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 Andrés 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 perceived by university students, teachers, and researchers as a difficult process in the computing area. Therefore, the objective of this paper is to present an analysis of the results of the application of a constructionist teaching and learning methodology mediated by the programming environment Scratch. The research was conducted in the Algorithmics subject, which is taught in the Undergraduate Degree in Software Engineering offered by the Autonomous University of Yucatan (Uady) at the Multidisciplinary Unit Tizimín, located in Tizimín, Yucatán, México. The study used the quantitative paradigm under an experimental perspective with a longitudinal approach, and worked with a non-probabilistic sample, which consisted of all the students who took the aforementioned subject. The results indicate that the implemented teaching and learning methodology, carried out through the use of Scratch,

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 of the subject matter. 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 Algorithmics subject.

Keywords: Algorithms, Teaching and 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

Introduction

The word *algorithm* emerged in the 9th century and was proposed by the Arab mathematician Al-Khwarizmi (Juarismi or Khorezmi), who developed much of his career around the year 800 A.D., working particularly with algebra and astronomy. His contribution to solving equations and the treatise on numbers *Algoritmi de numero indorum* places him as the oldest reference for the word *algorithm* , and over the years this word has become one of the foundations of computing (Cátedra Conceptos de Algorithms, Data and Programs, UNLP, 2016; Puig, 2009) .

"Algorithm" can be defined as a sequence of steps, procedures or actions that allow us to achieve a result or solve a problem. In other words, it is an ordered set of rules to address a certain type of situations or problems and a manner to propose their solution (Cairó, 2005; Joyanes, 2003) . Finally, an algorithm must be able to be easily translated into a programming language (Pinales and Velázquez, 2014).

Despite the years that have passed since the emergence of this concept, and despite all the studies that have been carried out in the area of computing, it is a fact that in the first university courses there are problems and difficulties for students when they start to study the fundamentals of programming (Colorado, 2020; González *et al* ., 2012) .

In addition to the results and the opinions of the students, various teachers and researchers agree on how complex it is to teach and learn programming and its foundations. Novice programmers manifest a wide variety of difficulties and deficiencies due to the complexity involved in learning algorithmic concepts (Insuasti, 2016; Robins *et al* ., 2003) . In this sense, Winslow (1996, cited by Robins *et al* ., 2003) states that at least 10 years of experience are required to go from a novice to an expert programmer level, which indicates that when a student finishes his career he /she still has a long way to go to become an experienced programmer.

The initial programming classes that college students take should be captivating and engaging to spark their interest in computer science. However, evidence suggests that these classes are far from engaging for most students compared to other science courses (Campbell and Atagana , 2022).

Additionally, acquiring programming skills is a complicated activity, since, first, you must understand the problem, then create an algorithm that outlines the solution and, finally, translate it into a programming language. This last activity is a repetitive process of four stages: coding, the compilation process, testing the code and, if it fails, debugging it (López

et al ., 2011). For this reason, Soloway and Spohrer (1989, cited by Insuasti , 2016) state that these tasks can be very difficult for a person.

On the other hand, in the area of computer science, the curriculum in the first semesters focuses on mathematics and introductory programming courses, which are related to the high dropout and failure rates. This aspect is an important point to consider because basic programming skills are developed in these courses (Rodríguez, 2014) .

In this regard, Muñoz *et al* . (2012) point out that many approaches, methodologies and teaching tools have been proposed to help students in this learning process; however, to date, there does not appear to be a satisfactory path forward. According to Jiménez-Toledo et al. (2019), it is important that the student has problem-solving skills that he/she can apply and at the same time create his/her own cognitive, logical, mathematical and algorithmic models. However, although it may seem that the application of basic programming concepts is a relatively simple process, various authors—including those in this study—agree that these basic principles are often complicated for students, a situation considered as a product in which various factors converge. In this sense, Insuasti (2016) complements the previous ideas by indicating that some of the reasons why many students do not achieve the expected results may be the difficulty of mastering the programming language, the basic principles of the area, limited background, and the lack of good teacher didactics, to mention the most important.

It can be concluded, therefore, that acquiring programming knowledge and training expert professionals in this area presents various challenges for both teachers and students, especially when teaching is carried out with the traditional approach, that is, with paper and pencil (Malliarakis *et al* ., 2014) . This form of knowledge transmission generates high academic dropout rates in programming fundamentals courses (Rodríguez, 2014) . Consequently, the following question has been formulated: *How to reduce dropout levels in introductory computer programming courses?*

In response to the previous lines, the Autonomous University of Yucatán (UADY) has the Software Engineering program (LIS), which began to be offered at the Tizimín Multidisciplinary Unit (UMT) from the August-December 2016 semester. As part of the study plan, the subject named *Algorithms* is included, in order to develop algorithm design skills for solving problems in the computational area. In this course the student starts with a real-life problem, which he/she must analyze and understand and then build an algorithm that allows solving the initial problem. This process helps him/her in the development of orderly

and logical thinking, as well as in the acquisition of skills to create diverse ways to solve a problem, so that he/she can later translate his/her solution into a programming language (Gómez et al., 2016). This subject does not aim to introduce the student to the world of programming, but rather studies those aspects that precede this task.

On the first two occasions in which the Algorithms course was taught in the Software Engineering program at the UADY-UMT, the passing rates were not as expected. The percentages obtained are detailed in table 1.

Table 1. Student performance at the beginning of Software Engineering

Semester	Satisfactory/Outstanding	Sufficient/Not approved
August-December 2016	58.3%	41.7%
August-December 2017	46.2%	53.8%
Average	52.25%	47.75%

Source: own development

Based on the data in Table 1, a lack of student knowledge of the topics is evident, which can cause problems in subsequent courses of the curriculum in which a solid mastery of the skills derived from programming principles is required.

Due to the above, the teachers involved in such a situation decided to implement a teaching/learning methodology developed expressly to support students in the introductory processes to computer programming, which was mediated by the use of the block-based programming tool called *Scratch*. According to Gómez and Martínez (2021), in block-based programming the validity of the generated combinations is represented by the blocks themselves, that is, everything is in visual form, so that the representation of the syntax is explicitly eliminated, which is of great importance for the subject. Thus, the main goal of this study was to analyze with statistical methods whether it is possible to raise the academic performance of students in the subject *Algorithms* after the implementation of the teaching/ learning methodology previously mentioned.

Problems in learning programming

The difficulty of learning programming has been addressed in various research works described in the literature, where a variety of solution strategies have been proposed. Some of them are presented below.

Rizvi et al. (2011) describe a study that addresses the worrying retention of first-semester students in computer science programs at Norfolk State University, United States. In their work, these authors propose to design and teach a previous course, as a preparatory course to students, before they take their first introductory course to programming, called *CS1*, which is part of the mandatory academic load in these careers. In the previous course, called *CS0*, ICT-assisted teaching was implemented. The proposal consisted of using Scratch as a tool to learn the basic concepts of programming. This was implemented because, at Norfolk State University, in the evaluated period from 2005 to 2007, only 42% of students had passed the CS1 course, which clearly represents a warning that requires actions that favor the retention of students. The authors used this strategy to address retention problems, and it was found that 66% of the students who took the CS0 course managed to pass CS1 compared to 44% the previous year, when the students were enrolled directly in the CS1 course.

Similarly, Muñoz *et al*. (2012) describe the case of the Civil Engineering program with computer applications at the University of Valparaíso, Chile, where there was a high dropout rate in the first years of the degree. In that program, one of the subjects with the highest failure rate was Programming Fundamentals with 32.4%, followed by Calculus and Elementary Algebra, results that coincide with the findings of other Mexican authors (Colorado, 2020; González *et al* ., 2012). In that Chilean university, with the purpose of supporting students, a study was carried out using surveys to find the topics that were difficult for them during the learning process, which were basically two: cyclical structures and management of arrays of two or more dimensions.

Based on the high dropout rate reported by Chilean universities, a study was carried out by the University of Valparaíso in the Computer Engineering program of the School of Computer Engineering, given that the same dropout situation was experienced. Specifically, the highest dropout rate was in the fourth semester of the degree, specifically in Programming Fundamentals, one of the subjects with the highest number of failed students. Students, in addition to taking the aforementioned subject, had to take two related courses called Programming I and Programming II. A small group enrolled in this last subject compared to those who started (almost 30% less) and the retention rate did not exceed 50% in the first

three years. To support students in passing these courses, strategies were implemented based on the use of tools such as Scratch and Lego Mindstorms (Muñoz *et al* ., 2015, 2017) .

In Colombia, in the Systems Engineering program at the Mariana University, it was considered that the introductory programming subjects are a determining factor for the continuity of students in the program, since dropout from the course predominates. The results suggest that this situation is caused by demotivation and frustration, which is reflected in the abandonment of the career or in the increase in the time spent (Hernández, 2013) .

In the Systems Engineering program at the Universidad del Valle in Colombia, the subject Introduction to Programming is taught (labeled CS1). In this course it is essential that students develop essential skills for solving complex computational problems through the acquisition of skills that allow them to use a programming language. It was verified that at the end of the course few students completed it satisfactorily to continue with the next subject (CS2). In addition, low grades, excessive absences and a high dropout rate were observed. To support its students, the university proposed designing collaboration strategies and use of ICT tools that would result in better learning. Over time, these strategies raised the students' grades, as well as their interpersonal skills, which encouraged them to improve their programming courses. (Hidalgo *et al* ., 2021) .

Similarly, Bedoya (2021) points out that, at the Universidad Pontificia Bolivariana, in Medellín, Colombia, there are problems of low performance and student dropout when courses related to programming fundamentals are taught in several engineering careers (not only in those expressly linked to computing), so a diagnosis was carried out on the students to provide them with support in their academic process.

Studies that support the proposed methodology

With the idea of employing a new and effective teaching and learning methodology, the use of the Scratch tool was considered. According to Jiménez-Toledo *et al.* (2019), Scratch is a tool classified in the category of *Educational Games focused on teaching multiple learning units*. In this category, with the support of ludic, students are assisted by encouraging their participation with activities that promote enjoyment and creativity in the teaching process. Scratch is a block-based programming language with a visual interface, which helps the student to graphically see the algorithms that he/she designs. Consequently, he/she can understand them more easily, since the graphical and executable visualization is usually

easier to analyze than creating the algorithm with paper and pencil, as occurs with the traditional method of teaching the subject.

In addition to Scratch, in this same category of tools there are others such as the following: JKarel Robot, Blockly, Robot Karel20, Greenfoot, APP Inventor, Alice, Lego, Turingal (Jiménez-Toledo et al., 2019) and *Logo*, project pioneer in the area, developed by Seymour Papert and collaborators. The latter was initially designed to introduce logical-mathematical concepts to children using a programming environment constructively, and was subsequently oriented towards teaching basic programming concepts (Solomon et al., 2020). The above was a key element that was taken into account in the design of our methodology, for which various studies focused on solving the problem described were reviewed.

In this regard, Rodríguez (2014) states that in the area of programming the use of languages such as Python is being novel, as well as the use of integrated development environments oriented to the use of ludic elements, such as Jiménez-Toledo et al. (2019) later proposed. Rodríguez (2014) argues that in order to reduce dropout in programming subjects, various methodologies have currently been created, which include playful activities as a pillar of learning. In this sense, three proposals of interest can be mentioned:

- Creations through video games.
- Paradigm-oriented instructional development.
- Instructional development focused on particular topics.

Regarding the classification called *Creations through video games*, the projects Greenfoot from the University of Kent, Alice from Carnegie Mellon University, and Scratch from the Massachusetts Institute of Technology stand out (of great interest for this research process). They are tools that can help reduce the dropout rate in programming subjects (Rodríguez, 2014) .

Considering the characteristics of Scratch and its compatibility with the skills to be developed in the Algorithms subject, this tool was included in the design of the methodology that was implemented. Scratch has an environment aimed at creating interactive projects, animations, and games using blocks. In addition, it is free software, it is a cutting-edge programming language (Calderón, 2020), and it has the support of the institution that develops it. It also has a large global user community, as well as discussion and technical support forums, a wiki and a website that provide support.

Scratch can be defined as a visual programming language for learning programming concepts and creating interactive projects such as animations, games, and stories. In addition, it does not have specific syntax (like traditional languages), but is based on drag and drop blocks, which have the code integrated. This makes learning easier and brings it closer to people with little or no programming experience. Scratch encourages creativity, systematic reasoning, and promotes work and collaboration among its users (MIT Scratch Team, sf; Monjelat *et al* ., 2018) .

At the State University of Milagro, Ecuador, despite the fact that various teaching methodologies had been implemented, the historical passing rate in programming fundamentals courses remained below 43%. Consequently, the university decided to address the problem through the use of visual programming languages to teach in a more practical and didactic way (particularly Scratch was used). At the end of the tests, the experimental group showed a pass rate four times higher than the control group, and although the results are incipient, it is considered that the university is on the right path to provide students with the support they require to pass. Furthermore, this strategy was attractive to the majority of students (Cárdenas *et al* ., 2021).

Additionally, a group of teachers from the Rey Juan Carlos University (Madrid, Spain) thinks that support should be provided to students from the pre-university level. Therefore, they suggest that pre-university teachers must first be trained to teach courses related to computer programming. To do that, they created a teacher training master's degree focused on digital competence and programming, which prepares the teacher in related topics through the use of tools such as ScratchJr, Scratch and App Inventor in order to help students before their entry to the University. Being a recent study, there are still no conclusive results; However, the importance of tools like Scratch to support students is highlighted (Velázquez *et al* ., 2023).

A group of researchers, after applying the Scratch tool in an introductory programming course, concluded that the use of this resource is based on a constructionist process that favorably involves computer science students in the first years, especially those without prior programming experience (Campbell and Atagana, 2022) .

As a complement to the literature that has been presented in this section, the use of the Scratch tool at the Central University of Ecuador is described in the work of Pérez-Narváez *et al.* (2020), specifically in the Computer Science degree for the Development of Computational Thinking. Although the use of this tool is not expressly implemented in

programming fundamentals, it focuses on the development of various computational topics, such as variables, sequence, use of operators, among others, which supports the development of skills related to computational thinking.

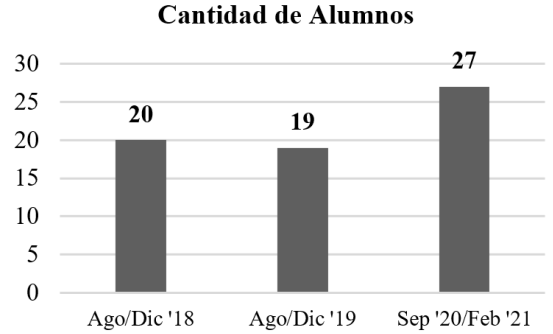
For the reasons previously established, and according to the reviewed literature, the Scratch language was integrated into the methodology detailed in this work, in order to improve the learning results of the students of the Algorithms subject in the Software Engineering career. This work documents the research process carried out in order to support the teaching and learning of Algorithms at our institution.

Method

The context of this study was the Tizimín Multidisciplinary Unit, which belongs to the Autonomous University of Yucatán, Mexico. During the semesters January-May 2022 and August-December 2022, the enrollment of this academic unit was 454 students, according to internal data from the institution, distributed across the four bachelor's degrees offered: Public Accounting, Nursing, Education and Software Engineering.

The research was experimental, with a quantitative and longitudinal approach (Hernández *et al* ., 2014). This was carried out with the participation of students from the first semester of the aforementioned degree, who come from the high schools of Tizimín, as well as from other neighboring states, although in smaller numbers. The study documents the work that was carried out with three generations of students, who were enrolled in the school periods August-December 2018-2019 and September 2020-February 2021. This last period had to be adequate due to the covid-19 health crisis. The total number of students enrolled during these semesters was 66, distributed as shown in Figure 1.

Figure 1. Participants distributed by semesters



Source: own development

Considering the size of the population, it was decided not to apply a sampling method. All students registered in the Algorithms subject from the first semester of the Software Engineering degree participated during the aforementioned academic periods.

Design of the proposed methodology

Specific data was collected from the students of the Algorithms subject during several school years, for which the academic records of the three groups of students with whom the methodology was implemented were required. The records were from the 2018, 2019 and 2020-2021 academic years. Additionally, the academic records of the students from previous semesters were analyzed, that is, 2016 and 2017. These data were collected to make comparisons between teaching methods. The academic records were obtained from the UADY through its Institutional School Information and Control System (SICEI).

Next, the methodology used with the students taking the Algorithms subject is presented, which was divided into four phases:

Phase 1

Before starting the formal Algorithms course, a diagnostic evaluation was applied to the students, which allowed us to obtain information related to mathematics topics, algorithms and general data. The evaluation included topics identified as priorities for the development of algorithms, which were: *use of operators, calculation of percentages, simple operations with an unknown variable and problem decomposition*. These specific topics were suggested as they were identified as essential for developing programming-related skills.

Subsequently, a preparatory course was taught to the students in order to balance their knowledge related to mathematics and programming. This course was carried out with some sessions taught in the classroom and others in the computer laboratory. At the end, a performance test was applied and the data obtained were analyzed (Table 2).

Table 2. Phase 1. Diagnosis applied before the Algorithms course

1. Obtaining general data	
2. Application of a diagnostic test	
Course	Take the course in the classroom and in the laboratory Final Performance Test Application
Final process	Analysis of the final performance test Identification of main errors

Source: own development

Phase 2

In this phase, two learning processes were carried out in parallel: on the one hand, and based on the curricular plan of the degree, the Algorithms course was taught in the classroom; The second process consisted of taking a practical workshop in the laboratory with the support of the Scratch tool. With this workshop, the topics covered in the classroom were reinforced, since the students were able to code previously solved exercises using the graphical programming environment offered by Scratch. Thus, each algorithm designed as a flowchart was coded. Table 3 illustrates the detailed process of the actions that were carried out in this second phase.

Table 3. Phase 2. Activities implemented during the Algorithms course

1. Activities in the classroom	
1	Explain the content
2	Clarify the problem description
3	Analyze the problem
4	Develop the algorithm
2. Activities in the laboratory	
5	Translate the algorithm into a program
6	Run the program Check for errors Yes: go to step 3 No: continue
7	Analyze the program
8	Clarify doubts
9	Assessment

Source: own development

Phase 3

Students were distributed into teams to develop a final project through the use of the Scratch language, using the project-based learning approach. This activity included the use of all the structures studied during the intervention process. The teacher decided when to start this phase according to the progress of the group. Table 4 shows the details of the activities of this phase.

Table 4. Phase 3. Final project development

1. Team building	
2. Work dynamics	
Yo	Determine the project Design the narrative Search and collect information
II	Determine activities Build elements
III	Codify Document the program Deliver the project Assessment

Source: own development

Phase 4

In this phase all the results achieved from the previous phases were analyzed. It is worth mentioning that this methodology was applied again in another academic period of the Algorithms subject, so at the end of all phases the necessary adjustments were made to use it in the next course delivery (table 5).

Table 5. Phase 4. Evaluation of results

1. Evaluate data obtained
2. Make adjustments for the next deployment

Source: own development

Description of the data analysis strategy

The strategy used to perform the analysis of the data collected was carried out as detailed below. First, descriptive statistics were applied to the data collected, both from those courses in which no methodology was implemented (2016 and 2017) and those in which the proposed methodology was used (2018, 2019 and 2020-2021). Three aspects were taken into account: grade average, number of approved students and level of proficiency. As a result of these tests, in all evaluations the use of the methodology gave a differentiating advantage. To prove that this difference was significant, inferential statistics were applied to the aforementioned data, using tests detailed below.

The statistical software named RStudio in version R-4.2.3 (2023) was used in all evaluations, using a significance level of 5% ($\propto = 0.05$). This value is suitable for data analysis, allowing you to balance the risk of making type I and type II errors.

The following inferential statistical tests were carried out:

- Regarding the first aspect evaluated (grade average), it was first verified whether the data had a normal distribution, in order to decide the type of statistics to use. The result was a non-normality and it was decided to use the Mann-Whitney test, which is non-parametric statistics and is used to compare two independent groups and determine if there are significant differences between the distributions of two samples, using the medians instead of the averages. To complement the test, effect size measures were calculated. In this case, the r biserial rank coefficient was used. The interpretation of this coefficient is recommended when the Mann-Whitney test already indicated a significant difference.
- To evaluate the number of approved students, and considering the sample size, the Chi-square test was applied. Regarding the coefficients that help quantify a strength of association between the variables, the Phi coefficient, Cramer's V and the contingency coefficient were found.
- Regarding the level of mastery, and considering the sample size, the Chi-square test was applied. Regarding the coefficients that help quantify a strength of association between the variables, the Phi coefficient, Cramer's V and the contingency coefficient were found.
- In addition, we proceeded to check if the average was higher than 80 points. First, it was verified whether the data followed a normal behavior in order to decide the type of statistics to use. The result was a non-normality and it was decided to use the Wilcoxon test. To complement it, effect size measures were calculated, in this case using the r biserial rank coefficient.

Results

The application of the described methodology was carried out with the purpose of verifying the improvement of the academic achievements of the students in the Algorithms course. In this sense, and as described in the previous section, the examination records recovered from SICEI were the instrument through which the data could be obtained. These data were subsequently analyzed to verify the achievement of the proposed objective.

The report cards were obtained at the end of the aforementioned school semesters and the results obtained when using this methodology were evaluated. In addition, the data corresponding to the previous semesters that were taught in a traditional way were taken back in order to make a comparative analysis. The above is detailed in table 6. This information was collected during the five courses involved in the study.

Table 6. Analyzed semesters of the Algorithm subject

Course	Semester	Strategy
1	August-December 2016	Traditional
2	August-December 2017	Traditional
3	August-December 2018	With methodology using Scratch
4	August-December 2019	With methodology using Scratch
5	September 2020-February 2021	With methodology using Scratch

Source: own development

During the semesters in which the Algorithms subject was taught, the same teacher was in charge of all the sessions in the various school periods. Likewise, the didactic planning was developed in 2016 and since that moment the same topics have been followed in each school period. Finally, it is important to mention that students have diverse educational backgrounds due to the wide variety of systems from which they come (seven in total), data obtained through a brief survey conducted at the beginning of the courses.

The data collected was evaluated with the purpose of determining if the methodology used was differentiating. To do that, a comparative analysis was carried out between the data collected in the courses taught in a traditional way and those in which the Scratch tool was used, which was specified based on three variables:

- Average, traditional mode - use of the Scratch tool.
- Number of approved and failed students.

- Mastery level

The level of mastery is defined as those characteristics that describe the degree of development of the student's competencies in a course, which reflects the result achieved in a subject in two ways: quantitative (0 to 100) and qualitative (outstanding, satisfactory, sufficient and not accredited). This is recorded in the examination report (table 7) (UADY, 2012).

Table 7. Mastery levels at UADY

Mastery levels	
Score	Category
90 – 100	Outstanding (SS)
80 – 89	Satisfactory (SA)
70 – 79	Enough (S)
0 - 69	Not accredited (NA)

Source: UADY (2012)

Regarding the first aspect evaluated, average of the five semesters (traditional mode-use of Scratch), the results can be seen in table 8. One column contains the medians of the two groups, given that the median plays a fundamental role in inferential analysis of data; Furthermore, there is an increase of 9.37 points in the average score and 8 points in the median score in the groups where the proposed methodology was used.

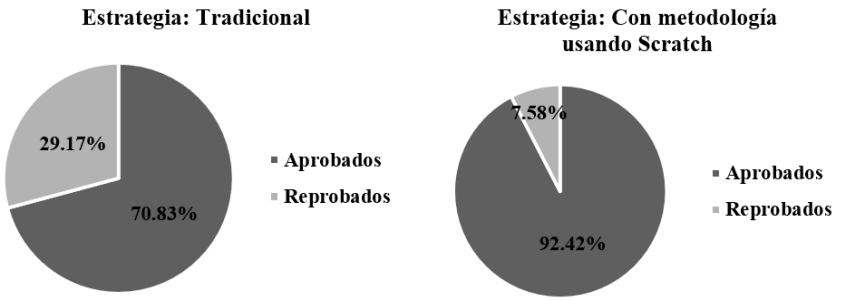
Table 8. Final achievement average in Algorithms 2016 to 2021

Semesters	Strategy	Average	Median
2016 and 2017	Traditional	75.75	80
2018, 2019 and 2020–2021	With methodology	85.12	88

Source: own development

Regarding the second aspect, that is, the evaluation of the number of both passed and failed students, Figure 2 reflects the results generated from all courses using percentages. It is worth mentioning that the number of students evaluated with the traditional method was 24 and the number of students with the methodology mediated with Scratch was 66. In a similar way to the evaluation of the average and median, in this second aspect an increase was also generated in favor of the proposed methodology (21.59 %). Consequently, the number of failed students is lower in the courses where the methodology was applied compared to those that were taught traditionally.

Figure 2. Evaluation of the number of approved and failed students



Source: own development

Table 9 shows the results of the last aspect evaluated (level of mastery), where an increase in favor of the proposed methodology is appreciated (24.62%).

Table 9. Taking into account the level of mastery in Algorithms 2016 to 2021

Strategy	Percentage at satisfactory or outstanding level
Traditional	54.17%
With methodology using Scratch	78.79%

Source: own development

As we can see, the methodology implemented with Scratch managed to increase the performance of the participants; However, this was not enough for the study, since in order to verify that the results found were statistically significant, other tests were carried out using the RStudio software. The statistics used are detailed in the following paragraphs.

Regarding the first aspect examined, it was verified whether the median of the students' grades when using the proposed methodology had a statistically significant difference compared to using the traditional method. Thus, the average was changed to the median as a result of the data having a non-normal distribution. Table 10 presents the proposed hypothesis, as well as the evolution of the results.

Table 10. Evolution of the results of the average analysis

Hypothesis	Proof	2018 Results	2019 Results	2020 – 2021 Results
H0: Normal data H1: Non-normal data	Shapiro-Wilk	p-value = 0.000589 < α = 0.05 H0 rejection, not normality	p-value = 0.0000221 < α = 0.05 H0 rejection, not normality	p-value = 1.849-06 < α = 0.05 H0 rejection, not normality
H01: M1 = M2 H11: M1 ≠ M2	Mann-Whitney	p-value = 0.6623 > α = 0.05 Not enough evidence was found to conclude that there is a significant difference between the medians of achievement. Equal medians of achievement. r biserial rank coefficient = 0.07 Since 0 ≤ |r| < 0.1, there is an insignificant effect size. There is no difference between the medians of achievement.	p-value = 0.1713 > α = 0.05 Not enough evidence was found to conclude that there is a significant difference between the medians of achievement. Equal medians of achievement. r biserial rank coefficient = 0. 2531344 Since 0.1 ≤ |r| < 0.3, there is a small effect size. There is a difference between the medians of achievement, but it is not very substantial.	p-value = 0.01669 < α = 0.05 H01 is rejected. There is a significant difference between the median achievements. The medians of achievement are different. r biserial rank coefficient = 0.4999021 As 0.3 < |r| ≤ 0.5, there is a moderate effect size. The difference between the medians of achievement is of moderate magnitude.

Note : M1 = Median of scores, traditional method (2016 and 2017 semesters), M2 = Median of scores, with methodology using Scratch (first column period 2018, second column period 2019, third column period 2020-2021).

Source: own development

When applying the Shapiro-Wilk test in each of the three years in which the intervention was carried out, the tests showed that the data did not show normal behavior (row 1, table 10). Based on the results obtained, the non-parametric Mann-Whitney test was used.

Row 2 of table 10 presents a more detailed result of the application of the Mann-Whitney test, where it is observed that up to the third year H01 could be rejected, so it is concluded that the difference with respect to the medians is statistically different and

improves when the Scratch methodology is used compared to the traditional way, with a significance level of 5%. These results were better with the interventions, as seen in the table, which shows that the p-value became increasingly smaller from 2018 to 2021.

In accordance with the above, the effect size measure used was also better, although only a moderate effect was reflected. Therefore, the use of the coefficient used is suggested when the Mann-Whitney test indicates a significant difference (Agresti, 2018; Lehmann, 2006).

Secondly, the data was analyzed regarding the number of students who passed and failed. Table 11 presents the proposed hypothesis and the evolution of the results.

Table 11. Evolution of results based on the approval rate

Chi-square test			
Hypothesis	2018 Results	2019 Results	2020-2021 Results
H02: The number of approved and failed students are independent of the method. H12: The number of approved and failed students are not independent of the method.	p-value = 0.4501 > α = 0.05 Not enough evidence was found to conclude that there are significant differences. Result: The number of approved and failed students are independent of the method applied. Phi = 0.168 Cramer's V = 0.1683 ContCoef = 0.16599 The three coefficients are similar and are in the range, 0.1 ≤ Coefficient < 0.3 There is a weak or moderate association between the variables. There is some relationship between the variables, but it is not relevant.	p-value = 0.2026 > α = 0.05 Not enough evidence was found to conclude that there are significant differences. Result: The number of approved and failed students are independent of the method applied. phi= 0.202 Cramer's V =0.2020 ContCoeff=0.19814 The three coefficients are similar and are in the range, 0.1 ≤ Coefficient < 0.3 There is a weak or moderate association between the variables. There is some relationship between the variables, but it is not significant.	p-value = 0.0206 < α = 0.05 H02 is rejected. Result: The number of approved and failed students are not independent of the method. phi= 0.281 Cramer's V =0.2809 ContCoeff=0.2704096 The three coefficients are similar and are in the range, 0.1 ≤ Coefficient < 0.3 There is a weak or moderate association between the variables. But since it is close to the upper bound 0.3, then this relationship could be appreciable, but not strong.

Source: own development

The non-parametric Chi-square test was performed with the data presented in Figure 2. This test also added evidence in favor of the proposed methodology, since in the third year (as seen in table 11, last column) H02 was rejected based on the p-value found and the alpha used. Consequently, the number of approved and failed students depends on the method, which favors the use of the Scratch tool instead of the traditional way.

These results continued to be better as the interventions continued, as can be seen in table 11, where the p-value became increasingly smaller from 2018 to 2021. Likewise, the measures of strength of dependence (phi, Cramer's V and the contingency coefficient) were better, although these only reflect a moderate result (Agresti, 2018; Lehmann, 2006).

Finally, the results achieved by the students were evaluated according to the level of mastery. The levels of interest considered in this research were *satisfactory* and *outstanding*. Table 12 presents the proposed hypothesis and the evolution of the results.

Table 12. Evolution of results depending on the level of mastery

Chi-Square Test			
Hypothesis	2018 Results	2019 Results	2020-2021 Results
H03: Qualitative evaluation is independent of the teaching method. H13: Qualitative evaluation is dependent on the teaching method.	p-value = 0.6746 > α = 0.05 Not enough evidence was found to conclude that there are significant differences. Result: The qualitative evaluation is independent of the method. Phi = 0.11 Cramer's V = 0.1097 ContCoef = 0.109 The three coefficients are similar and are in the range, 0.1 ≤ Coefficient < 0.3 This indicates a weak or moderate association between the variables. There is some relationship between the variables, but it is not relevant.	p-value=0.1689 > α=0.05 Not enough evidence was found to conclude that there are significant differences. Result: The qualitative evaluation is independent of the method. Phi = 0.208 Cramer's V = 0.208 ContCoeff = 0.2036533 The three coefficients are similar and are in the range, 0.1 ≤ Coefficient < 0.3 This indicates a weak or moderate association between the variables. There is some relationship between the variables, but it is not relevant.	p-value = 0.04135 < α = 0.05 H03 is rejected. Result: The qualitative evaluation is dependent on the method. Phi = 0.243 Cramer's V = 0.2431 ContCoef = 0.2362075 The three coefficients are similar and are in the range, 0.1 ≤ Coefficient < 0.3 This indicates a weak or moderate association between the variables. There is some relationship between the variables, but it is not relevant.

Note: SS = Outstanding, SA = Satisfactory, S = Sufficient, NA = Not approved.

Source: own development

The non-parametric Chi-square test was performed with the information presented in Figure 2. In a similar way to the two concepts evaluated previously, the results of this test were also favorable for the proposed methodology, since with the evidence presented in the third year, H03 could be rejected. Therefore, it is concluded that the use of the methodology showed a statistically significant difference compared to the traditional manner. This indicator also improved during the implementation of the interventions, as seen in Table 12,

where the p-value became increasingly smaller from 2018 to 2021. In addition, the measures of strength of dependence (phi, V of Cramer and the contingency coefficient) were better. However, although these reflect that there is some relationship between the variables, it only represents a moderate result (Agresti, 2018; Lehmann, 2006).

According to the trend shown, it is anticipated that the p-value will continue to be lower in the following courses in relation to the evaluated indicators.

In addition to the previous tests, it was also verified that the grades obtained through the application of the methodology had a median of 88 points, which, in terms of mastery level, represents *satisfactory*. Table 13 presents the proposed hypothesis and the evolution of the results.

Table 13. Average performance with the use of Scratch

Hypothesis	Proof	2018 Results	2019 Results	2020 - 2021 Results
H0: Normal data H1: Non-normal data	Shapiro-Wilk	p-value = 0.007 < α = 0.05 H0 rejection, not normality	p-value = 0.000095 < α = 0.05 H0 rejection, not normality	p-value = 1.589-05 < α = 0.05 H0 rejection, not normality
H04: M2 = 80 H14: M2 > 80	Wilcoxon	p-value = 0.25 > α = 0.05 Not enough evidence was found to conclude that there are significant differences, the median obtained when the methodology is used is not greater than 80 points. r biserial rank coefficient = -0.06984424 |r|<0.1, which is interpreted as a negligible effect size. It indicates that the median achievement is not greater than 80 points.	p-value = 0.0125 < α = 0.05 H04 is rejected, the median obtained when the methodology is used is greater than 80 points. r biserial rank coefficient = 0.1067457 0.1< |r|<0.3, which is interpreted as a small effect size. It indicates that the median achievement is not substantially greater than 80 points.	p-value = 7.488-06 < α = 0.05 H04 is rejected, the median obtained when the methodology is used is greater than 80 points. r biserial rank coefficient = = 0.31 0.3 ≤ |r| < 0.5, which is interpreted as a moderate effect size. It indicates that the median achievement is moderately greater than 80 points.

Note: M2 = Median using the methodology.

Source: own development

The Shapiro-Wilk test was used to check whether the data (the scores obtained after the methodology was used) have a normal distribution (row 1, table 13). In all cases, H0 was rejected, since the data do not have normal behavior. Therefore, the non-parametric Wilcoxon sign test was used to test the proposed hypothesis (row 2, table 13). According to the p-value and the alpha used, H04 is rejected, which indicates that the median score when the methodology is used exceeds 80 points.

The Wilcoxon test was also used with the data found in the intervention. The first time (2018) there was not enough evidence to reject H04; However, it was rejected from the second year of the intervention (2019) onwards (row 2, table 13). Finally, regarding the 2020-2021 results, the effect size is moderate, since it shows a median achievement that is moderately higher than 80 points.

Discussion

The methodology described in this work allows us to verify the usefulness of the use of technologies such as Scratch to achieve significant learning in students of the first computer programming courses, which, in turn, can help to improve their academic performance, particularly in the analyzed case of the LIS at the UADY – UMT.

Researchers and teachers in the area of computer programming agree on the difficulties that new engineering students face in being able to successfully develop the complex skills related to algorithms and programming. However, a large part of the success of their professional career are related with the adequate mastery of the content of the courses that are studied at the beginning of their training. For this reason, in the review of the literature it has been confirmed that different authors have proposed a diversity of approaches, strategies and methodologies in the teaching process, which have been used as support tools for the learner in their introductory process to the programming. The study presented in this document contributes in this sense, providing a novel methodology that has been successfully implemented in the previously described context.

The importance of improving the traditional approach used in teaching and learning in introductory programming courses is evident in the literature. For instance, the study performed by Jiménez-Toledo *et al*. (2019), where a large number of experiences in higher education courses was collected, synthesized and categorized. This kind of results is the basis for this research, which seeks to support teaching/learning processes.

As Insuasti (2016) argues, one of the reasons for the disappointing results in programming fundamental courses may be the complexity of learning the structures and rules of syntax and semantics of languages. Therefore, to address this problematic, it was decided to include the Scratch tool in the proposed methodology, since by not using a specific textual syntax, it can reduce the complexity of learning. In fact, Scratch is a tool that focuses on the assembly of blocks, which is relevant in the Algorithms subject, since it is not intended to teach the student how to program, but rather to study the fundamentals and basic structures of programming.

As explained by Malliarakis *et al*. (2014), traditional learning of computer science fundamentals courses involves many challenges, which considerably increases dropout rates (Rodríguez, 2014). In the reviewed literature, several authors such as Rizvi *et al* . (2011), Muñoz *et al* . (2012), Hernández (2013), Muñoz *et al.* (2017), Hidalgo *et al.* (2021) and Bedoya (2021) report high dropout rates in introductory programming subjects in careers related to computer science.

This was a problem that the UMT had in the first courses in which the subject of Algorithms was taught, hence the decision to implement a learning methodology mediated by Scratch. In this sense, the results have been satisfactory after several applications and adjustments.

Since the Scratch tool belongs to the category of educational games focused on teaching multiple learning units (Jiménez-Toledo et al., 2019), it was initially used in the Algorithms subject to create small applications related to everyday problems. Subsequently, the students advanced in their mastery until they created complete games that included each of the structures analyzed in the subject. Additionally, the students added elements of the Mayan culture to their games, which is of great importance in the specific context of the state of Yucatán, where the research was carried out.

Considering the above, it can be stated that the Scratch programming language is an appropriate ICT for educational activities, since the motivation of the students was also appreciated during the laboratory activities. In addition, collaborative work was promoted, as well as the stimulation of algorithmic thinking and creativity. In short, the developed projects exceeded the expectations of the subject teacher due to the diversity of elements included in the video games, such as the characters, setting elements, backgrounds and sounds.

The obtained results coincide with the use of Scratch that other teachers have performed to provide support to their students in introductory programming courses, among which we can mention Rizvi et al. (2011) and Muñoz et al. (2015, 2017).

Finally, with the data obtained at the conclusion of the experiments, it has been confirmed that the results have progressively improved throughout the three interventions. This shows that the used approach makes a significant difference.

However, some limitations were identified in the study. For instance, the Scratch tool has an obstacle to communicating ideas, since the blocks are previously configured with shapes that cannot be modified. Furthermore, these take up both horizontal and vertical space, which restricts the flexibility of possible combinations.

On the other hand, it is known that environments that use blocks do not reveal syntax errors. However, inevitably the student will have to face them, since they will arise when they begin to use other types of languages (Gómez and Martínez, 2021). Additionally, it was observed that not all the structures that are included in the didactic planning can be represented in Scratch, which is clearly a major limitation.

Conclusions

The literature shows the effort of teachers and researchers to improve the teaching and learning processes of students who are in introductory programming courses, since good performance in such courses will result in the success of the following subjects they must take.

In the specific case of the Algorithms course described in this research, it was taught at the UMT between 2016 and 2017 through the traditional mode. Then, with the information collected during this period and with the support of the reviewed literature, a methodological proposal emerged, which includes the use of the Scratch tool. The use of the methodology mediated by Scratch has shown an increase in the academic performance of the students in the different interventions carried out in the course.

In this sense, a comparative analysis was carried out. This involved the results of the final report cards of the courses taught in a traditional way, as well as the results obtained when using the methodology mediated by Scratch. For that, three aspects were considered: the average academic achievement, the passing rate and the level of mastery regarding the student's skills.

In the first aspect evaluated, the median academic achievement turned out to be better using the Scratch tool compared to those courses taught in a traditional way. Regarding the approval rate, it was found that the result is dependent on the method used (with or without the use of Scratch), since notable progress was achieved when the technological tool was used. Finally, the results were evaluated considering the level of mastery, and it was found that these depend on the method implemented, with an increase in the percentage of students with a *satisfactory* or *outstanding* level when Scratch was used.

Finally, other benefits of the methodology were evident when an analysis of the general median of the courses involved was carried out. The result was satisfactory in accordance with what was established by the UADY guidelines, which is convenient within the context of this research.

Future lines of research

Currently, we continue to support new students in the subject of Algorithms from two approaches. On the one hand, based on the results found, Scratch continues to be used; However, based on the fact that there may be other types of tools that provide support to students not only to pass the Algorithms course, but also to help them in the related subjects of the following semesters, it was decided to use another software tool, the which is also a visual programming language based on block management, called Blockly. This tool generates code in other languages, such as JavaScript, Python, PHP, etc., which are important in the Software Engineering degree, and can give the student a broader overview of programming concepts. We have been working with Blockly for a semester, so evaluations are expected to be made and results generated after two additional interventions. The goal, as future work, is to verify which of the two tools can achieve better results.

On the other hand, a study has begun to find out the level of computational thinking with which students arrive at the UMT, especially in the content of algorithms, with the aim that this information can serve as support in the teaching process of courses related to algorithms.

References

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

Bedoya, A. (2021). *Diagnosis of engineering students in digital entertainment design to facilitate the learning of programming fundamentals.* XVI International Design Week in Palermo.

Cairó, O. (2005). *Programming methodology. Algorithms, flowcharts and programs*. Alpha Omega.

Calderón, C. (2020). *Computer archaeology: design and implementation of Facit mechanical calculators with Scratch* (degree work). Polytechnic University of Valencia.

Campbell, O. and 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. and Benavides, D. (2021). Using Scratch to Improve Learning Programming in College Students: A Positive Experience from a Non-WEIRD Country. *Electronics*, *10* (10), 1180.

Chair of Concepts of Algorithms, Data and Programs, UNLP. (2016). Why is "thinking algorithms" so important in Computer Science? *Institutional Magazine of the Faculty of Informatics Bit & Byte*, (4).

Colorado, M. (2020). *Compendium and development of digital tools to support teaching-learning-evaluation in Software Engineering* (degree work). Tizimín Multidisciplinary Unit, Autonomous University of Yucatán.

Gómez, J., Narváez, L., Rejón, E. and Reyes, J. (2016). *Didactic planning of Algorithms* (unpublished manuscript). Autonomous University of Yucatan.

Gómez, M. and Martínez, P. (2021). *Scratch, Python, or what? Criteria for choosing an environment to teach programming to beginners.* Argentine Conference on Computer Science Didactics.

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

Hernandez, G. (2013). Teaching beliefs and didactics of computer programming. *University Magazine, Teaching, Research and Innovation*, *2* (2), 87-103.

Hernández, R., Fernández, C. and Baptista, P. (2014). *Research methodology* (6th ed .). McGraw Hill.

Hidalgo, C., Bucheli, V., Restrepo, F. and González, F. (2021). Teaching strategy based on collaboration and automatic evaluation of source code in a CS1 programming course. *Engineering Research and Innovation*, *9* (1), 50-60. https://doi.org/10.17081/invinno.9.1.4185

Insuasti, J. (2016). Problems of teaching and learning programming fundamentals. *Education and Social Development Magazine*, *10* (2). https://doi.org/10.18359/reds.1966

Jiménez-Toledo, JA, Collazos, C. and Revelo-Sánchez, O. (2019). Considerations in the teaching-learning processes for a first computer programming course: a systematic review of the literature. *TecnoLógicas*, 22, 83-117. https://doi.org/10.22430/22565337.1520

Joyanes, L. (2003). *Programming Fundamentals: Algorithms and Data Structure*. McGraw-Hill.

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

López, J., Hernández, C. and Farran Leiva, Y. (2011). An automatic evaluation platform with an effective methodology for teaching/learning in computer programming. *I will engineer. Chilean Engineering Magazine*, *19* (2), 265-277. https://doi.org/10.4067/S0718-33052011000200011

Malliarakis, C., Satratzemi, M. and Xinogalos, S. (2014). Educational Games for Teaching Computer Programming. In 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 (nd). *Scratch-Imagine, Program, Share.* https://scratch.mit.edu/

Monjelat, N., Cenacchi, M. and San Martín, P. (2018). Programming for everyone? Tools and accessibility: a case study. *Latin American Journal of Inclusive Education*, *12* (1), 213-227. https://doi.org/10.4067/S0718-73782018000100213

Muñoz, R., Barcelos, S., Villarroel, R., Barría, M., Becerra, C., Noel, R. and Silveira, F. (2015). *Use of Scratch and Lego Mindstorms to support teaching in Programming Fundamentals.* Proceedings of the XXI Conference on University Education in Informatics, Andorra La Vella.

Muñoz, R., Barcelos, T., Villarroel, R. and 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. and Quiroz, P. (2012). *Determining the difficulties in learning the first programming subject in computer civil engineering students*. New Ideas in Educational Computing. Proceedings of the XVII International Congress of Educational Informatics, TISE, Santiago, Chile.

Pérez-Narváez, H., Roig-Vila, R. and Jaramillo–Naranjo, L. (2020). Use of SCRATCH in learning programming in higher education. *Cátedra Magazine*, *3* (1), 28-45. https://doi.org/10.29166/10.29166/catedra.v3i1.2006

Pinales, F. and Velázquez, C. (2014). *Algorithms solved with flowcharts and pseudocode.* Autonomous University of Aguascalientes.

Puig, L. (2009). Stories of al-Khwarizmi (3rd installment): origins of algebra. S *uma: Journal of Mathematics Teaching and Learning*, *60*, 103-108.

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

Robins, A., Rountree, J., & 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). Teaching computer programming to beginners: A historical context. *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., and Silverman, B. (2020). History of logo. *Proceedings of the ACM Programming Languages*, *4* , 1-66. https://doi.org/10.1145/3386329

University of Yucatán (2012). *Educational model for comprehensive training*. (unpublished manuscript). Autonomous University of Yucatan.

Velázquez, J., Paredes, M., Cavero, S. and Palacios, *D.* (2023). *Improvement of a subject for teacher training in block-based programming.* Proceedings of the XXIX Conference on University Teaching of Informatics, Granada, Spain.

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