Revista Iberoamericana para la Investigación y el Desarrollo Educativo
ISSN 2007 - 7467

https://doi.org/10.23913/ride.v14i27.1714

Artículos científicos

Propuesta metodológica para mejorar el desempeño académico de los estudiantes en fundamentos de programación

Methodological Proposal to Improve the Academic Performance of Students in Fundamentals of Programming

Proposta metodológica para melhorar o desempenho acadêmico de alunos em fundamentos de programação

Lizzie Edmea Narváez Díaz
Universidad Autónoma de Yucatán, México
lendiaz@correo.uady.mx
https://orcid.org/0000-0003-0595-1932

Manuel Escalante Torres
Universidad Autónoma de Yucatán, México
manuel.escalante@correo.uady.mx
https://orcid.org/0009-0009-5727-7506

Cinhtia Maribel González Segura
Universidad Autónoma de Yucatán, México
gsegura@correo.uady.mx
https://orcid.org/0000-0002-9042-8320

Carlos Miranda Palma
Universidad Autónoma de Yucatán, México
cmiranda@correo.uady.mx
https://orcid.org/0000-0002-9692-4851

Maximiliano Canché Euán
Universidad Autónoma de Yucatán, México
maximiliano.canche@correo.uady.mx
https://orcid.org/0000-0003-0427-5207

Revista Iberoamericana para la Investigación y el Desarrollo Educativo
ISSN 2007 - 7467

Resumen

El dominio de los conceptos fundamentales de programación y su aplicación en la solución de problemas es percibido por estudiantes universitarios, docentes e investigadores como un proceso difícil del área de la computación. Por ende, el objetivo de este artículo es presentar un análisis de los resultados de la aplicación de una metodología construccionista de enseñanza y aprendizaje mediada por la herramienta Scratch. La investigación se realizó en la asignatura Algoritmia que se imparte en la licenciatura en Ingeniería de Software ofertada en la Unidad Multidisciplinaria Tizimín. El estudio empleó el paradigma cuantitativo, con una perspectiva experimental y enfoque longitudinal. Se trabajó con una muestra no probabilística que estuvo conformada por los alumnos que cursaron la citada materia. Los resultados indican que la metodología de enseñanza y aprendizaje implementada mediante el uso de Scratch es diferenciadora respecto al método tradicional que se venía empleando. Esto se reflejó después del análisis estadístico realizado a los datos obtenidos, los cuales fueron evaluados en función de tres aspectos: el promedio de las calificaciones, la cantidad de aprobados versus reprobados y el nivel de dominio de la asignatura. Con base en las evidencias del estudio se concluye que el camino propuesto es más simple y efectivo para la asimilación, apropiación y aplicación de los conceptos básicos de programación, lo cual ayuda al estudiante a tener un mejor rendimiento en la materia Algoritmia.

Palabras clave: algoritmo, metodología de enseñanza y aprendizaje, lenguaje de programación, *scratch*, universidad.

Abstract

The mastery of fundamental programming concepts and their application in problem solving is identified by university students, teachers, and researchers as a difficult process in the area of computing. The objective of this paper is to present an analysis of the results of the application of a constructionist teaching–learning methodology mediated by the programming environment Scratch. The research was conducted in the subject of Algorithmics taught in the Bachelor's Degree in Software Engineering offered by the Autonomous University of Yucatan (Uady) at the Tizimin Multidisciplinary Unit, located in Tizimin, Yucatan, Mexico. The study used the quantitative paradigm under an experimental perspective with a longitudinal approach and worked with a non-probabilistic sample, which was made up of all the students who took the aforementioned subject. The results indicate that the implemented teaching and learning methodology is differentiating from the

traditional method that had been used, conclusion reached after the statistical analysis of the data obtained, which were evaluated in terms of three aspects: the average of the grades, the pass versus fail rate, and the level of mastery on the subject. Based on the evidence of the study, it is concluded that the proposed pathway is simpler and more effective for the assimilation, appropriation and application of basic programming concepts, which helps students to perform better in the subject of Algorithmics.

Keywords: Algorithms, Teaching–learning methodology, Programming languages, Scratch, University.

Resumo

Dominar os conceitos fundamentais de programação e sua aplicação na resolução de problemas é percebido por estudantes universitários, professores e pesquisadores como um processo difícil na área de computação. Portanto, o objetivo deste artigo é apresentar uma análise dos resultados da aplicação de uma metodologia construcionista de ensino e aprendizagem mediada pela ferramenta Scratch. A pesquisa foi realizada na disciplina Algoritmos ministrada na graduação em Engenharia de Software oferecida na Unidade Multidisciplinar de Tizimín. O estudo utilizou o paradigma quantitativo, com perspectiva experimental e abordagem longitudinal. Trabalhamos com uma amostra não probabilística composta por alunos que cursaram a referida disciplina. Os resultados indicam que a metodologia de ensino e aprendizagem implementada através do uso do Scratch é diferente do método tradicional que vinha sendo utilizado. Isso se refletiu após a análise estatística realizada dos dados obtidos, que foi avaliada com base em três aspectos: a média das notas, o número de aprovados versus reprovados e o nível de domínio da matéria. Com base nas evidências do estudo, conclui-se que o caminho proposto é mais simples e eficaz para a assimilação, apropriação e aplicação dos conceitos básicos de programação, o que auxilia o aluno a ter melhor desempenho na disciplina de Algoritmos.

Palavras-chave: algoritmo, metodologia de ensino e aprendizagem, linguagem de programação, scratch, universidade.

Fecha Recepción: Marzo 2023 **Fecha Aceptación:** Noviembre 2023

Introducción

La palabra *algoritmo* surge en el siglo IX y fue planteada por el matemático árabe Al–Khwarizmi (Juarismi o Khorezmi), quien desarrolló gran parte de su carrera alrededor del año 800 d. C., trabajando de modo particular con el álgebra y la astronomía. Su aporte en la resolución de ecuaciones y el tratado sobre números *Algoritmi de numero indorum* lo sitúa como el referente más antiguo de la palabra *algoritmo*, y a través de los años esta palabra se ha convertido en uno de los fundamentos de la computación (Cátedra Conceptos de Algoritmos, Datos y Programas, UNLP, 2016; Puig, 2009).

El algoritmo se puede definir como una secuencia de pasos, procedimientos o acciones que permiten alcanzar un resultado o resolver un problema. En otras palabras, es un conjunto ordenado de reglas para abordar cierta clase de situaciones o problemas y una forma de plantear su solución (Cairó, 2005; Joyanes, 2003). El algoritmo debe tener como característica final la posibilidad de transcribirlo fácilmente a un lenguaje de programación (Pinales y Velázquez, 2014).

A pesar de los años que han pasado desde el surgimiento de ese concepto, y pese a todos los estudios que se han realizado en el área de la computación, es un hecho que en los primeros cursos universitarios existen problemas y dificultades para los alumnos cuando empiezan a estudiar los fundamentos de la programación (Colorado, 2020; González *et al*., 2012).

Además de los resultados y de la opinión de los estudiantes, diversos docentes e investigadores coinciden en lo complejo que resulta enseñar y aprender, así como aplicar la programación y sus fundamentos. Los programadores novatos manifiestan una gran variedad de dificultades y deficiencias debido a la complejidad que conlleva (Insuasti, 2016; Robins *et al*., 2003). En tal sentido, Winslow (1996, citado por Robins *et al*., 2003) afirma que se requieren al menos 10 años de experiencia para pasar de un nivel de programador novato a uno experto, lo cual indica que cuando un alumno termina su carrera todavía tiene que recorrer un largo camino de aprendizaje para convertirse en un programador experimentado.

Las clases iniciales de programación a las que se enfrentan los estudiantes universitarios deben resultar cautivadoras y atractivas para despertar el interés en las ciencias de la computación. No obstante, la evidencia sugiere que estas clases distan mucho de resultar atractivas para la mayoría de los alumnos, en comparación con otros cursos de ciencias, matemáticas, tecnología e ingeniería (Campbell y Atagana, 2022).

Adicionalmente, adquirir habilidades para la programación es una actividad complicada, pues, en primer lugar, se debe comprender el problema, luego crear un algoritmo que esboce la solución y, por último, traducirlo a un lenguaje de programación. Esta última actividad es un proceso repetitivo de cuatro etapas: la codificación, el proceso de compilación, prueba del código y, si falla, depurarlo (López *et al*., 2011). Por eso, Soloway y Spohrer (1989, citados por Insuasti, 2016) afirman que para una persona estas tareas pueden ser muy difíciles.

Por otro lado, en los planes de estudio del área de ciencias de la computación e informática, el interés del currículo en los primeros semestres se concentra en las matemáticas y los cursos introductorios a la programación, los cuales tienen relación con los altos índices de deserción y pérdida académica. Este aspecto es un punto muy importante para considerar debido a que es precisamente en estos cursos donde se desarrollan las competencias básicas para los diferentes programas académicos (Rodríguez, 2014).

Al respecto, Muñoz *et al*. (2012) señalan que se han propuesto muchos enfoques, metodologías y herramientas de enseñanza para ayudar a los alumnos en este proceso de aprendizaje; sin embargo, a la fecha no parece existir un camino satisfactorio. De acuerdo con Jiménez-Toledo *et al*. (2019), es importante que el alumno tenga habilidades para resolver problemas, que pueda aplicar y al mismo tiempo crear sus propios modelos cognitivos, lógicos, matemáticos y algorítmicos. Sin embargo, aunque podría parecer que la aplicación de los conceptos básicos de programación es un proceso relativamente simple, diversos autores —incluyendo a los del presente estudio— coinciden en que estos principios básicos con frecuencia resultan complicados para los alumnos, situación considerada como un producto en el que convergen diversos factores. En este sentido, Insuasti (2016) complementa las ideas anteriores indicando que algunas de las razones por las que muchos estudiantes no alcanzan los resultados esperados pueden ser la dificultad de dominar el lenguaje de programación, los principios básicos del área, los antecedentes de los alumnos y la falta de una buena didáctica por parte del maestro, por mencionar las más importantes.

Se puede concluir, por tanto, que adquirir conocimientos de programación y formar profesionales expertos en esta área presenta diversos retos tanto para los docentes como para los mismos estudiantes, especialmente cuando se enseña con el enfoque tradicional, es decir, a papel y lápiz (Malliarakis *et al*., 2014). Esta transmisión de conocimientos genera elevados índices de deserción académica en cursos de fundamentos o principios básicos de programación (Rodríguez, 2014). En consecuencia, se ha formulado la siguiente

interrogante: ¿cómo disminuir los niveles de deserción y/o abandono en las materias o cursos introductorios a la programación de computadoras?

En atención a las líneas anteriores, la Universidad Autónoma de Yucatán (UADY) cuenta con el programa de Ingeniería de Software (LIS), el cual se empezó a ofertar en la Unidad Multidisciplinaria Tizimín (UMT) a partir del semestre agosto-diciembre de 2016. Como parte del plan de estudios se incluye la materia Algoritmia, cuya competencia radica en diseñar algoritmos para la solución de problemas del área computacional. En este curso el estudiante parte de un problema de la vida real, el cual debe analizar y comprender para luego aplicar sus habilidades con el fin de construir un algoritmo que permita resolver el problema inicial. Este proceso lo ayuda en el desarrollo del pensamiento ordenado y lógico, así como en la adquisición de habilidades para crear formas diversas de solucionar un problema, de forma que luego pueda trasladar su solución en algún lenguaje de programación (Gómez *et al*., 2016). Esta asignatura no pretende introducir al estudiante al mundo de la programación, sino que estudia aquellos aspectos que anteceden esta tarea.

En las primeras dos ocasiones en las que se enseñó el curso de Algoritmia en el programa de Ingeniería de Software de la UADY-UMT, los índices de aprobación no fueron los esperados. Los porcentajes obtenidos se detallan en la tabla 1.

Tabla 1. Desempeño de estudiantes al inicio de Ingeniería de Software

Semestre	Satisfactorio/Sobresaliente	Suficiente/No aprobado
Agosto-diciembre 2016	58.3 %	41.7%
Agosto-diciembre 2017	46.2 %	53.8%
Promedio	52.25 %	47.75%

Fuente: Elaboración propia

Teniendo como base lo expuesto en la tabla 1, se puede afirmar que existe una falta de conocimiento de los estudiantes en los temas, lo cual puede ocasionar problemas en cursos posteriores del plan de estudios en los que se requiere un dominio sólido de las habilidades derivadas de las bases de la programación.

A partir de lo anterior, los docentes implicados decidieron implementar en la asignatura de Algoritmia una metodología de enseñanza y aprendizaje desarrollada expresamente para apoyar al alumnado en los procesos introductorios a la programación de computadoras, la cual estuvo mediada por el uso de la herramienta TIC de programación por bloques Scratch,

el cual fue considerado por ser amigable e intuitivo para los estudiantes. Sobre este aspecto, Gómez y Martínez (2021) mencionan que en la programación basada en bloques la validez de las combinaciones que se hagan con estos está representada por los mismos bloques, es decir, todo es en forma visual, de modo que se suprime la representación de la sintaxis de modo explícito, lo cual resulta de suma importancia para la asignatura.

Tomando en consideración lo que se ha planteado, el objetivo del estudio fue analizar con métodos estadísticos si se logra elevar el rendimiento académico de los estudiantes de la materia Algoritmia después de la implementación de la metodología de enseñanza y aprendizaje previamente citada en la LIS de la UADY–UMT.

Problemas en el aprendizaje de la programación

El problema descrito en los párrafos previos, que puede resumirse como la dificultad de aprender a programar, ha sido abordado en diversas investigaciones descritas en la literatura, donde se ha planteado una variedad de estrategias de solución, algunas de las cuales son expuestas en los siguientes párrafos.

Por ejemplo, Rizvi *et al.* (2011) describen un estudio que aborda la preocupante retención de estudiantes de los primeros semestres en programas del área de la computación en la Norfolk State University, Estados Unidos. En su trabajo, estos autores proponen diseñar e impartir un curso previo, a manera de curso propedéutico, antes de inscribir el primer curso introductorio a la programación, denominado CS1, el cual forma parte de la carga académica obligatoria en estas carreras. En el curso previo, denominado CS0, se implementa la enseñanza asistida con TIC. La propuesta consistió en utilizar Scratch como herramienta para aprender los conceptos básicos de programación. Esta fue implementada debido a que, en la Norfolk State University, en el periodo evaluado del año 2005 a 2007, únicamente el 42 % de los estudiantes habían aprobado el curso CS1, lo cual enciende un foco de alerta por la necesidad de implementar acciones que favorezcan la retención de estudiantes. Los autores emplearon esta estrategia para enfrentar los problemas de retención que tenían, y se obtuvo que el 66 % de los alumnos que llevaron el curso CS0 lograron aprobar el CS1 contra el 44 % del año anterior, cuando los alumnos fueron inscritos directamente al curso CS1.

Análogamente, Muñoz *et al*. (2012) describen el caso del programa de Ingeniería Civil en Informática ofertado en la Universidad de Valparaíso, Chile, donde se presentó una alta deserción en los primeros años de la carrera y una de sus asignaturas con mayor índice de reprobación era Fundamentos de Programación con el 32.4 %, seguida de Cálculo y Álgebra

Elemental, resultados que coinciden con los hallazgos de otros autores mexicanos (Colorado, 2020; González *et al*., 2012). En la universidad chilena, con el propósito de apoyar a los estudiantes, se realizó un estudio utilizando encuestas para encontrar los temas que se les dificultaban durante el proceso de aprendizaje, los cuales fueron dos básicamente: estructuras cíclicas y manejo de arreglos de dos o más dimensiones.

A partir de la alta tasa de deserción reportada por universidades chilenas se realizó un estudio en la Universidad de Valparaíso en la carrera de Ingeniería en Informática ofrecido en la Escuela de Ingeniería Informática, dado que experimentaban la misma situación de deserción. En concreto, la tasa más alta de abandono se hallaba en el cuarto semestre de la carrera, específicamente en Fundamentos de Programación, una de las materias con mayor número de alumnos reprobados. Los estudiantes, además de cursar la asignatura citada, tenían que tomar dos cursos relacionados llamados Programación I y Programación II. A esta última materia se matriculó un grupo reducido en comparación con los que iniciaban (casi 30 % menos) y la tasa de retención no lograba superar el 50 % en los tres primeros años. Para apoyar a los alumnos a pasar estos cursos se propusieron cambios implementando el uso de herramientas como Scratch y Lego Mindstorms (Muñoz *et al*., 2015, 2017).

En Colombia, en la carrera de Ingeniería en Sistemas ofertada por la Universidad Mariana, se consideró que las asignaturas introductorias a la programación son un factor determinante para la continuidad de los estudiantes en la carrera, ya que predomina la pérdida y abandono del curso. Los resultados sugieren que esta situación es causada por la desmotivación y frustración, lo cual se ve reflejado en el abandono de la carrera o en el incremento del tiempo de permanencia (Hernández, 2013).

En la Universidad del Valle en Colombia se imparte la carrera Ingeniería de Sistemas, donde se dicta la materia de Introducción a la Programación, denominada CS1 como requisito. En este curso es fundamental que los estudiantes desarrollen competencias esenciales para la resolución de problemas computacionales complejos mediante la adquisición de destrezas que les permitan el empleo de un lenguaje de programación. Se verificó que al finalizar el curso pocos alumnos lo concluían satisfactoriamente para continuar con la siguiente asignatura (CS2). Además, se observaron bajas notas, exceso de faltas y una alta tasa de deserción. Para apoyar a su planta de estudiantes la universidad propuso diseñar estrategias de colaboración y uso de herramientas TIC que repercutieran en un mejor aprendizaje. Con el tiempo estas estrategias elevaron las calificaciones de los

estudiantes, así como también sus habilidades interpersonales, lo cual los incentivó a mejorar los cursos de programación (Hidalgo *et al*., 2021).

De igual modo, Bedoya (2021) señala que, en la Universidad Pontificia Bolivariana, ubicada en Medellín, Colombia, también tienen problemas de bajo rendimiento y deserción estudiantil cuando imparten cursos orientados a los fundamentos de programación en diversas carreras de ingeniería (no solo en aquellas ligadas expresamente a la computación), por lo que realizaron un diagnóstico a los estudiantes para brindarles apoyo en el proceso.

Estudios que fundamentan la metodología propuesta

Con la idea de emplear una metodología de enseñanza y aprendizaje novedosa y efectiva, se consideró el uso de la herramienta Scratch. Según Jiménez-Toledo *et al*. (2019), es una herramienta que se ubica en la categoría de *Juegos educativos centrados en la enseñanza de unidades múltiples de aprendizaje*. En esta categoría, con el apoyo de la lúdica, se asiste al estudiante fomentando su participación con actividades que propician el disfrute y la creatividad en el proceso de enseñanza. Scratch es un lenguaje de programación basado en bloques con una interfaz visual, lo que ayuda al alumno a ver de manera gráfica los algoritmos que diseña. En consecuencia, los puede comprender con mayor facilidad, ya que la visualización gráfica y ejecutable suele resultar más fácil de analizar que la creación del algoritmo con papel y lápiz, como ocurre con el método tradicional de enseñanza de la materia.

En esta misma categoría de herramientas, además de Scratch, se encuentran otras como las siguientes: JKarel Robot, Blockly, Robot Karel20, Greenfoot, APP Inventor, Alice, Lego, Turingal (Jiménez-Toledo *et al*., 2019) y Logo, proyecto pionero en el área, desarrollado por Seymour Papert y colaboradores. Este último inicialmente se diseñó para introducir conceptos lógico-matemáticos en niños utilizando constructivamente un entorno de programación, pero posteriormente se orientó a la enseñanza de conceptos básicos relacionados con la programación (Solomon *et al*., 2020). Lo anterior fue un elemento clave que se tomó en cuenta en el diseño de la metodología que se utilizó, para lo cual se revisaron diversos estudios enfocados en la resolución de la problemática planteada.

Al respecto, Rodríguez (2014) afirma que en el área de la programación está siendo novedoso el uso de lenguajes como Python, así como el empleo de entornos de desarrollo integrado orientados al uso de elementos lúdicos, como Jiménez-Toledo *et al*. (2019) propusieron posteriormente. Rodríguez (2014) argumenta que con el fin de disminuir la

deserción en asignaturas del área de programación actualmente se han venido creado diversas metodologías, las cuales incluyen la lúdica como un pilar del aprendizaje. En este sentido, se pueden mencionar tres propuestas de interés:

- Creaciones por medio de videojuegos.
- Desarrollo instruccional orientado hacia el paradigma.
- Desarrollo instruccional enfocado en temáticas particulares.

En cuanto a la clasificación *Creaciones por medio de vídeo juegos*, se destacan los proyectos Greenfoot de la Universidad de Kent, Alice de la Universidad Carnegie Mellon, y Scratch del Massachusetts Institute of Technology (de gran interés para este proceso investigativo), herramientas que procuran reducir la tasa de abandono en programación (Rodríguez, 2014).

Considerando las características de Scratch y su compatibilidad con las competencias que se procuran lograr en la materia Algoritmia, se incluyó esta herramienta en el diseño de la metodología que se implementó. Uno de los rasgos comunes que se puede mencionar es que no se requiere la elaboración de programas mediante alguna sintaxis o lenguaje de programación específico. En este sentido, Scratch cuenta con un entorno orientado a la creación proyectos interactivos, animaciones, juegos y más por medio de bloques. Además, es un *software* libre y gratuito, es un lenguaje de programación que está a la vanguardia (Calderón, 2020), y cuenta con el respaldo de la institución que lo desarrolla. Asimismo, tiene una gran comunidad de usuarios a nivel global, así como foros de discusión y apoyo técnico, una wiki y un sitio web que proporcionan respaldo.

Scratch se puede definir como un lenguaje de programación visual para aprender conceptos de programación y crear proyectos interactivos como animaciones, juegos e historias. Además, no tiene sintaxis específicas (como los lenguajes tradicionales), sino que se basa en arrastrar y soltar bloques, los cuales tienen el código integrado. Esto facilita el aprendizaje y lo acerca a personas con poca o nula experiencia en programación. Scratch fomenta la creatividad, el razonamiento sistemático y promueve el trabajo y la colaboración entre sus usuarios (MIT Scratch Team, s. f.; Monjelat *et al*., 2018).

En la Universidad Estatal de Milagro, Ecuador, a pesar de haber implementado una variedad de metodologías de enseñanza, la tasa histórica de aprobación en cursos de inicio de programación se mantenía por debajo del 43 %. En consecuencia, decidieron abordar la problemática optado por usar lenguajes de programación visuales para enseñar de una manera más práctica y didáctica (en particular Scratch). Al concluir las pruebas, encontraron que el

grupo experimental mostró una tasa de aprobación cuatro veces mayor que el grupo de control, y a pesar de que el resultado fue incipiente, se considera que están en buen camino para brindar a los estudiantes el apoyo que requieren para aprobar. Además, esta estrategia resultó atractiva para la mayoría de los estudiantes (Cárdenas *et al*., 2021).

Como apoyo a lo que se ha venido comentando en los párrafos previos, un grupo de docentes de la Universidad Rey Juan Carlos (Madrid) piensa que se le debe brindar apoyo al estudiante desde el nivel preuniversitario. Por eso, sugieren que primero se debe de formar al docente preuniversitario, de modo que luego pueda impartir cursos relacionados con la programación de computadoras. Para ello, crearon una maestría de formación docente enfocada en la competencia digital y la programación, la cual prepara al profesor en temas relacionados mediante el uso de herramientas como ScratchJr, Scratch y App Inventor con el fin de ayudar a los alumnos antes de su ingreso a la vida universitaria. Al ser un estudio reciente, aún no se cuentan con resultados contundentes; sin embargo, se destaca la importancia de herramientas como Scratch (Velázquez *et al*., 2023).

Un grupo de investigadores, después de la aplicación de la herramienta Scratch en un curso introductorio a la programación, llegaron a la conclusión de que el uso de este recurso se sustenta en un proceso construccionista que involucra favorablemente a los estudiantes de ciencias de la computación en los primeros años, especialmente a aquellos sin experiencia previa en programación (Campbell y Atagana, 2022).

Como complemento de la literatura que se ha venido exponiendo en este apartado, también se encontró en Pérez-Narváez *et al*. (2020) el uso de la herramienta Scratch en la Universidad Central del Ecuador, específicamente en la carrera de Informática para el Desarrollo del Pensamiento Computacional. Si bien es cierto esta no se enfoca expresamente en los fundamentos de programación, uno de sus pilares son los algoritmos, los cuales desarrollaron al incluir diversos temas computacionales, como variables, secuencia, uso de operadores, entre otros. Los resultados evidenciaron un mayor desarrollo en los pilares del pensamiento computacional, incluido el citado.

Por las razones expuestas, y de acuerdo con la información encontrada en la literatura revisada, se integró el lenguaje Scratch a la metodología implementada para mejorar los resultados de aprendizaje obtenidos con los alumnos de la materia de Algoritmia que cursaban la carrera Ingeniería de Software. Por ende, el presente trabajo documenta el proceso de investigación llevado a cabo con el fin de apoyar la enseñanza y el aprendizaje de Algoritmia en la UMT.

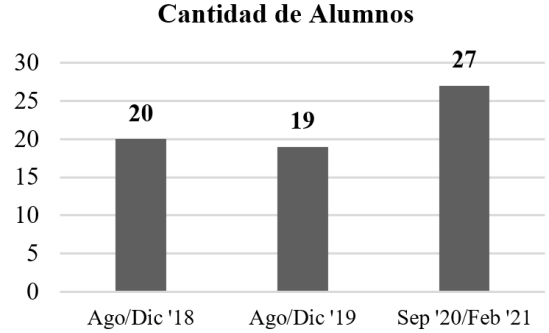
Revista Iberoamericana para la Investigación y el Desarrollo Educativo
ISSN 2007 - 7467

Método

El contexto de este estudio fue la Unidad Multidisciplinaria Tizimín, ubicada en la ciudad de Tizimín, la cual pertenece a la Universidad Autónoma de Yucatán. Durante los semestres enero-mayo y agosto-diciembre 2022 la matrícula de esta unidad académica fue de 454 alumnos, de acuerdo con datos internos de la institución, distribuidos en las cuatro licenciaturas ofertadas: Contaduría Pública, Enfermería, Educación e Ingeniería de Software.

La investigación fue experimental, con un enfoque de tipo cuantitativo y longitudinal (Hernández *et al*., 2014). Esta fue realizada con la participación de estudiantes del primer semestre de la citada licenciatura, los cuales vienen de las preparatorias de Tizimín y sus comisarías, así como de otros estados aledaños, aunque en menor número. El estudio documenta el trabajo que se realizó con tres generaciones de estudiantes, quienes estuvieron matriculados en los periodos escolares agosto-diciembre 2018-2019 y septiembre 2020-febrero 2021. Este último periodo tuvo que ser adecuado debido a la crisis sanitaria del covid-19. El número total de estudiantes regulares inscritos durante estos semestres fue de 66, distribuidos como se muestra en la figura 1.

Figura 1. Alumnos participantes distribuidos por semestres



Fuente: Elaboración propia

Considerando el tamaño de la población, se decidió no aplicar un método de muestreo, sino que se consideró a todos los estudiantes como parte de la muestra, es decir, participaron todos los estudiantes registrados en la materia Algoritmia del primer semestre de la carrera Ingeniería de Software durante los periodos académicos mencionados.

Diseño de la metodología propuesta

Para verificar lo anterior fue necesario obtener ciertos datos de la materia de Algoritmia durante varios ciclos escolares, por lo que se recabaron las actas de calificaciones finales de la asignatura en los tres grupos con los que se trabajó la metodología diseñada empleando Scratch, correspondientes a los cursos 2018, 2019 y 2020-2021. Asimismo, se analizaron los datos análogos de los semestres previos llevados a cabo sin usar algún tipo de metodología específica, es decir 2016 y 2017. Estos datos se recabaron para realizar las comparaciones entre los métodos de enseñanza. Las actas fueron obtenidas de la UADY por medio de su Sistema de Información y Control Escolar Institucional (SICEI).

A continuación, se presenta la metodología empleada con los alumnos de Algoritmia de la LIS, la cual estuvo dividida en cuatro fases:

Fase 1

Previo al inicio del curso formal de Algoritmia se aplicó una evaluación diagnóstica que permitió obtener información relacionada con temas de matemáticas, algoritmos y datos generales de los estudiantes. En la evaluación fueron incluidos tópicos identificados como prioritarios para el desarrollo de algoritmos, es decir, utilización de operadores, cálculo de porcentajes, operaciones simples con una incógnita y descomposición de problemas. Se sugirieron estos temas específicos, dado que fueron identificados como relevantes y esenciales para desarrollar las habilidades relacionadas con la programación.

Posteriormente, se impartió a los estudiantes un curso propedéutico para nivelar sus conocimientos relacionados con las matemáticas y la programación. Este curso se llevó a cabo con algunas sesiones impartidas en el aula de clases y otras en el laboratorio. Al finalizar se aplicó una prueba de desempeño y se analizaron los datos obtenidos (tabla 2).

Tabla 2. Fase 1. Diagnóstico aplicado antes del curso Algoritmia

1. Obtención de datos generales	
2. Aplicación de una prueba diagnóstica	
Curso	Impartición del curso en el aula de clase y en el laboratorio Aplicación de prueba de desempeño final
Proceso final	Análisis de la prueba de desempeño final Identificación de errores principales

Fuente: Elaboración propia

Fase 2

En esta fase, la propuesta consistió en realizar de manera paralela dos procesos de aprendizaje: por un lado, y a partir del plan curricular de la licenciatura, se impartió la asignatura Algoritmia en el aula de clases; el segundo proceso consistió en invertir tiempo adicional para cursar un taller práctico en el laboratorio con apoyo de la herramienta Scratch. Con este taller se reforzaron los temas vistos en el aula, ya que el estudiante pudo codificar los ejercicios planteados y resueltos previamente utilizando el ambiente gráfico de programación que ofrece Scratch. Así, cada algoritmo diseñado como diagrama de flujo fue codificado. La tabla 3 ilustra el proceso detallado de las acciones que se realizaron en esta segunda fase.

Tabla 3. Fase 2. Actividades implementadas durante el curso de Algoritmia

1. Actividades en el salón de clase	
1	Explicar el contenido
2	Plantear el problema
3	Analizar el problema
4	Elaborar el algoritmo
2. Actividades en el laboratorio	
5	Traducir el algoritmo a un programa
6	Ejecutar el programa Verificar si tiene errores Afirmativo ir al paso 3 Negativo continuar
7	Analizar el programa
8	Aclarar dudas
9	Evaluar

Fuente: Elaboración propia

Fase 3

Los estudiantes se organizaron en equipos para el desarrollo de un proyecto final mediante el uso del lenguaje Scratch, y aplicando el enfoque educativo del aprendizaje basado en proyectos. Esta actividad contempló el uso de todas las estructuras que fueron estudiadas durante el proceso de intervención. El docente fue quien decidió el momento de empezar con esta fase de acuerdo con el avance del grupo. La tabla 4 enseña el detalle de cada una de las partes en las que estuvo dividido el trabajo realizado.

Tabla 4. Fase 3. Desarrollo de proyecto final

1. Formación de equipos	
2. Mecánica de trabajo	
I	Determinar el proyecto Diseñar la narrativa Buscar y recopilar información
II	Determinar actividades Construir elementos
III	Programar Documentar Entregar el proyecto Evaluar

Fuente: Elaboración propia

Fase 4

Aquí se analizaron todos los resultados alcanzados de las fases previas. Cabe mencionar que esta metodología se volvió a aplicar cuando se impartió de nuevo la materia Algoritmia, por lo que al término de todas las fases se hicieron los ajustes necesarios para emplearla en la siguiente impartición del curso (tabla 5).

Tabla 5. Fase 4. Evaluación de resultados

1. Evaluar datos obtenidos
2. Realizar ajustes para la siguiente implementación

Fuente: Elaboración propia

Descripción de la estrategia de análisis de datos

La estrategia empleada para llevar a cabo el análisis de los datos recabados se realizó del modo que se detalla a continuación. En primera instancia, se aplicó estadística descriptiva a los datos recabados, tanto de aquellos cursos en los que no se implementó ninguna metodología (2016 y 2017) como de aquellos en los que sí se empleó la metodología mencionada (2018, 2019 y 2020-2021). Para la evolución se tomaron en cuenta tres aspectos: promedio de calificaciones, cantidad de alumnos aprobados y nivel de dominio. Como resultado de estas pruebas, en todas las evaluaciones el uso de la metodología dio una ventaja diferenciadora. Para probar que esta diferencia era significativa, a los datos mencionados se les aplicó estadística inferencial, mediante el empleo de pruebas que se detallan seguidamente.

En todas las evaluaciones se empleó el *software* estadístico RStudio en la versión R-4.2.3 del 2023, utilizando un nivel de significancia del 5 % ($\propto = 0.05$). Este valor es adecuado para análisis de datos, lo que permite equilibrar el riesgo de cometer un error tipo I y tipo II.

Mediante la estadística inferencial, en esta segunda parte del análisis de datos se llevaron a cabo las siguientes pruebas:

- En cuanto al primer aspecto evaluado (promedio de calificaciones), primero se verificó si los datos seguían un comportamiento normal para que a partir del resultado se decidiera sobre el tipo de estadístico por emplear. El resultado fue una no normalidad y se decidió utilizar la prueba de Mann-Whitney, la cual es estadística no paramétrica y se emplea para comparar dos grupos independientes y determinar si hay diferencias significativas entre las distribuciones de dos muestras, utilizando las medianas en lugar de los promedios. Para complementar la prueba se calcularon las medidas de tamaño del efecto. En este caso, se empleó el coeficiente de rango de biserial r. La interpretación de este coeficiente es recomendable cuando la prueba de Mann-Whitney ya indicó una diferencia significativa.
- Para la evaluación de la cantidad de aprobados, y considerando el tamaño de la muestra, se aplicó la prueba Xi cuadrada. Respecto a los coeficientes que ayudan a cuantificar una fuerza de asociación entre las variables, se encontraron el coeficiente Phi, V de Cramer y el coeficiente de contingencia.
- En cuanto al nivel de dominio, y considerando el tamaño de la muestra, se aplicó la prueba Xi cuadrada. Respecto a los coeficientes que ayudan a cuantificar una fuerza de asociación entre las variables, se encontraron el coeficiente Phi, V de Cramer y el coeficiente de contingencia.
- De modo complementario, se procedió a comprobar si el promedio era superior a 80 puntos. Primero se verificó si los datos seguían un comportamiento normal para que a partir del resultado se decidiera sobre el tipo de estadístico por emplear. El resultado fue una no normalidad y se decidió utilizar la prueba Wilcoxon. Para complementarla se calcularon las medidas de tamaño del efecto, en este caso usando el coeficiente de rango de biserial r.

Resultados

La aplicación de la metodología descrita se realizó con el propósito de que los logros académicos de los estudiantes fueran mejor en la asignatura de Algoritmia. En tal sentido, y como fue descrito en la sección anterior, las actas de examen recuperadas del SICEI fueron el instrumento mediante el cual se pudieron obtener los datos que posteriormente fueron analizados para verificar el logro del objetivo propuesto.

Las actas de calificaciones fueron obtenidas al finalizar los semestres escolares citados y se evaluaron los resultados en aquellos en los que se empleó esta metodología. Además, se retomaron los datos correspondientes a los semestres previos que fueron impartidos de modo tradicional (sin el empleo de metodología alguna) con el fin de hacer el análisis comparativo correspondiente. Lo anterior se detalla en la tabla 6. Esta información fue recabada durante los cinco cursos implicados en el estudio.

Tabla 6. Semestres analizados de la materia Algoritmia

Curso	Semestre	Estrategia
1	Agosto-diciembre 2016	Tradicional
2	Agosto-diciembre 2017	Tradicional
3	Agosto-diciembre 2018	Con metodología usando Scratch
4	Agosto-diciembre 2019	Con metodología usando Scratch
5	Septiembre 2020-febrero 2021	Con metodología usando Scratch

Fuente: Elaboración propia

Cabe destacar que durante los semestres en los cuales fue impartida la asignatura de Algoritmia siempre el mismo maestro estuvo a cargo de todas las sesiones en los diversos periodos escolares. Asimismo, la planeación didáctica fue desarrollada en el año 2016 y desde ese momento se siguen los mismos temas en cada periodo escolar. Por último, es importante señalar que los estudiantes tienen antecedentes educativos diversos debido a la gran variedad de sistemas de los cuales provienen (siete en total), dato obtenido mediante una encuesta breve realizada al inicio de los cursos.

Los datos recabados fueron evaluados con el propósito de determinar si la metodología empleada era diferenciadora. Para ello, se hizo una división entre aquellos datos recabados en los cursos impartidos de modo tradicional y aquellos en los que fue empleada la herramienta Scratch, lo cual fue concretado en función de tres variables:

- Promedio, modo tradicional-uso de la herramienta Scratch.
- Cantidad de aprobados-reprobados.
- Nivel de dominio.

Es importante citar que el nivel de dominio queda definido como aquellas características que describen el grado del desarrollo de las competencias de una materia por parte del estudiante, el cual ve reflejado el resultado logrado en una asignatura de dos modos: cuantitativo (0 al 100) y cualitativo (sobresaliente, satisfactorio, suficiente y no acreditado). Esto se registra en el acta de examen (tabla 7) (UADY, 2012).

Tabla 7. Niveles de dominio en la UADY

Niveles de dominio	
Puntaje	Categoría
90 – 100	Sobresaliente (SS)
80 – 89	Satisfactorio (SA)
70 – 79	Suficiente (S)
0 - 69	No acreditado (NA)

Fuente: UADY (2012)

En cuanto al primer aspecto evaluado, promedio de los cinco semestres (modo tradicional-uso de Scratch), los resultados se pueden ver en la tabla 8. Una columna contiene las medianas de los dos grupos, dado que la mediana juega un papel fundamental en el análisis inferencial de los datos; además, en ella se aprecia un aumento de 9.37 puntos en la calificación promedio y de 8 puntos en la calificación mediana en los grupos donde se utilizó la metodología propuesta.

Tabla 8. Promedio de aprovechamiento final en Algoritmia 2016 a 2021

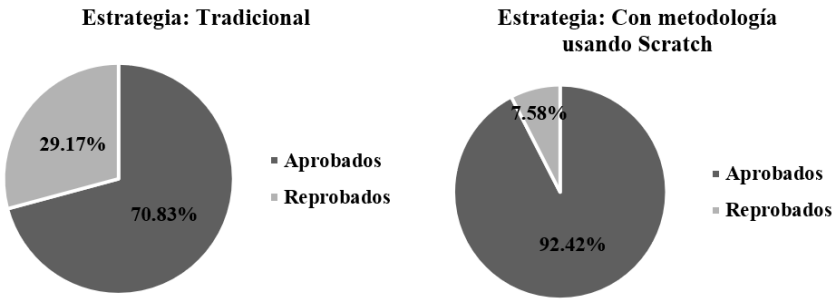
Semestres	Estrategia	Promedio	Mediana
2016 y 2017	Tradicional	75.75	80
2018, 2019 y 2020–2021	Con metodología	85.12	88

Fuente: Elaboración propia

En cuanto al segundo aspecto, es decir, la evaluación de la cantidad de aprobados y reprobados, la figura 2 refleja los resultados generados de todos los cursos mediante porcentajes. Cabe mencionar que la cantidad de estudiantes evaluados con el método tradicional fue de 24 y al usar la metodología mediada con Scratch participaron 66 alumnos. De modo similar a la evaluación del promedio y mediana, en este segundo aspecto también

se generó un aumento a favor de la metodología empleada (21.59 %). En consecuencia, la cantidad de reprobados es menor en los cursos donde se aplicó la metodología en comparación con los que fueron impartidos de modo tradicional.

Figura 2. Evaluación de la cantidad de aprobados y reprobados



Fuente: Elaboración propia

En la tabla 9 se exponen los resultados del último aspecto evaluado (nivel de dominio), donde también se aprecia un incremento a favor después de la aplicación de la metodología mediada con Scratch (24.62 %).

Tabla 9. Atendiendo al nivel de dominio en Algoritmia 2016 a 2021

Estrategia	Porcentaje en nivel satisfactorio o sobresaliente
Tradicional	54.17 %
Con metodología usando Scratch	78.79 %

Fuente: Elaboración propia

Como se observa, la metodología implementada con Scratch logró incrementar el desempeño de los participantes; sin embargo, esto no fue suficiente para el estudio, ya que con el fin de verificar que los resultados encontrados eran estadísticamente significativos, se implementaron otras pruebas con el uso del *software* RStudio. En los siguientes párrafos se detallan los estadísticos empleados.

En cuanto al primer aspecto examinado, se verificó si la mediana de los alumnos cuando emplean la metodología propuesta tenía una diferencia estadísticamente significativa comparada con el empleo del método tradicional. Así, se cambió el promedio por la mediana como consecuencia de que los datos resultaron con una distribución no normal. La tabla 10 presenta la hipótesis planteada, así como la evolución de los resultados.

Tabla 10. Evolución de los resultados del análisis del promedio

Hipótesis	Prueba	Resultados 2018	Resultados 2019	Resultados 2020 - 2021
H0: Datos normales H1: Datos no normales	Shapiro-Wilk	p-valor = 0.000589 < α = 0.05 Rechazo H0, no normalidad	p-valor = 0.0000221 < α = 0.05 Rechazo H0, no normalidad	p-valor = 1.849-06 < α = 0.05 Rechazo H0, no normalidad
H01: M1 = M2 H11: M1 ≠ M2	Mann-Whitney	p-valor = 0.6623 > α = 0.05 No se encontró suficiente evidencia para concluir que existe diferencia significativa entre las medianas de aprovechamiento. Las mismas medianas de aprovechamiento. Coeficiente de rango de biserial r = 0.07 Como 0 ≤ |r| < 0.1, se interpreta como un tamaño del efecto insignificante, no hay una diferencia entre las medianas de aprovechamiento.	p-valor = 0.1713 > α = 0.05 No se encontró suficiente evidencia para concluir que existe diferencia significativa entre las medianas de aprovechamiento. Las mismas medianas de aprovechamiento. Coeficiente de rango de biserial r = 0. 2531344 Como 0.1 ≤ |r| < 0.3, se interpreta como un tamaño del efecto pequeño, hay una diferencia entre las medianas de aprovechamiento, pero no es muy sustancial.	p-valor = 0.01669 < α = 0.05 Se rechaza H01. Si existe diferencia significativa entre las medianas de aprovechamiento. Las medianas de aprovechamiento son diferentes. Coeficiente de rango de biserial r = 0.4999021 Como 0.3 < |r| ≤ 0.5, se interpreta como un tamaño del efecto moderado, la diferencia entre las medianas de aprovechamiento es de magnitud moderada.

Nota: M1 = Mediana de calificaciones, método tradicional (semestres 2016 y 2017), M2 = Mediana de calificaciones, con metodología usando Scratch (primera columna período 2018, segunda columna período 2019, tercera columna periodo 2020-2021).

Fuente: Elaboración propia

Asimismo, se averiguó si los datos recabados tenían un comportamiento normal (análisis de residuales); sin embargo, al aplicar la prueba de Shapiro-Wilk en cada uno de los tres años en que se realizó la intervención, las pruebas demostraron que los datos no

presentaron un comportamiento normal (fila 1, tabla 10). Partiendo de los resultados obtenidos, se recurrió a la prueba no paramétrica Mann-Whitney.

La fila 2 de la tabla 10 presenta un resultado más detallado de la aplicación de esta prueba Mann-Whitney, donde se observa que hasta el tercer año se pudo rechazar H01, por lo que se concluye que la diferencia respecto a las medianas es estadísticamente diferente y mejora cuando se emplea la metodología con Scratch en comparación con la forma tradicional, con un nivel de significancia del 5 %. Estos resultados continuaron siendo mejores con el correr de las intervenciones, como se aprecia en la tabla, donde se observa que el p-valor se fue haciendo cada vez más pequeño desde el año 2018 al 2021.

En concordancia con lo anterior, la medida de tamaño del efecto empleada también fue siendo mejor, aunque solo se está reflejando un efecto moderado. Por tanto, se sugiere el uso del coeficiente empleado cuando la prueba de Mann-Whitney ya indicó una diferencia significativa (Agresti, 2018; Lehmann, 2006).

En segundo lugar, se procedió a analizar con respecto a la cantidad de alumnos que había aprobado y reprobado. La tabla 11 presenta la hipótesis planteada y la evolución de los resultados.

Tabla 11. Evolución de los resultados con base en el índice de aprobación

Prueba Xi cuadrada			
Hipótesis	Resultados 2018	Resultados 2019	Resultados 2020-2021
H02: La cantidad de aprobados y reprobados son independientes del método. H12: La cantidad de aprobados y reprobados no son independientes del método.	p-valor = 0.4501 > α = 0.05 No se encontró suficiente evidencia para concluir que existen diferencias significativas. Resultado: Independiente del método aplicado. Phi = 0.168 V de Cramer = 0.1683 ContCoef = 0.16599 Los tres coeficientes son similares y se encuentran en el rango, 0.1 ≤ Coeficiente < 0.3 Indica una asociación débil o moderada entre las variables. Existe alguna relación entre las variables, pero no es relevante.	p-valor = 0.2026 > α = 0.05 No se encontró suficiente evidencia para concluir que existen diferencias significativas. Resultado: Independiente del método aplicado. phi= 0.202 V de Cramer =0.2020 ContCoef=0.19814 Los tres coeficientes son similares y se encuentran en el rango, 0.1 ≤ Coeficiente < 0.3 Indica una asociación débil o moderada entre las variables. Existe alguna relación entre las variables, pero no es relevante.	p-valor = 0.0206 < α = 0.05 Se rechaza H02. Resultado: Dependiente del método aplicado. phi= 0.281 V de Cramer =0.2809 ContCoef=0.2704096 Los tres coeficientes son similares y se encuentran en el rango, 0.1 ≤ Coeficiente < 0.3 Indica una asociación débil o moderada entre las variables. Pero como se está cercano al extremo derecho 0.3, entonces esta relación podría ser relevante, pero no fuerte.

Fuente: Elaboración propia

Se implementó la prueba no paramétrica Xi cuadrada con los datos ya presentados en la figura 2 para ejecutarla. Esta prueba también sumó a favor de la propuesta, ya que en el tercer año (como se observa en la tabla 11, última columna) se procedió a rechazar H02 en función del p-valor encontrado y al alfa empleado. En consecuencia, la cantidad de aprobados y reprobados dependen del método, lo que favorece el uso de la herramienta Scratch en lugar del modo tradicional.

Estos resultados continuaron siendo mejores con el correr de las intervenciones, como se puede apreciar en la tabla 11, donde el p-valor se fue haciendo cada vez más pequeño del año 2018 al 2021. Asimismo, las medidas de fuerza de dependencia (phi, V de Cramer y el

coeficiente de contingencia) fueron mejores, aunque estas solo reflejan un resultado moderado (Agresti, 2018; Lehmann, 2006).

Por último, se evaluaron los resultados alcanzados por los estudiantes de acuerdo con el nivel de dominio. Los niveles considerados de interés en esta investigación fueron *satisfactorio* y *sobresaliente*. La tabla 12 presenta la hipótesis planteada y la evolución de los resultados.

Tabla 12. Evolución de los resultados en función del nivel de dominio

Prueba Xi Cuadrada			
Hipótesis	Resultados 2018	Resultados 2019	Resultados 2020-2021
H03: La evaluación cualitativa es independiente del método de enseñanza utilizado. H13: La evaluación cualitativa es dependiente del método de enseñanza utilizado.	p-valor = 0.6746 > α = 0.05 No se encontró suficiente evidencia para concluir que existen diferencias significativas. Resultado: La evaluación cualitativa es independiente del método. Phi = 0.11 V de Cramer = 0.1097 ContCoef = 0.109 Los tres coeficientes son similares y se encuentran en el rango, $0.1 \leq$ Coeficiente < 0.3 Esto indica una asociación débil o moderada entre las variables. Existe alguna relación entre las variables, pero no es relevante.	p-valor=0.1689 > α=0.05 No se encontró suficiente evidencia para concluir que existen diferencias significativas. Resultado: La evaluación cualitativa es independiente del método. Phi = 0.208 V de Cramer = 0.208 ContCoef = 0.2036533 Los tres coeficientes son similares y se encuentran en el rango, $0.1 \leq$ Coeficiente < 0.3 Esto indica una asociación débil o moderada entre las variables. Existe alguna relación entre las variables, pero no es relevante.	p-valor = 0.04135 < α = 0.05 Se rechaza H03. Resultado: La evaluación cualitativa es dependiente del método. Phi = 0.243 V de Cramer = 0.2431 ContCoef = 0.2362075 Los tres coeficientes son similares y se encuentran en el rango, $0.1 \leq$ Coeficiente < 0.3 Esto indica una asociación débil o moderada entre las variables. Existe alguna relación entre las variables, pero no es relevante.

Nota: SS = Sobresaliente, SA = Satisfactorio, S = Suficiente, NA = No aprobado.

Fuente: Elaboración propia

Se utilizó en este apartado la prueba no paramétrica de Xi cuadrada con la información presentada en la figura 2 para ejecutarla. De modo similar a los dos conceptos evaluados con anterioridad, los resultados de esta prueba también fueron favorables para la propuesta, ya que con la evidencia presentada en el tercer año se pudo rechazar H03. Por tanto, se concluye

que el empleo de la metodología Scratch marcó una diferencia estadísticamente significativa respecto del modo original. Este indicador también mejoró con el correr de las intervenciones, como se aprecia en la tabla 12, donde el p-valor se fue haciendo cada vez más pequeño del año 2018 al 2021. Además, las medidas de fuerza de dependencia (phi, V de Cramer y el coeficiente de contingencia) fueron mejores. Sin embargo, aunque estas solo reflejan que existe alguna relación entre las variables, no es relevante, pues se tiene un resultado moderado (Agresti, 2018; Lehmann, 2006).

Según la tendencia mostrada, se anticipa que el p-valor continúe siendo menor en los siguientes cursos que se impartan en relación con los indicadores evaluados.

Además de las pruebas anteriores, también se verificó que las calificaciones obtenidas mediante la aplicación de la metodología tienen una mediana de 88 puntos, y en cuanto al nivel de dominio se ubica en el *satisfactorio*. La tabla 13 presenta la hipótesis planteada y la evolución de los resultados.

Tabla 13. Promedio de aprovechamiento con el uso de Scratch

Hipótesis	Prueba	Resultados 2018	Resultados 2019	Resultados 2020 - 2021
H0: Datos normales H1: Datos no normales	Shapiro-Wilk	p-valor = 0.007 < α = 0.05 Rechazo H0, no normalidad	p-valor = 0.000095 < α = 0.05 Rechazo H0, no normalidad	p-valor = 1.589-05 < α = 0.05 Rechazo H0, no normalidad
H04: M2 = 80 H14: M2 > 80	Wilcoxon	p-valor = 0.25 > α = 0.05 No se encontró suficiente evidencia para concluir que existen diferencias significativas, la mediana usando la metodología no es mayor a 80 pts. Coeficiente de rango de biserial r= -0.06984424 Como |r|<0.1, se interpreta como un tamaño del efecto insignificante. Indica que la mediana de aprovechamiento no es mayor de 80 puntos.	p-valor = 0.0125 < α = 0.05 Se rechaza H04, la mediana usando la metodología es mayor a 80 pts. Coeficiente de rango de biserial r= 0.1067457 Como 0.1< |r|<0.3, se interpreta como un tamaño del efecto pequeño. Indica que la mediana de aprovechamiento no es sustancialmente mayor de 80 puntos.	p-valor = 7.488-06 < α = 0.05 Se rechaza H04, la mediana usando la metodología es mayor a 80 pts. Coeficiente de rango de biserial r= = 0.31 Como 0.3 ≤ |r| < 0.5, se interpreta como un tamaño del efecto moderado. Indica la mediana de aprovechamiento, es moderadamente mayor de 80 puntos.

Nota: M2 = Mediana usando la metodología.

Fuente: Elaboración propia

Primero se verificó si los datos tenían una distribución normal (fila 1, tabla 13). Para eso, se utilizó la prueba de Shapiro-Wilk en las calificaciones obtenidas luego de la metodología empleada. En todos los casos se rechazó H0, pues los datos no tienen un comportamiento normal.

Partiendo de los resultados obtenidos de la no normalidad, se empleó la prueba no paramétrica de signos de Wilcoxon para probar la hipótesis planteada (fila 2, tabla 13). Según el p-valor y el alfa empleada, se rechaza H04, es decir, se afirma que la mediana de calificaciones cuando se emplea la metodología con la herramienta Scratch supera los 80 puntos. Esto es ideal para los objetivos del proceso investigativo.

La prueba de Wilcoxon se usó también con los datos encontrados en la intervención. La primera ocasión (2018) no se contó con evidencia suficiente para rechazar H04; sin embargo, sí se logró desde el segundo año de la intervención (2019) en adelante (fila 2, tabla 13). Cabe mencionar que aun cuando se calcula el tamaño de efecto, hay que tener cuidado con la interpretación cuando la prueba no paramétrica no indica una diferencia significativa (Agresti, 2018; Lehmann, 2006). Por último, en cuanto a los resultados de 2020-2021, el efecto de tamaño es moderado, ya que muestra una mediana de aprovechamiento moderadamente mayor de 80 puntos.

Discusión

La metodología que se ha descrito en este trabajo permite constatar la utilidad que tiene el uso de tecnologías como Scratch para lograr aprendizajes significativos en estudiantes de los primeros cursos de programación de computadoras, lo cual, a su vez, facilita la mejora del desempeño académico de los participantes, particularmente en el caso analizado de la LIS en la UADY–UMT.

Investigadores y docentes del área de programación de computadoras coinciden en las dificultades que enfrentan los alumnos de nuevo ingreso de carreras de ingeniería para poder superar con éxito las problemáticas a las que se enfrentan y desarrollar habilidades y aprendizajes complejos requeridos, especialmente aquellos relacionados con la algoritmia y la programación. Sin embargo, gran parte del éxito de la carrera profesional tiene que ver con el adecuado dominio del contenido de estos cursos que se estudian al inicio de la formación. Por tal motivo, en la revisión de la literatura se ha constatado que diferentes autores han planteado una diversidad de abordajes en el proceso de enseñanza, estrategias y metodologías, que se han estado empleando como herramientas de apoyo para el aprendiz en su proceso introductorio a la programación. El estudio que se expone en este documento abona en ese sentido, pues brinda una metodología novedosa que ha sido implementada con éxito en el contexto previamente descrito.

La importancia de mejorar el enfoque empleado en la enseñanza y aprendizaje en cursos introductorios a la programación es evidente, de ahí que existan reportes de investigación como el encontrado en Jiménez-Toledo *et al*. (2019), quienes recopilaron, sintetizaron y categorizaron un gran número de experiencias con estos cursos en la educación superior. En este tipo de resultados se sustenta la presente investigación, la cual busca apoyar en estos procesos.

Como argumenta Insuasti (2016), una de las razones de los resultados poco alentadores en este tipo de cursos puede ser lo complejo de aprender las estructuras y las reglas de sintaxis y semántica de los lenguajes. Por eso, para atender este tipo de situaciones se decidió incluir la herramienta Scratch en la metodología implementada, ya que al no emplear una sintaxis textual específica, puede disminuir la complejidad para su aprendizaje. De hecho, Scratch es una herramienta que se enfoca al ensamblado de bloques, lo cual resulta relevante en la asignatura de Algoritmia, ya que en ella no se pretende enseñar a programar al educando, sino estudiar los fundamentos y estructuras básicas de la programación.

Como explican Malliarakis *et al*. (2014), el aprendizaje tradicional de cursos de fundamentos de computación implica muchos retos, lo cual aumenta considerablemente los índices de deserción (Rodríguez, 2014). En la literatura revisada, diversos autores como Rizvi *et al*. (2011), Muñoz *et al*. (2012), Hernández (2013), Muñoz *et al.* (2017), Hidalgo *et al.* (2021) y Bedoya (2021) dan cuenta de los elevados índices de deserción en este tipo de materias introductorias a la programación en carreras relacionadas con las ciencias de la computación.

Este era un problema que se tenía en la UMT en los primeros cursos en los que se enseñó la materia de Algoritmia, de ahí que se haya decidido implementar una metodología de aprendizaje mediada por Scratch. En tal sentido, los resultados han llegado a ser satisfactorios después de varias aplicaciones y ajustes.

Por otro lado, se coincide con lo que mencionan Jiménez-Toledo *et al.* (2019), quienes clasifican a la herramienta Scratch dentro de la categoría de juegos educativos centrados en la enseñanza de unidades múltiples de aprendizaje. Si bien es cierto que esta fue utilizada inicialmente en Algoritmia para crear pequeñas aplicaciones relacionadas con problemas cotidianos, posteriormente los alumnos fueron avanzando en su dominio hasta llegar a la creación de juegos completos que incluyeron cada una de las estructuras analizadas en la materia. Además, como aporte complementario a su proceso de creación, también anexaron a sus juegos elementos de la cultura maya, lo cual es de gran importancia en el contexto específico del estado de Yucatán, donde se realizó el proceso investigativo.

Considerando lo anterior, se puede afirmar que el lenguaje de programación Scratch es una TIC adecuada para estas actividades educativas, pues también se pudo apreciar la motivación de los estudiantes durante las actividades de laboratorio. Además, se promovió el trabajo colaborativo entre equipos, así como la estimulación del pensamiento algorítmico y la creatividad. En definitiva, los proyectos generados superaron las expectativas del docente

de la asignatura debido a la diversidad de elementos incluidos en los videojuegos creados, como son los personajes, elementos de ambientación, fondos de acompañamiento y sonidos.

Los resultados obtenidos coinciden con el empleo de Scratch que otros docentes han hecho para brindar apoyo a sus estudiantes en cursos introductorios a la programación, entre los cuales se puede mencionar a Rizvi *et al*. (2011) y Muñoz *et al*. (2015, 2017).

Finalmente, con la información obtenida al concluir los experimentos se ha podido constatar que los resultados recabados han ido mejorando progresivamente a lo largo de las tres intervenciones. Esto demuestra que el enfoque empleado marca una diferencia significativa.

Sin embargo, a pesar de lo encontrado hasta ahora durante el proceso investigativo, así como en la revisión de la literatura, se vuelve relevante mencionar las limitaciones encontradas. La herramienta Scratch, por ejemplo, tiene un obstáculo al momento de comunicar ideas, pues los bloques que emplea están previamente configurados con formas que no pueden ser modificadas. Además, estos ocupan espacio tanto horizontal como vertical, lo que restringe la flexibilidad de las posibles combinaciones.

Por otro lado, se sabe que los entornos que usan bloques no dejan ver errores de sintaxis. Sin embargo, inevitablemente el alumno deberá de enfrentarlos, ya que surgirán cuando comience a utilizar otro tipo de lenguajes (Gómez y Martínez, 2021). En particular, se observó que no todas las estructuras que se incluyen en la planeación didáctica se pueden representar en Scratch, lo cual es un gran obstáculo. Asimismo, es relevante señalar que a la fecha solo se cuenta con opiniones de los estudiantes respecto de la percepción que tienen de esta herramienta, lo cual queda pendiente para ser investigado.

Conclusiones

Actualmente, es evidente el esfuerzo de los maestros, investigadores, así como de grupos de investigación que procuran mejorar los procesos de enseñanza y aprendizaje de alumnos que se encuentran en los primeros cursos de programación, ya que un buen desempeño redundará en el éxito de las siguientes asignaturas que deban cursar.

En el caso específico del curso de Algoritmia descrito en la presente investigación, este se impartió en la UMT entre los años 2016 y 2017 a través del modo tradicional. Luego, con la información recabada durante este periodo y con el apoyo de la literatura encontrada, surgió una propuesta metodológica que incluyó el uso de la herramienta Scratch, la cual ha

venido mejorando al término de cada intervención. Actualmente, continúa en funcionamiento y brinda apoyo al estudiante, lo que favorece su rendimiento académico en la asignatura.

En este sentido, se procedió a realizar un análisis de los resultados de las actas de calificaciones finales de los cursos impartidos con papel y lápiz, así como de aquellos en los que fue empleada la metodología mediada por Scratch. Para esto, fueron considerados tres aspectos: el promedio del aprovechamiento académico, el índice de aprobación y el nivel de dominio en lo que respecta a las habilidades del estudiante.

En el primer aspecto evaluado, la mediana de aprovechamiento académico resultó ser mejor utilizando la herramienta tecnológica Scratch en comparación de aquellos cursos impartidos de modo tradicional. En cuanto al índice de aprobación, se obtuvo que el resultado tiene una dependencia respecto al método que se emplea (con o sin el uso de Scratch), pues se consiguió un avance notable cuando se usó la herramienta tecnológica. Por último, se evaluaron los resultados considerando el nivel de dominio, y se halló que estos dependen del método implementado, con un incremento en el porcentaje de estudiantes con nivel *satisfactorio* o *sobresaliente* cuando se usó Scratch.

Finalmente, como una ventaja en la aplicación de esta metodología se procedió a hacer el análisis de la mediana general de aquellos cursos en los que se empleó la herramienta Scratch. El resultado fue satisfactorio de acuerdo con lo establecido por los lineamientos de la UADY, lo cual es conveniente dentro del contexto de este proceso investigativo.

Futuras líneas de investigación

Actualmente, se sigue trabajando en el apoyo a los estudiantes de nuevo ingreso en la materia de Algoritmia desde dos enfoques. Por un lado, atendiendo los resultados encontrados, se sigue empleando Scratch; sin embargo, partiendo del hecho de que puede haber otro tipo de herramientas que brinden una mejor opción al estudiantado no solo para aprobar Algoritmia, sino para ayudarlo en las materias relacionadas de los siguientes semestres, se decidió implementar otro *software* semejante, el cual también es un lenguaje de programación visual basado en el manejo de bloques, denominado Blockly. La diferencia de este se halla es que genera código en otros lenguajes, como JavaScript, Python, PHP, etc., los cuales son relevantes en la licenciatura en Ingeniería de Software, y pueden darle al alumno un panorama más amplio de lo que sigue. Hasta la fecha, se ha trabajado durante un semestre con Blockly, por lo que se espera hacer evaluaciones y generar resultados después

de dos intervenciones más. El objetivo, en un futuro próximo, es verificar con cuál de las dos herramientas se pueden lograr mejores resultados.

Por otro lado, se ha empezado un estudio para averiguar el nivel de pensamiento computacional con el que llegan los estudiantes a la UMT, especialmente en el contenido de algoritmos, con el objetivo de que esta información pueda servir de apoyo en el proceso de las clases.

Referencias

Agresti, A. (2018). *Categorical data analysis* (3rd ed.). Wiley.

Bedoya, A. (2021). *Diagnóstico de los estudiantes de ingeniería en diseño de entretenimiento digital para facilitar el aprendizaje de los fundamentos de programación.* XVI Semana Internacional de Diseño en Palermo.

Cairó, O. (2005). *Metodología de la programación. Algoritmos, diagramas de flujo y programas*. Alfaomega.

Calderón, C. (2020). *Arqueología informática: diseño e implementación de calculadoras mecánicas Facit con Scratch* (trabajo de grado). Universitat Politècnica de València.

Campbell, O. y Atagana, H. (2022). Impact of a Scratch programming intervention on student engagement in a Nigerian polytechnic first-year class: verdict from the observers. *Heliyon, 8*(3).

Cárdenas, J., Puris, A., Novoa, P., Parra, A., Moreno, J. y Benavides, D. (2021). Using Scratch to Improve Learning Programming in College Students: A Positive Experience from a Non-WEIRD Country. *Electronics, 10*(10), 1180.

Cátedra Conceptos de Algoritmos, Datos y Programas, UNLP. (2016). ¿Por qué "pensar algoritmos" es tan importante en Informática? *Revista Institucional de la Facultad de Informática Bit & Byte*, (4).

Colorado, M. (2020). *Compendio y desarrollo de herramientas digitales para apoyar la enseñanza-aprendizaje-evaluación en Ingeniería de Software* (trabajo de grado). Unidad Multidisciplinaria Tizimín, Universidad Autónoma de Yucatán.

Gómez, J., Narváez, L., Rejón, E. y Reyes, J. (2016). *Planeación didáctica de Algoritmia* (manuscrito no publicado). Universidad Autónoma de Yucatán.

Gómez, M. y Martínez, P. (2021). *¿Scratch, Python, o qué? Criterios para elegir un entorno para enseñar a programar a principiantes.* Jornadas Argentinas de Didáctica de las Ciencias de la Computación.

González, C., Montañez, T., Chí, V., Miranda, C. y González, S. (2012). Analysis of subjects with greater difficulty for university students in the area of computer science. I*nternational Journal of Computer Science and Network Security*, *12*(10), 1-6.

Hernández, G. (2013). Creencias docentes y didáctica de la programación de computadoras. *Revista Universitaria, Docencia, Investigación e Innovación, 2*(2), 87-103.

Hernández, R., Fernández, C. y Baptista, P. (2014). *Metodología de la investigación* (6.ª ed.). McGraw Hill.

Hidalgo, C., Bucheli, V., Restrepo, F. y González, F. (2021). 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*, *9*(1), 50-60. https://doi.org/10.17081/invinno.9.1.4185

Insuasti, J. (2016). Problemas de enseñanza y aprendizaje de los fundamentos de programación. *Revista Educación y Desarrollo Social*, *10*(2). https://doi.org/10.18359/reds.1966

Jiménez-Toledo, J. A., Collazos, C. y Revelo-Sánchez, O. (2019). Consideraciones en los procesos de enseñanza-aprendizaje para un primer curso de programación de computadores: una revisión sistemática de la literatura. *TecnoLógicas*, 22, 83-117. https://doi.org/10.22430/22565337.1520

Joyanes, L. (2003). *Fundamentos de programación: algoritmos y estructura de datos*. McGraw-Hill.

Lehmann, E. (2006). *Nonparametrics: Statistical Methods Based on Ranks*. Springer.

López, J., Hernández, C. y Farran Leiva, Y. (2011). Una plataforma de evaluación automática con una metodología efectiva para la enseñanza/aprendizaje en programación de computadores. *Ingeniare. Revista Chilena de Ingeniería*, *19*(2), 265-277. https://doi.org/10.4067/S0718-33052011000200011

Malliarakis, C., Satratzemi, M. y Xinogalos, S. (2014). Educational Games for Teaching Computer Programming. En C. Karagiannidis, P. Politis and I. Karasavvidis (eds.), *Research on e-Learning and ICT in Education* (pp. 87-98). Springer. https://doi.org/10.1007/978-1-4614-6501-0_7

MIT Scratch Team (s. f.). *Scratch-Imagine, Program, Share.* https://scratch.mit.edu/

Monjelat, N., Cenacchi, M. y San Martín, P. (2018). ¿Programación para todos? Herramientas y accesibilidad: un estudio de caso. *Revista Latinoamericana de*

Educación Inclusiva, *12*(1), 213-227. https://doi.org/10.4067/S0718-73782018000100213

Muñoz, R., Barcelos, S., Villarroel, R., Barría, M., Becerra, C., Noel, R. y Silveira, F. (2015). *Uso de Scratch y Lego Mindstorms como apoyo a la docencia en Fundamentos de Programación.* Actas de las XXI Jornadas de la Enseñanza Universitaria de la Informática, Andorra La Vella.

Muñoz, R., Barcelos, T., Villarroel, R. y Silveira, F. (2017). Using Scratch to Support Programming Fundamentals. *Journal on Computational Thinking (JCThink)*, *1*(1). https://doi.org/10.14210/ijcthink.v1.n1.p68

Muñoz, R., Barría, M., Nöel, R., Providel, E. y Quiroz, P. (2012). *Determinando las dificultades en el aprendizaje de la primera asignatura de programación en estudiantes de ingeniería civil informática*. Nuevas Ideas en Informática Educativa. Memorias del XVII Congreso Internacional de Informática Educativa, TISE, Santiago, Chile.

Pérez-Narváez, H., Roig-Vila, R. y Jaramillo–Naranjo, L. (2020). Uso de SCRATCH en el aprendizaje de programación en educación superior. *Revista Cátedra*, *3*(1), 28-45. https://doi.org/10.29166/10.29166/catedra.v3i1.2006

Pinales, F. y Velázquez, C. (2014). *Algoritmos resueltos con diagramas de flujo y pseudocódigo.* Universidad Autónoma de Aguascalientes.

Puig, L. (2009). Historias de al-Khwarizmi (3.ª entrega): orígenes del álgebra. S*uma: Revista sobre Enseñanza y Aprendizaje de las Matemáticas*, *60*, 103-108.

Rizvi, M., Humphries, T., Major, D., Jones, M. y Lauzun, H. (2010). A CS0 course using Scratch. *The Journal of Computing Sciences in Colleges*, *26*(3), 19–27.

Robins, A., Rountree, J. y Rountree, N. (2003). Learning and Teaching Programming: A Review and Discussion. *Computer Science Education*, *13*(2), 137-172. https://doi.org/10.1076/csed.13.2.137.14200

Rodríguez, G. (2014). Enseñanza de la programación de computadoras para principiantes: un contexto histórico. *Inventum*, *9*(17). https://doi.org/10.26620/uniminuto.inventum.9.17.2014.51-61

Solomon, C., Harvey, B., Kahn, K., Lieberman, H., Miller, M. L., Minsky, M., Papert, A. y Silverman, B. (2020). History of logo. *Proceedings of the ACM Programming Languages*, *4*, 1-66. https://doi.org/10.1145/3386329

Universidad Autónoma de Yucatán (2012). *Modelo educativo para la formación integral*. (manuscrito no publicado). Universidad Autónoma de Yucatán.

Velázquez, J., Paredes, M., Cavero, S. y Palacios, *D.* (2023). *Mejora de una asignatura para la formación del profesorado en programación basada en bloques.* Actas de las XXIX Jornadas sobre Enseñanza Universitaria de la Informática, Granada, España.

Rol de Contribución	Autor (es)
Conceptualización	Lizzie Edmea Narváez Díaz
Metodología	Lizzie Edmea Narváez Díaz
Validación	Manuel Escalante Torres
Análisis Formal	Manuel Escalante Torres
Investigación	Maximiliano Canché Euán
Recursos	Carlos Andrés Miranda Palma
Escritura - Preparación del borrador original	Carlos Andrés Miranda Palma
Escritura - Revisión y edición	Maximiliano Canché Euán
Visualización	Cinhtia Maribel González Segura
Supervisión	Lizzie Edmea Narváez Díaz
Administración de Proyectos	Cinhtia Maribel González Segura
Adquisición de fondos	Lizzie Edmea Narváez Díaz