - ScienceDirect». https://www.sciencedirect.com/science/article/pii/S2405844020300955 (accedido ago. 02, 2021).

[5] L. Carvajal-Ortiz, B. Florian-Gaviria, y J. F. Díaz, «Models, methods and software prototype to support the design, evaluation, and analysis in the curriculum management of competency-based for higher education», en *2019 XLV Latin American Computing Conference (CLEI)*, sep. 2019, pp. 1-10. doi: 10.1109/CLEI47609.2019.235114.

[6] T. Crow, A. Luxton-Reilly, y B. Wuensche, «Intelligent tutoring systems for programming education: a systematic review», en *Proceedings of the 20th Australasian Computing Education Conference*, New York, NY, USA, ene. 2018, pp. 53-62. doi: 10.1145/3160489.3160492.

[7] F. Y. Gamarra-Mendoza, «La aplicabilidad de la evaluación por competencias en los escenarios universitarios: The skills evaluation applicability in the university settings», *Prohominum*, vol. 2, n.º 1, Art. n.º 1, jul. 2020, doi: 10.47606/ACVEN/PH0003.

[8] N.-T. Le, S. Strickroth, S. Gross, y N. Pinkwart, «A Review of AI-Supported Tutoring Approaches for Learning Programming», en *Advanced Computational Methods for Knowledge Engineering*, Heidelberg, 2013, pp. 267-279. doi: 10.1007/978-3-319-00293-4_20.

[9] M. Gómez-Albarrán, «The Teaching and Learning of Programming: A Survey of Supporting Software Tools», *Comput. J.*, vol. 48, n.º 2, pp. 130-144, ene. 2005, doi: 10.1093/comjnl/bxh080.

[10] J. P. Barros, «Assessment and grading for CS1: Towards a complete toolbox of criteria and techniques», 2010, pp. 106-111. doi: 10.1145/1930464.1930483.

[11] Bucheli. V. Hidalgo C., «Modelo soportado en inteligencia artificial para el desarrollo de actividades de aprendizaje activo basadas en colaboración asistida por computador (M-IDEA) | Request PDF», presentado en 17th LACCEI International Multi-Conference for Engineering, Education, and Technology, 2019. doi: 10.18687/LACCEI2019.1.1.528.

[12] F. Restrepo-Calle, J. J. Ramírez-Echeverry, y F. A. Gonzalez, «Uncode: Interactive System for Learning and Automatic Evaluation of Computer Programming Skills», en *EDULEARN18 Proceedings*, 2018, vol. 1, pp. 6888-6898. doi: 10.21125/edulearn.2018.1632.

[13] D. Fonte, D. Cruz, A. L. Gançarski, y P. Henriques, «A Flexible Dynamic System for Automatic Grading of Programming Exercises», 2013. doi: 10.4230/OASIcs.SLATE.2013.129.

[14] Cheang Brenda, Kurnia Andy, Lim Andrew, y Oon Wee-Chong, «On automated grading of programming assignments in an academic institution | Computers & Education», 2003. https://dl.acm.org/doi/10.1016/S0360-1315%2803%2900030-7 (accedido abr. 07, 2021).

[15] T. Hariprasad, G. Vidhyagaran, K. Seenu, y C. Thirumalai, «Software complexity analysis using halstead metrics», en *2017 International Conference on Trends in Electronics and Informatics (ICEI)*, may 2017, pp. 1109-1113. doi: 10.1109/ICOEI.2017.8300883.

[16] S. A. Abdulkareem y A. J. Abboud, «Evaluating Python, C++, JavaScript and Java Programming Languages Based on Software Complexity Calculator (Halstead Metrics)», *IOP Conf. Ser. Mater. Sci. Eng.*, vol. 1076, n.º 1, p. 012046, feb. 2021, doi: 10.1088/1757-899X/1076/1/012046.

[17] X. Polanco, «Transformación de la información en conocimiento y del conocimiento en decisiones estratégicas», *Temas Actuales Indicadores Defic. Tecnol. En América Lat. El Caribe B. Aires RICYT Edicionestextless Httpwww Ricyt Edu ArinteriornormalizacionIVtallerpolanco PdftextgreaterConsulta 30 Septiembre 2009*, 2001.

[18] J. Figueiredo y F. J. García-Peñalvo, «Estrategias de enseñanza y aprendizaje de la programación en cursos universitarios», p. 4.

[19] M. O. G. Fernández y P. H. Gaytán, «Experiencia del aula invertida para promover estudiantes prosumidores del nivel superior», *RIED Rev. Iberoam. Educ. Distancia*, vol. 22, n.º 2, pp. 245-263.

[20] J. López Reguera, C. Hernández Rivas, y Y. Farran Leiva, «Una plataforma de evaluación automática con una metodología efectiva para la enseñanza/aprendizaje en programación de computadores», *Ingeniare Rev. Chil. Ing.*, vol. 19, n.º 2, pp. 265-277, ago. 2011, doi: 10.4067/S0718-33052011000200011.

[21] A. M. Tocci, «Estilos de aprendizaje de los alumnos de ingeniería según la programación neuro lingüística», *Rev. Estilos Aprendiz.*, vol. 6, n.º 12, Art. n.º 12, oct. 2013, Accedido: abr. 07, 2021. [En línea]. Disponible en: http://revistaestilosdeaprendizaje.com/article/view/994

[22] Hieke Keuning, Jeuring Johan, y Heeren Bastiaan, «Towards a Systematic Review of Automated Feedback Generation for Programming Exercises | Proceedings of the 2016 ACM Conference on Innovation and Technology in Computer Science Education», 2016. https://dl.acm.org/doi/10.1145/2899415.2899422 (accedido abr. 07, 2021).

[23] Ramadhan Haider y du Boulay Benedict, «Programming Environments for Novices | SpringerLink», 1993. https://link.springer.com/chapter/10.1007/978-3-662-11334-9_12 (accedido abr. 07, 2021).

[24] F. Restrepo-Calle, J. J. R. Echeverry, y F. A. González, «Continuous assessment in a computer programming course supported by a software tool», *Comput. Appl. Eng. Educ.*, vol. 27, n.º 1, pp. 80-89, 2019, doi: https://doi.org/10.1002/cae.22058.

[25] L.-K. Soh, N. Khandaker, X. Liu, y H. Jiang, «I-MINDS: A multiagent system for intelligent computer- supported collaborative learning and classroom management.», *Int. J. Artif. Intell. Educ.*, vol. 18, n.º 2, pp. 119-151, 2008.

[26] M. T. Pham y T. B. Nguyen, «The DOMJudge Based Online Judge System with Plagiarism Detection», en *2019 IEEE-RIVF International*

Conference on Computing and Communication Technologies (RIVF), mar. 2019, pp. 1-6. doi: 10.1109/RIVF.2019.8713763.

[27] K. A. Reek, «The TRY system -or- how to avoid testing student programs», en *Proceedings of the twentieth SIGCSE technical symposium on Computer science education*, New York, NY, USA, feb. 1989, pp. 112-116. doi: 10.1145/65293.71198.

[28] «WWW-TRAKLA». http://www.cs.hut.fi/~tred/WWW-TRAKLA/ (accedido sep. 29, 2020).

[29] 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

[30] Universidad del Valle, «Reforma Currícular - Facultad de Ingeniería / Universidad del Valle / Cali, Colombia», *Reforma curricular 2020*. http://ingenieria.univalle.edu.co/reforma-curricular (accedido abr. 08, 2021).

[31] S. A. Abdulkareem y A. J. Abboud, «Evaluating Python, C++, JavaScript and Java Programming Languages Based on Software Complexity Calculator (Halstead Metrics)», *IOP Conf. Ser. Mater. Sci. Eng.*, vol. 1076, n.º 1, p. 012046, feb. 2021, doi: 10.1088/1757-899X/1076/1/012046.

[32] R. Falotico and P. Quatto, "Fleiss' kappa statistic without paradoxes," Qual. Quant., vol. 49, no. 2, pp. 463–470, Mar. 2015, doi: 10.1007/s11135-014-0003-1.

[33] J. Cerda L and L. Villarroel Del P, "Evaluación de la concordancia inter-observador en investigación pediátrica: Coeficiente de Kappa," Rev. Chil. Pediatría, vol. 79, no. 1, Feb. 2008, doi: 10.4067/S0370-41062008000100008.

[34] M. Macdougall, S. C. Riley, H. S. Cameron, and B. Mckinstry, "halos and horns in the assessment of undergraduate medical students: a consistency-based approach" p. 13.

[35] G. O. Young, "Synthetic structure of industrial plastics (Book style with paper title and editor)," in *Plastics*, 2nd ed. vol. 3, J. Peters, Ed. New York: McGraw-Hill, 1964, pp. 15–64.

[36] W.-K. Chen, *Linear Networks and Systems* (Book style). Belmont, CA: Wadsworth, 1993, pp. 123–135.

[37] H. Poor, *An Introduction to Signal Detection and Estimation*. New York: Springer-Verlag, 1985, ch. 4.

[38] B. Smith, "An approach to graphs of linear forms (Unpublished work style)," unpublished.

[39] E. H. Miller, "A note on reflector arrays (Periodical style—Accepted for publication)," *IEEE Trans. Antennas Propagat.*, to be published.

[40] J. Wang, "Fundamentals of erbium-doped fiber amplifiers arrays (Periodical style—Submitted for publication)," *IEEE J. Quantum Electron.*, submitted for publication.

[41] C. J. Kaufman, Rocky Mountain Research Lab., Boulder, CO, private communication, May 1995.

[42] Y. Yorozu, M. Hirano, K. Oka, and Y. Tagawa, "Electron spectroscopy studies on magneto-optical media and plastic substrate interfaces" *IEEE Transl. J. Magn.Jpn.*, vol. 2, Aug. 1987, pp. 740–741 [*Dig. 9th Annu. Conf. Magnetics* Japan, 1982, p. 301].

[43] M. Young, *The Techincal Writers Handbook.* Mill Valley, CA: University Science, 1989.

[44] J. U. Duncombe, "Infrared navigation—Part I: An assessment of feasibility (Periodical style)," *IEEE Trans. Electron Devices*, vol. ED-11, pp. 34–39, Jan. 1959.

[45] S. Chen, B. Mulgrew, and P. M. Grant, "A clustering technique for digital communications channel equalization using radial basis function networks," *IEEE Trans. Neural Networks*, vol. 4, pp. 570–578, July 1993.

[46] R. W. Lucky, "Automatic equalization for digital communication," *Bell Syst. Tech. J.*, vol. 44, no. 4, pp. 547–588, Apr. 1965.

[47] S. P. Bingulac, "On the compatibility of adaptive controllers (Published Conference Proceedings style)," in *Proc. 4th Annu. Allerton Conf. Circuits and Systems Theory*, New York, 1994, pp. 8–16.

[48] G. R. Faulhaber, "Design of service systems with priority reservation," in *Conf. Rec. 1995 IEEE Int. Conf. Communications*, pp. 3–8.

[49] W. D. Doyle, "Magnetization reversal in films with biaxial anisotropy," in *1987 Proc. INTERMAG Conf.*, pp. 2.2-1–2.2-6.

[50] G. W. Juette and L. E. Zeffanella, "Radio noise currents n short sections on bundle conductors (Presented Conference Paper style)," presented at the IEEE Summer power Meeting, Dallas, TX, June 22–27, 1990, Paper 90 SM 690-0 PWRS.

[51] J. G. Kreifeldt, "An analysis of surface-detected EMG as an amplitude-modulated noise," presented at the 1989 Int. Conf. Medicine and Biological Engineering, Chicago, IL.

[52] J. Williams, "Narrow-band analyzer (Thesis or Dissertation style)," Ph.D. dissertation, Dept. Elect. Eng., Harvard Univ., Cambridge, MA, 1993.

[53] N. Kawasaki, "Parametric study of thermal and chemical nonequilibrium nozzle flow," M.S. thesis, Dept. Electron. Eng., Osaka Univ., Osaka, Japan, 1993.

[54] J. P. Wilkinson, "Nonlinear resonant circuit devices (Patent style)," U.S. Patent 3 624 12, July 16, 1990.

[55] *IEEE Criteria for Class IE Electric Systems* (Standards style), IEEE Standard 308, 1969.

[56] *Letter Symbols for Quantities*, ANSI Standard Y10.5-1968.

[57] R. E. Haskell and C. T. Case, "Transient signal propagation in lossless isotropic plasmas (Report style)," USAF Cambridge Res. Lab., Cambridge, MA Rep. ARCRL-66-234 (II), 1994, vol. 2.

[58] E. E. Reber, R. L. Michell, and C. J. Carter, "Oxygen absorption in the Earth's atmosphere," Aerospace Corp., Los Angeles, CA, Tech. Rep. TR-0200 (420-46)-3, Nov. 1988.

[59] (Handbook style) *Transmission Systems for Communications*, 3rd ed., Western Electric Co., Winston-Salem, NC, 1985, pp. 44–60.

[60] *Motorola Semiconductor Data Manual*, Motorola Semiconductor Products Inc., Phoenix, AZ, 1989.

[61] (Basic Book/Monograph Online Sources) J. K. Author. (year, month, day). *Title* (edition) [Type of medium]. Volume(issue). Available: http://www.(URL)

[62] J. Jones. (1991, May 10). Networks (2nd ed.) [Online]. Available: http://www.atm.com

[63] (Journal Online Sources style) K. Author. (year, month). Title. *Journal* [Type of medium]. Volume(issue), paging if given. Available: http://www.(URL)

[64] R. J. Vidmar. (1992, August). On the use of atmospheric plasmas as electromagnetic reflectors. *IEEE Trans. Plasma Sci.* [Online]. *21(3).* pp. 876—880. Available: http://www.halcyon.com/pub/journals/21ps03-vidmar

Carlos G Hidalgo-Suarez Ingeniero de Sistemas, Magíster en Ingeniería, estudiante de Doctorado en la Universidad del Valle. Cali, Colombia. miembro del grupo de investigación en inteligencia artificial (GUIA).

Victor A Bucheli Ingeniero de Sistemas, Magíster en Ingeniería y Computación, Doctorado en Ingeniería. Profesor Asociado Universidad del Valle. Cali, Colombia. Director del Grupo de Investigación en Inteligencia Artificial (GUIA), Investigador Senior.

Hugo Ordoñez Ingeniero en Sistemas, Especialista en Administración Informática, Magíster en Ingeniería y Computación, Doctorado en Ingeniería. Profesor Universidad del Cauca. Cali, Colombia. Miembro del grupo de investigación GTI.