Journal of Information Technology Education: Research

An Official Publication of the Informing Science Institute
InformingScience.org

JITEResearch.org

Volume 22, 2023

DISCOVERING INSIGHTS IN LEARNING ANALYTICS THROUGH A MIXED-METHODS FRAMEWORK: APPLICATION TO COMPUTER PROGRAMMING EDUCATION

Edna Johanna Chaparro Amaya	Universidad Nacional de Colombia, Bogotá, Colombia	edchaparroa@unal.edu.co
Felipe Restrepo-Calle*	Universidad Nacional de Colombia, Bogotá, Colombia	ferestrepoca@unal.edu.co
Jhon J. Ramírez-Echeverry	Universidad Nacional de Colombia, Bogotá, Colombia	jjramireze@unal.edu.co

* Corresponding author

ABSTRACT

Aim/Purpose	This article proposes a framework based on a sequential explanatory mixed-methods design in the learning analytics domain to enhance the models used to support the success of the learning process and the learner. The framework consists of three main phases: (1) quantitative data analysis; (2) qualitative data analysis; and (3) integration and discussion of results. Furthermore, we illustrated the application of this framework by examining the relationships between learning process metrics and academic performance in the subject of Computer Programming coupled with content analysis of the responses to a students' perception questionnaire of their learning experiences in this subject.
Background	There is a prevalence of quantitative research designs in learning analytics, which limits the understanding of students' learning processes. This is due to the abundance and ease of collection of quantitative data in virtual environments and learning management systems compared to qualitative data.
Methodology	This study uses a mixed-methods, non-experimental, research design. The quantitative phase of the framework aims to analyze the data to identify behaviors, trends, and relationships between measures using correlation or regression analysis. On the other hand, the qualitative phase of the framework focuses on conducting a content analysis of the qualitative data. This framework was applied to

Accepting Editor Stamatis Papadakis | Received: April 28 2023 | Revised: July 14, July 25, August 13, 2023 | Accepted: August 14, 2023.

Cite as: Chaparro Amaya, E. J., Restrepo-Calle, F., & Ramírez-Echeverry, J. J. (2023). Discovering insights in learning analytics through a mixed-methods framework: Application to computer programming education. *Journal of Information Technology Education: Research, 22,* 339-372. https://doi.org/10.28945/5182

(CC BY-NC 4.0) This article is licensed to you under a Creative Commons Attribution-NonCommercial 4.0 International License. When you copy and redistribute this paper in full or in part, you need to provide proper attribution to it to ensure that others can later locate this work (and to ensure that others do not accuse you of plagiarism). You may (and we encourage you to) adapt, remix, transform, and build upon the material for any non-commercial purposes. This license does not permit you to use this material for commercial purposes.

	historical quantitative and qualitative data from students' use of an automated feedback and evaluation platform for programming exercises in a programming course at the National University of Colombia during 2019 and 2020. The research question of this study is: How can mixed-methods research applied to learning analytics generate a better understanding of the relationships between the variables generated throughout the learning process and the academic performance of students in the subject of Computer Programming?
Contribution	The main contribution of this work is the proposal of a mixed-methods learning analytics framework applicable to computer programming courses, which allows for complementing, corroborating, or refuting quantitatively evidenced results with qualitative data and generating hypotheses about possible causes or explanations for student behavior. In addition, the results provide a better understanding of the learning processes in the Computer Programming course at the National University of Colombia.
Findings	A framework based on sequential explanatory mixed-methods design in the field of learning analytics has been proposed to improve the models used to support the success of the learning process and the learner. The answer to the research question posed corresponds to that the mixed methods effectively complement quantitative and qualitative data. From the analysis of the data of the application of the framework, it appears that the qualitative data, representing the perceptions of the students, generally supported and extended the quantitative data. The consistency between the two phases allowed us to generate hypotheses about the possible causes of student behavior and provide a better understanding of the learning processes in the course.
Recommendations for Practitioners	We suggest implementing the proposed mixed-methods learning analytics framework in various educational contexts and populations. By doing so, practitioners can gather more diverse data and insights, which can lead to a better understanding of learning processes in different settings and with different groups of learners.
Recommendations for Researchers	Researchers can use the proposed approach in their learning analytics projects, usually based exclusively on quantitative data analysis, to complement their results, find explanations for their students' behaviors, and understand learning processes in depth thanks to the information provided by the complementary analysis of qualitative data.
Impact on Society	The prevalence of exclusively quantitative research designs in learning analytics can limit our understanding of students' learning processes. Instead, the mixed-methods approach we propose suggests a more comprehensive approach to learning analytics that includes qualitative data, which can provide deeper insight into students' learning experiences and processes. Ultimately, this can lead to more effective interventions and improvements in teaching and learning practices.
Future Research	Potential lines of research to continue the work on mixed-method learning analytics methodology include the following: first, implementing the framework on a different population sample, such as students from other universities or other knowledge areas; second, using techniques to correct unbalanced data sets in learning analytics studies; third, analyzing student interactions with the automated grading platform and their academic activities in relation with their activity grades; last, using the findings to design interventions that positively impact academic performance and evaluating the impact statistically through

experimental study designs. In the context of introductory programming education, AI/large language models have the potential to revolutionize teaching by enhancing the learning experience, providing personalized support, and enabling more efficient assessment and feedback mechanisms. Future research in this area is to implement the proposed framework on data from an introductory programming course using these models.

Keywords learning analytics, mixed methods, computer programming, correlation analysis, content analysis

INTRODUCTION

The past few decades have seen an increase in the use of technology in education, including computers, electronic boards, virtual environments, and learning management systems. As a result, the amount of data collected during the learning process has increased exponentially, providing potential insights into the factors that contribute to academic success (Baker & Inventado, 2014; Siemens, 2013). This information can guide institutions, faculty, and students in making decisions related to educational administration, teaching, and learning (Kumar et al., 2015; Lazarinis et al., 2022), as well as learning outcomes assessment (Ladias et al., 2022). Learning analytics, which involves the analysis of educational data, is considered the future of education, particularly in higher education contexts (Arnold & Pistilli, 2012; Long & Siemens, 2011). Learning analytics builds on traditional educational research principles, and leverages innovations such as new forms of digital data collection and advanced computational analysis techniques from data science and artificial intelligence (Pistilli et al., 2014).

In the context of computer programming courses, learning analytics has been used for various purposes, such as detecting students at risk of failing a course (Azcona et al., 2019; Lagus et al., 2018), tracking course progress (Shen et al., 2020), and providing personalized feedback to students (Lu et al., 2017). The importance of incorporating learning analytics into computer programming education stems from the inherent complexity of programming tasks (Salguero et al., 2021). For example, students must develop problem-solving skills to tackle complex tasks such as understanding the problem at hand, translating the problem statement into an algorithm using techniques such as pseudocode or flowcharts, manually calculating the output using specific input data, implementing the program based on the designed algorithm, compiling the program, and identifying and correcting any syntax errors or bugs (Aissa et al., 2020). In addition, computer programming courses often face the challenge of maintaining student interest in the field (Margulieux et al., 2020) and ensuring that students acquire the expected knowledge as perceived by their instructors (Salguero et al., 2021). Therefore, having tools and techniques that can help improve the learning design and facilitate student proficiency is of great value (Shen et al., 2020).

However, research by Mangaroska and Giannakos (2017) suggests that quantitative research designs still predominate over mixed methods and qualitative studies in learning analytics. This finding is consistent with those of Macfadyen and Dawson (2010) and Tempelaar et al. (2016), who highlight the limitations of using quantitative data as the sole source of information to understand students' learning processes. This problem arises due to the abundance, greater availability, and ease of collection of quantitative data in virtual environments and learning management systems compared to qualitative data (Mangaroska & Giannakos, 2017).

Thus, in this work, we propose a framework based on a sequential explanatory mixed-methods design in the learning analytics domain to enhance the models used to support the success of the learning process and the learner. The framework consists of three main phases: (1) quantitative data analysis; (2) qualitative data analysis; and (3) integration and discussion of the results. Furthermore, we apply this framework by examining the relationships between learning process metrics and academic

performance in the subject of Computer Programming coupled with a questionnaire on students' perceptions of their learning experiences in this subject. We propose to answer the research question:

> How can mixed-methods research applied to learning analytics generate a better understanding of the relationships between the variables generated throughout the learning process and the academic performance of students in the subject of Computer Programming?

The proposed methodological design for this study is non-experimental and uses a mixed approach. The quantitative phase of the research aims to determine the relationships between the calculated metrics of the learning process and the academic performance of students in the subject of Computer Programming. On the other hand, the qualitative phase of the methodology focuses on the content analysis of the qualitative data obtained from a questionnaire in which students expressed their learning experiences in the subject.

This document is structured as follows. The second section provides a description of the conceptual framework and related work on learning analytics, both in general and in the context of computer programming courses. The third section explains the methodological framework of learning analytics based on mixed methods proposed in this research. The fourth and fifth sections describe the preparation, transformation, and analysis of quantitative and qualitative data, respectively. The sixth section presents a discussion of the results, integrating the quantitative and qualitative analysis. Finally, the last section presents the conclusions and future work derived from this research.

BACKGROUND AND RELATED WORKS

LEARNING ANALYTICS

Learning analytics is a multidisciplinary field that studies different aspects of education across different contexts. While it is not the aim to provide an exhaustive review of the extensive literature on the topic, several key aspects can be highlighted. These include academic performance, student retention, motivation (Lonn et al., 2015), engagement (Coffrin et al., 2014), learning gains, satisfaction (Elia et al., 2019), metacognitive skills (Wu & Wu, 2018), and self-regulated learning ability, which is determined by analyzing individual records of academic performance, interactions with course content, and personal information (Kizilcec et al., 2017). Researchers have proposed various models for data analysis and the development of personalized feedback systems (Arnold & Pistilli, 2012), as well as predictive models to identify at-risk students (Monllaó Olivé et al., 2020). Other researchers, such as Andergassen et al. (2014), and Barber and Sharkey (2012), have investigated potential relationships between learning outcomes, student use of the course learning management system (LMS), and demographic information.

LEARNING ANALYTICS IN COMPUTER PROGRAMMING COURSES

The proposed methodological framework of this work aims to apply learning analytics using a mixed research approach in a computer programming course. In the field of computer science, learning analytics has gained significant importance. Specifically, in computer programming courses, researchers are actively exploring ways to predict student behavior and provide personalized feedback. For instance, Azcona et al. (2019) proposed a model to identify students at risk of failing a Python programming course and provide personalized feedback. Shen et al. (2020) used a heat map to visualize student engagement with educational resources and activities in an introductory Python MOOC, examining access patterns and identifying similarities and differences. Lu et al. (2017) applied learning analytics to identify students in need of immediate intervention in a Python MOOC, allowing instructors to create adaptive learning guides based on the information gathered. Macfadyen and Dawson (2010) analyzed the usage tracking data from an LMS used in a course with Blackboard-Vista, while Vahdat et al. (2015) aimed to understand the behavior of systems and computer engineering students in a course using a circuit simulator. Additionally, researchers have also conducted several

studies to identify the variables of the learning process that correlate with students' academic performance. For example, Zacharis (2015) developed a model to predict students at risk of low performance using data collected from the Moodle platform.

Related Works

Researchers have proposed several methodological frameworks for applying learning analytics in educational research. Clow's (2012) cyclical model is a closed loop that compares the investigation's results with a reference point, such as previous data or expected results and design interventions that modify the same learning process studied. Aljohani et al. (2019) proposed a framework that adapts learning analytics applications to the specific requirements of the course, divided into instructional, data, analytical, and presentation levels. Carter et al. (2019) proposed a cyclical process consisting of observable behaviors' operationalization, data collection, data analysis, intervention design, and intervention implementation. Ihantola et al. (2015) established an architecture of the systems and subsystems present in learning analytics research applied in computer science courses. Siemens (2013) proposed a generalizable architecture that uses a top-down approach to systematize the educational resources used.

Despite the progress in learning analytics, there are still several challenges in this field. One of the main challenges is the over-reliance on quantitative methods in research, as opposed to qualitative or mixed methods (Mangaroska & Giannakos, 2017; Tempelaar et al., 2016). Moreover, with the recent shift towards semi-face-to-face or fully virtual classrooms, learning analytics applications based exclusively on quantitative methods face difficulties in comprehending learning processes entirely (Kop et al., 2017; Rienties & Toetenel, 2016). To address this limitation, high-quality educational information is needed to inform decision-making on the generation and implementation of educational interventions (Hilliger et al., 2020).

Mixed Methodological Design for Learning Analytics

The proposed methodological framework aims to apply learning analytics using a mixed research approach. The research design is non-experimental as the data have been collected without modifying the variables of the context. The proposed design for educational research is complemented by a mixed methods research approach, and an explanatory sequential type of study is suggested for this type of research (Bryman, 2015; Creswell, 2014). The explanatory sequential methodology uses the results found with qualitative methods to find a likely explanation for the findings found by quantitative methods.

Figure 1 illustrates the proposed methodological framework, which consists of three sequential global phases: (1) quantitative data, (2) qualitative data, and (3) discussion. The first two phases are divided into three stages, which are represented in the figure by the dotted black lines. These stages correspond to data preparation, data transformation, and data analysis. The quantitative analysis (Phase 1) is consistent with existing research in learning analytics, which has traditionally focused on quantitative analysis. While the specific approach in Phase 1 may have some novel aspects, it is based on established practices of data collection and analysis in the field of learning analytics. In contrast, the qualitative analysis (Phase 2) and the discussion of the results of both quantitative and qualitative analyses (Phase 3) can be seen as novel contributions of this research. The literature review indicated that the inclusion of qualitative analysis in learning analytics is an emerging area with limited existing research. Therefore, the inclusion of Phases 2 and 3 in the proposed framework adds value by addressing this gap and providing new insights into the learning analytics process. Each of the activities in the proposed methodology is described in detail below.

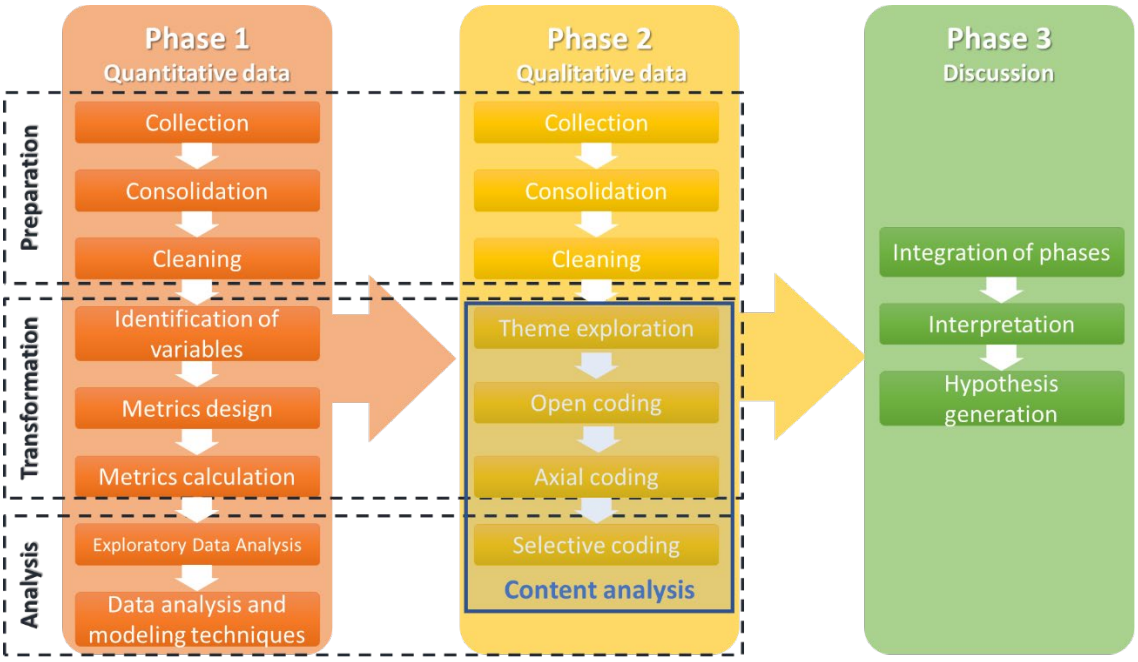

Figure 1. Proposed methodological framework for learning analytics through a mixed-methods research approach

Phase 1: Quantitative Data

In this phase, quantitative data related to students' interactions in the course and their academic performances are collected from the learning platform. This data includes information such as the number of times students accessed the platform, the time spent on each activity, the number of attempts made, and the scores obtained.

This phase begins with the *data collection*, where the location and format of the available data is identified. Then, the data on the students' learning process is gathered using a data management and analysis tool. The next step is *dataset consolidation*, which is necessary because raw data is often disaggregated. During this stage, the most appropriate data structures are identified for the organization and manipulation of the consolidated data.

After consolidation, *dataset cleaning* is performed to identify variables that provide useful information about the learning process. Variables that are not related to the objective of the study or those with data quality issues are discarded. The *identification of variables* follows, where a literature review of measurements and metrics used in educational research in the field of the target course is conducted. The measurements found in the literature that are present in the dataset are then identified, and appropriate metrics are built using them.

Next, *metrics design* is performed, where the equations needed for the estimation of metrics are established based on the results found in the literature. At this point, it is necessary to define the scale (nominal, ordinal, interval, ratio) for the metrics and units of measurement when appropriate. The equations proposed in the metrics design stage are then applied in the data management tool, and the values obtained are stored for later analysis (*metrics calculation*). *Exploratory data analysis* is then performed based on the previous measurements and metrics to identify behaviors and trends. Descriptive statistics, such as the arithmetic mean, dispersion measures, skewness, and visualizations like box plots, histograms, etc., are used during this stage.

Finally, *data analysis and modeling techniques* are applied, and the relationships between metrics and measurements are identified through correlation or regression analysis. In addition, supervised or

unsupervised machine learning techniques can be applied if the goal of the work is to obtain classifications, regressions, or clustering of data.

PHASE 2: QUALITATIVE DATA

To begin this activity, the research design for the qualitative methodology needs to be selected, such as grounded theory, ethnographic study, narrative, phenomenological, or action-participatory research (Hernández-Sampieri et al., 2014). Then, the population sample of interest should be identified, and the research method for the study (*data collection* tools), such as interviews, questionnaires, focus groups, etc., should be selected. All collected data should be stored in a defined location, such as a local storage or a shared file storage platform. The format of the stored files should also be determined, depending on the data source, whether it is text, image, video, or audio files.

The next step is *dataset consolidation*. In this activity, all the collected data is stored in a defined location, and the format of the data set is made uniform. After that, the *dataset cleaning* consists of removing data from the dataset; in the case of records identified as having data that is missing, incomplete, atypical, or irrelevant, it should be removed. In addition, if the amount of data is large, computational tools such as Atlas.ti, Decision Explorer, Etnograph, and NVivo may be used.

The content analysis of qualitative data begins with the *theme exploration*. This activity starts the process of content analysis of qualitative data, which is represented by the blue box in Figure 1. Content analysis is defined by Bryman (2015) as the systematic and reproducible quantification of documents and texts, both printed and visual, in terms of predetermined categories. This is a nonlinear and iterative process, as the tasks of coding and categorizing are not single events within the procedure (Hernández-Sampieri et al., 2014). The basic unit of analysis or meaning is chosen, and the information collected is divided into specific fragments labeled with codes that emerge from the interpretation of the data.

The process of *open coding* involves dividing the data into small fragments and labeling them with appropriate codes that indicate global ideas. Similar codes are grouped and labeled with the same code to ensure that segments related to the same topic are categorized accurately.

The *axial coding* identifies connections between the codes generated in open coding and groups them into categories (Corbin & Strauss, 1990). From these categories, associations are identified, such as causal relationships, context behind observations, or consequences of the phenomenon, and categories can be grouped into general themes.

Finally, *selective coding* identifies the central phenomenon or category that unifies all other categories and themes resulting from previous coding (Corbin & Strauss, 1990). This process, also known as data relativization, may refine some codes and result in the creation, mixing, splitting, or elimination of labels. The step-by-step approach to data relativization is as follows (Corbin & Strauss, 1990):

1. Based on the trends identified in the data, define the central category that groups all the themes and categories of the axial coding and captures the general idea of the qualitative research results.
2. Identify the links between the general category and the rest of the themes and categories to determine the final narrative of the research report.
3. Identify the themes, categories, and codes that appear to be unrelated to the central phenomenon identified and verify whether the amount of data from these labels is sufficient to consider the results relevant. In the case that the information is insufficient, the label should be eliminated.
4. Review the original data again and code the fragments of information considering the general category generated.

PHASE 3: DISCUSSION

As the proposed methodological design follows a sequential explanatory mixed methods approach, this phase consolidates the findings of the quantitative phase with those of the qualitative phase. The aim is to explain the findings of the first phase using those of the second phase, which helps to verify whether the behaviors identified through quantitative data are supported or refuted by qualitative data (*integration of phases* stage). This approach broadens the scope of the results of the quantitative phase and generates clarifications of the behaviors found from a qualitative perspective.

The stage of *interpretation* involves a detailed analysis of the research questions and their answers based on the specific results obtained. The results are then compared with the findings of related works that were studied in the literature review. In cases where the results differ from the existing literature, the possible reasons for such discrepancies should be stated, including factors that may be related to the dataset, the course characteristics, and the context, among others.

Finally, the *hypothesis generation* begins with an in-depth analysis of the results obtained, which includes a detailed description of the possible reasons for the identified behaviors. This approach helps to formulate hypotheses that describe aspects of learning processes that may occur in the educational environment under study. Based on these hypotheses, it is essential to reflect on how the research findings contribute to the scientific community, particularly in the field of learning analytics. These findings can guide future research projects and work.

PROPOSAL APPLICATION: QUANTITATIVE DATA

The proposed methodological framework for learning analytics using a mixed-methods research approach is intended to be applied in the context of computer programming courses using educational platforms that facilitate the collection and storage of data on student interactions. The selection of computer programming courses as the context for the case study is primarily intended to demonstrate the practical application of the proposed methodology in a real-world setting. However, it is important to note that the methodology itself can be applied to other domains within the field of learning analytics.

This study analyzes the use of the UNCode platform, an educational platform used in the Computer Programming courses at the National University of Colombia for the automatic evaluation of programming exercises (Restrepo-Calle et al., 2018, 2020). The research question of this proposal application is:

> How can mixed-methods research applied to learning analytics generate a better understanding of the relationships between the variables generated throughout the learning process and the academic performance of students in the subject of Computer Programming?

The study considers two sources of information: (1) the record of students' interactions with the UNCode platform, stored in a MongoDB database, and (2) questionnaires on students' perceptions about the use of the educational platform, stored in spreadsheets by academic period.

UNCode allows students to submit multiple attempts (source code or Jupyter notebooks) to solve programming tasks. For each solution attempt, the platform stores the program file, submission date, and time. It also provides automatic feedback through verdicts related to syntax, semantics, and program efficiency, as well as a numerical grading based on the test cases the program solved. UNCode provides several learners' support tools, such as syntax highlighting, code auto-completion, Linter (suggestions for good programming practices), visualization of code execution, custom tests, and grade reports. Further details on the functionalities of UNCode can be found in Restrepo-Calle et al. (2018).

In the context of the computer programming course, the objective of Phase I is to collect and analyze quantitative data from the students' interactions with the educational platform and their

corresponding academic performance, as well as quantitative data from the students' perception questionnaires. The collected quantitative data will provide a rich source of information that can help identify relationships, evaluate performance, plan interventions, and improve the computer programming course for better student outcomes.

DATA PREPARATION

Interaction with UNCode

The study population consisted of students who took the subject of Computer Programming at the National University of Colombia between the first academic period of 2019 and the second period of 2020 (2 years - 4 academic semesters), during which the UNCode educational platform was used in the course activities. The study is limited to 24 computer programming courses that used the UNCode platform to support mandatory academic assignments. The total number of students in these groups was 772. The platform was available throughout the study period. *The data collection* process is performed to select the 16 collections in the database. The selected collections are Aggregations, Analytics, Students Grades, Submissions, Tasks, User Tasks, and Users. The data was then organized into individual folders for each course, which contain the following files:

1. Students: This file contains a list of users identified with the student role.
2. Analytics: Information about the use of UNCode tools by users is stored here. The following five tools are available:
    a. Custom input: This tool enables the design and running of custom tests to evaluate the built programs.
    b. Python tutor: This tool allows visualization of the execution flow of the designed program step by step.
    c. Multiple languages code: This option enables the evaluation of source code written in different programming languages, such as C/C++, Java, and Python.
    d. Linter: A highlighting tool that identifies errors and provides recommendations in the source code based on principles of good programming practices.
    e. User statistics: This file contains statistical reports on the grades obtained by each student.
3. UNCode_grades: This file contains the final grades assigned to each student that corresponds to the weighted average of the grades obtained from the activities performed within the platform.
4. Submissions: This file contains a record of the solution attempts sent by the students in the course activities. Each solution attempt is specified by the date and time of submission, the activity identifier, the username, and the course identifier. Additionally, the file contains the identifier of the file sent, the numerical grade on a scale from 0 to 100, the tests performed, and the verdict obtained.
5. Input: This file contains the specifications of the files sent by each student in each solution attempt. The columns correspond to the file identifier (index), the name of the file loaded on the platform (file_name), and the programming language used (language).
6. Tasks: This file contains information about the course activities developed within the platform. Each file has a column with the course identifier (course_id) and the activity identifier (task_id).
7. User_tasks: This file summarizes the number of attempts made by each student in the course activities.

For *the data consolidation*, each course folder contains a directory containing all the files submitted by the students in each solution attempt. These files are organized into directories per student, which contain subdirectories for each course activity.

Regarding *the data cleaning* stage, first, a filter was applied to select computer programming courses with a high number of activities on the platform. The courses G15-2019-2 and G16-2019-2 have the maximum number of activities, 102. In contrast, G8-2020-1 has the least number of activities, only 15. All courses have sufficient interaction data recorded. However, the pilot courses group-5 and group-6 are discarded, as the platform was used for preliminary study, making their data incomparable to the other courses. Therefore, 22 groups are considered in the final dataset.

Secondly, activities with a low number of submissions are filtered out, using a minimum limit of 15 submissions per activity. Activities with low or no submissions possibly correspond to tests of UNCode operation or optional activities, making the data irrelevant. This filter eliminated 49 activities out of 1404 from the dataset. Activities with notebook-type files are discarded as they are not comparable with source code files submitted in other activities. Three such activities were identified in the course G18_2020_1, reducing the total number of activities in the dataset to 1352.

Subsequently, a student filter is applied based on the total number of submissions registered per student. Some students have few or no submissions registered, indicating early withdrawal. Using the same minimum limit of 15 submissions, 37 students with less than the minimum number of submissions were identified and excluded from the dataset. After this process, the resulting dataset contained data from a total of 735 students.

Perception questionnaires

Moreover, during the same period (from the first semester of 2019 to the second semester of 2020), we conducted a questionnaire-based approach to gather students' perceptions of using UNCode in the Computer Programming courses. The questionnaires were administered to 17 of the 24 course groups, and the responses were stored in spreadsheets by academic period. Although not all course groups participated in the questionnaires due to logistical inconveniences due to instructors' decisions, the representativeness of the selected course groups provides the perceptions of participants from the majority of the groups. Therefore, this fact might not have introduced any potential bias or limitation to the results. The questionnaires were administered before students learned their final grades during week 14 of the course (out of 16 weeks). In addition, the questionnaire was administered via Google Forms, which ensured a convenient and accessible method of data collection. The questionnaires also asked for informed consent from the participants, ensuring ethical considerations in the administration process.

The questionnaire data include demographic information about the students and their responses to questions about their use of the platform. However, only closed-ended questions were considered in this phase, as they provide quantitative data. The questions were presented as statements, and students were asked to answer using a Likert scale from 1 to 6. The statements were as follows:

1. Indicate your level of agreement or disagreement regarding the following statements, with a maximum value of 6 indicating the highest level of agreement and a value of 1 representing the highest level of disagreement:
    a. UNCode was useful in their computer programming learning process.
    b. UNCode was helpful in obtaining automatic grading for the programs you developed in this subject.
    c. The automatic feedback provided by UNCode was useful to know how to correct errors in my programs.
2. Rate the following UNCode features according to how useful you think they are for learning computer programming, with a maximum level of 6 indicating the highest level of usefulness and a value of 1 representing the least level of usefulness:
    a. Testing of programs using user-supplied inputs (custom input).
    b. Programming best practices verification tool (Linter).
    c. Visualization of program execution (Python Tutor).
    d. Performance reports (statistics).

The questionnaire responses on perception are compiled into one file that combines the information from all the courses. Using a Colab notebook, each spreadsheet is converted into a Pandas DataFrame, and then these DataFrames are concatenated into one, which includes the columns with the student's username, date of birth, gender, academic program, and course group (course_id). Furthermore, the responses to the closed-ended perception questions are also included.

To *clean the consolidated data* gathered from the perception questionnaires on the use of UNCode, initially, the dataset is filtered to remove students with insufficient information. This filtering eliminates 33 students, resulting in a final set of 349 students who participated in the perception questionnaire. This number represents 47.5% of the 735 students from the collected dataset with the interaction with UNCode. Next, a filter is applied to the closed-ended questions of the questionnaire, which are answered on a Likert scale. The questionnaire comprises 21 such questions, but 14 are discarded as they have responses from less than 25% of the total number of participants. As a result, only seven of the closed-ended questions are considered in the final dataset, as mentioned above.

Data Transformation

Table 1 lists the 15 measurements that are of interest in this research from the students' interaction with UNCode (*identification of variables*). These are classified into four categories:

1. Submissions: data related to the attempts made by each student to solve programming assignments.
2. Verdicts: data related to the feedback received for each solution attempt.
3. Tool usage: data on the number of times each platform tool is accessed.
4. Academic performance: numerical grading of the submissions made by the students.

Moreover, software metrics of the students' programs are obtained from the source code files submitted as solutions to the programming tasks.

Table 2 presents the 13 measures identified in the dataset obtained from the perception questionnaires (*identification of variables*). These measures are classified into two categories:

1. Demographic data: includes information that characterizes the student sample.
2. Closed-ended questions: include responses on a Likert scale regarding the use of the platform and its tools during the course activities.

Based on the measures identified in Table 1 and Table 2, 25 metrics were developed and categorized as follows (*metrics design*):

1. Verdict rates: These represent the ratio of a specific type of verdict to the total number of verdicts obtained by each student. The equations used to calculate them are specified in Table 3.
2. Tool usage rates: This category refers to the percentage of accesses to a specific tool in relation to the total number of accesses registered for all tools available on the platform per student. The equations used to calculate tool usage rates are also specified in Table 3.
3. Software metrics: This category represents specific characteristics of the source codes created by students. Table 4 describes the metrics and equations used to calculate them, which are based on the number of operands, operators, executable lines of code, and reserved words used in the code built as a solution to the course activities. These software metrics were calculated using the specialized Python libraries. The lizard library is used to quantify lines of code (NLOC) and token count. The radon library is applied to calculate the cyclomatic complexity (G), maintainability index (MI), and Halstead metrics. Subsequently, the average of the metrics of all the files submitted by each student was estimated.
4. Demographic data: Besides the measures related to demographic data from Table 2, this category includes students' age, which is calculated based on their date of birth recorded in the questionnaires and the date of completion.

It is worth noting that the first two categories of verdict rates and tool usage rates are aimed at identifying the most and least used verdicts and tools, respectively.

Table 1. Measurements considered in the dataset from interaction with UNCode

CATEGORY	MEASUREMENT	DESCRIPTION	SCALE	UNITS
Submissions	Total_Submissions	Number of attempts submitted per student.	Ratio	Count
	Duration_of_Submission	Average time spent by students between submission attempts.	Ratio	Minutes
Verdicts	Accepted	Number of solutions with correct answers.	Ratio	Count
	Wrong_Answer	Number of solutions with incorrect answers.	Ratio	Count
	Compilation_Error	Number of submitted attempts that fail to compile.	Ratio	Count
	Runtime_Error	Number of attempts that succeed in compiling but fail during execution.	Ratio	Count
	Time_Limit_Exceeded	Number of attempts that take too long to execute.	Ratio	Count
	Memory_Limit_Exceeded	Number of attempts that exceed the memory available for execution.	Ratio	Count
	Output_Limit_Exceeded	Number of attempts that exceed the expected program output size.	Ratio	Count
Tool usage	Python_Tutot	Number of logged accesses to the Python tutor tool that allows visualization step-by-step execution of a program.	Ratio	Count
	Custom_Input	Number of registered accesses to the Custom input tool where students perform custom tests on their programs.	Ratio	Count
	Linter	Number of registered accesses to the Linter tool, which highlights syntax and style problems in the source code.	Ratio	Count
	User_Statistics	Number of registered accesses to the interactive dashboard to report on students' individual statistics.	Ratio	Count
	Multiple_Languages_Code	Number of accesses to the Multiple Languages tool that allows submission in different programming languages.	Ratio	Count
Academic performance	uncode_grade	Weighted average of grades of the activities performed by students in UNCode.	Ratio	Percentage

Table 2. Measurements considered in the dataset from the perception questionnaires

CATEGORY	MEASUREMENT	DESCRIPTION	SCALE
Demographic data	Birthdate	Day, month, and year of the student's birth.	Date
	Sex	Variable that represents the sex of the student	Nominal
	Academic program	Corresponds to the student's university career, with 15 options available.	Nominal
Closed-ended questions	QUESTION: Learning process	Level of agreement or disagreement in Likert scale of the student with the statement: "UNCode was useful in their computer programming learning process".	Ordinal

CATEGORY	MEASUREMENT	DESCRIPTION	SCALE
	QUESTION: Automatic grading	Level of agreement or disagreement in Likert scale of the student with the statement: "UNCode was helpful in obtaining automatic grading for the programs you developed in this subject.".	Ordinal
	QUESTION: Feedback	Level of agreement or disagreement in Likert scale of the student with the statement: "The automatic feedback provided by UNCode was useful to know how to correct errors in my programs".	Ordinal
	A_Custom_input	Likert-scale response to the statement: "Rate the following UNCode features according to how useful you think they are for learning computer programming:", regarding testing of programs using user-supplied inputs (custom input).	Ordinal
	A_Linter	Likert scale response to the statement: "Please rate the following UNCode features according to how useful you think they are for learning computer programming:", regarding the programming best practices verification tool (Linter).	Ordinal
	A_PythonTutor	Likert-scale response to the statement: "Rate the following UNCode features according to how useful you think they are for learning computer programming:", regarding the visualization of program execution (Python Tutor).	Ordinal
	A_Statistics	Likert scale response to the statement: "Rate the following UNCode features according to how useful you think they are for learning computer programming:", regarding performance reports (statistics).	Ordinal

Table 3. Metrics based on the verdicts and tool usage measures

CATEGORY	METRIC	EQUATION	SCALE	UNITS
Verdicts rates	Success_rate	$\frac{Accepted}{\sum_i Veredicts_i} \cdot 100$	Ratio	Percentage
	Error_rate_Wrong_Answer	$\frac{Wrong_{Answer}}{\sum_i Veredicts_i} \cdot 100$	Ratio	Percentage
	Error_rate_Compilation_Error	$\frac{Compilation_{Error}}{\sum_i Veredicts_i} \cdot 100$	Ratio	Percentage
	Error_rate_Runtime_Error	$\frac{Runtime_{Error}}{\sum_i Veredicts_i} \cdot 100$	Ratio	Percentage
	Error_rate_Time_Limit_Exceeded	$\frac{Time_{Error}}{\sum_i Veredicts_i} \cdot 100$	Ratio	Percentage
	Error_rate_Memory_Limit_Exceeded	$\frac{Memory_{Error}}{\sum_i Veredicts_i} \cdot 100$	Ratio	Percentage
	Error_rate_Output_Limit_Exceeded	$\frac{Output_{Error}}{\sum_i Veredicts_i} \cdot 100$	Ratio	Percentage
Tool usage rates	Python_Tutor_rate	$\frac{Python_{Tutor}}{\sum_i Tools_i} \cdot 100$	Ratio	Percentage
	Custom_input_rate	$\frac{Custom_{input}}{\sum_i Tools_i} \cdot 100$	Ratio	Percentage
	Linter_rate	$\frac{Linter}{\sum_i Tools_i} \cdot 100$	Ratio	Percentage
	User_Statistics_rate	$\frac{User_{Statistics}}{\sum_i Tools_i} \cdot 100$	Ratio	Percentage
	Multiple_Languages_Code_rate	$\frac{Multiple_{LangCode}}{\sum_i Tools_i} \cdot 100$	Ratio	Percentage

Table 4. Software metrics from source code files submitted by students

CATEGORY	METRIC	DESCRIPTION/EQUATION	SCALE	UNITS
Software metrics	Lines of code (NLOC)	Number of lines of code excluding comments	Ratio	Count
	Tokens_count	Number of tokens of the programming language used in the code.	Ratio	Count
	Cyclomatic complexity (G)	Number of decision blocks contained in the code, plus one. Lower is better.	Ratio	Count
	Program vocabulary (n)	$n = n_1 + n_2$ n_1: The number of distinct operators. n_2: The number of distinct operands	Ratio	Count
	Program length (N)	$N = N_1 + N_2$ N_1: The total number of operators N_2: The total number of operands	Ratio	Count
	Calculated program length (L)	$L = n_1 \cdot log_2(n_1) + n_2 \cdot log_2(n_2)$	Ratio	Decimal
	Volume (V)	$V = N \cdot log_2(n)$ (Acceptable range between 20 and 1000)	Ratio	Decimal
	Difficulty (D)	$D = \frac{n_1}{2} \cdot \frac{N_2}{n_2}$ (Lower is better)	Ratio	Decimal
	Effort (E)	$E = D \cdot V$ (Lower is better)	Ratio	Decimal
	Time required to program (T)	$T = E/18$ (Lower is better)	Ratio	Minutes
	Number of delivered bugs (B)	$B = V/3000$ (Lower is better)	Ratio	Decimal
	Maintainability index (MI)	Measure of how easy to support and change the source code is (0-100). Higher is better.	Ratio	Decimal

DATA ANALYSIS AND RESULTS

This section summarizes the results of *the exploratory data analysis* (univariate analysis) of some of the measures and metrics considered in the dataset. For the total number of submissions made by the students (Figure 2), the average is 176.6 submissions with a standard deviation of 120.8. The standard deviation value corresponds to more than 68% of the average, indicating a high dispersion of the data, which may suggest that students use different techniques in problem-solving. Some students may make many submissions with small changes in each attempt, while others may make extensive modifications resulting in fewer attempts on the platform.

The tool usage (Figure 3) shows that the most used tool is "Custom_input" with 65.0% of the total recorded accesses, indicating that most students prefer to test the effectiveness of their programs with self-designed tests. The usage rates of "Python_Tutor" and "Multiple_Languages_Code" are 17.7% and 12.0%, respectively. The use of "Python_Tutor" indicates that some students find it helpful to observe the step-by-step execution flow of the constructed program, possibly for error location. On the other hand, the use of "Multiple_Languages_Code" reflects the proportion of student interactions related to code submissions in one of the supported programming languages. The least used tools are "Linter" (5.1%) and "User Statistics" (0.5%). The low use of these tools may indicate that students consider the information provided by these tools insufficient to help them improve their constructed solutions.

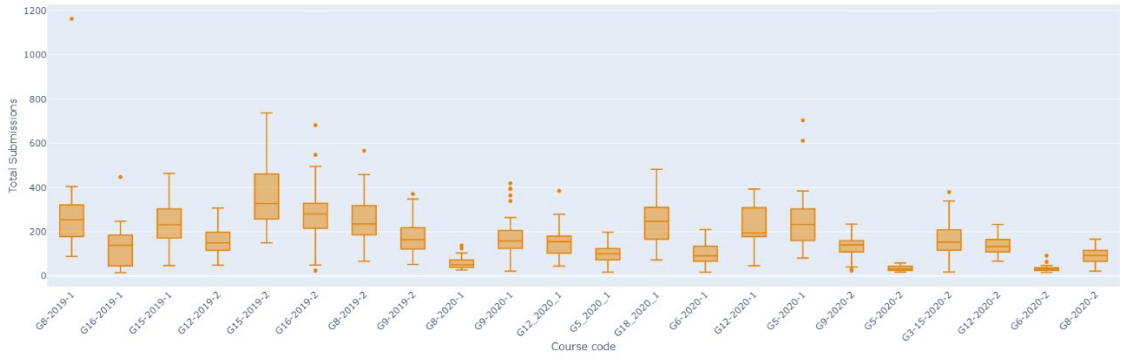

Figure 2. Exploratory Data Analysis: Total submissions box plots by group

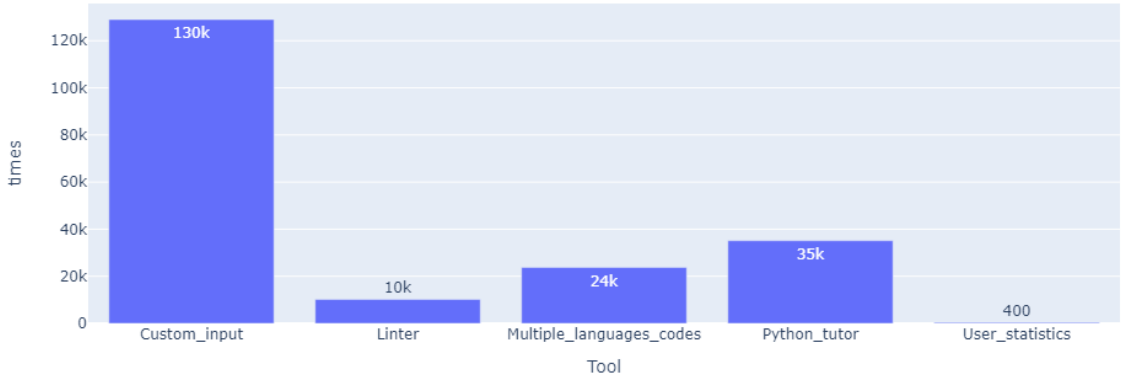

Figure 3. Exploratory Data Analysis: Tools usage

Regarding the verdicts obtained (Figure 4), the judgment with the highest number of records is "wrong answer" (48.9%). This result indicates that most students are successful in designing executable solutions but struggle to meet the specific objectives of the activities. The second verdict with a high number of records is "correct answer" (31.7%), indicating that many students are able to apply the knowledge of the course in solving programming problems. The verdicts that follow in magnitude are "execution error" (14.6%) and "time limit exceeded" (3.9%). The sum of the verdicts obtaining less than 1.0% – "compilation error," "memory limit exceeded," and "result limit exceeded" – represents less than 20% of the recorded judgments, indicating that few students have difficulties or doubts specifically in the executable program design process.

After analyzing the descriptive statistics derived from the software metrics calculated based on the programs constructed by the students, we observed a high degree of dispersion in the data, as indicated by the standard deviations, which are greater than the average in several cases. The metrics with the highest degree of data dispersion are the effort and time required to program, with deviations of 1603.8 and 89.1, respectively. This suggests a wide variety of solutions constructed by the students. On the other hand, the metrics with lower data dispersion are maintainability index (6.4) and difficulty (1.3), corresponding to 10.5% and 35.1% of their respective averages. The low dispersion of these metrics possibly indicates that the students in this course possess similar capabilities and abilities for program construction. These results are consistent with the fact that Computer Programming is an introductory course, and for many students, this is their first exposure to programming languages.

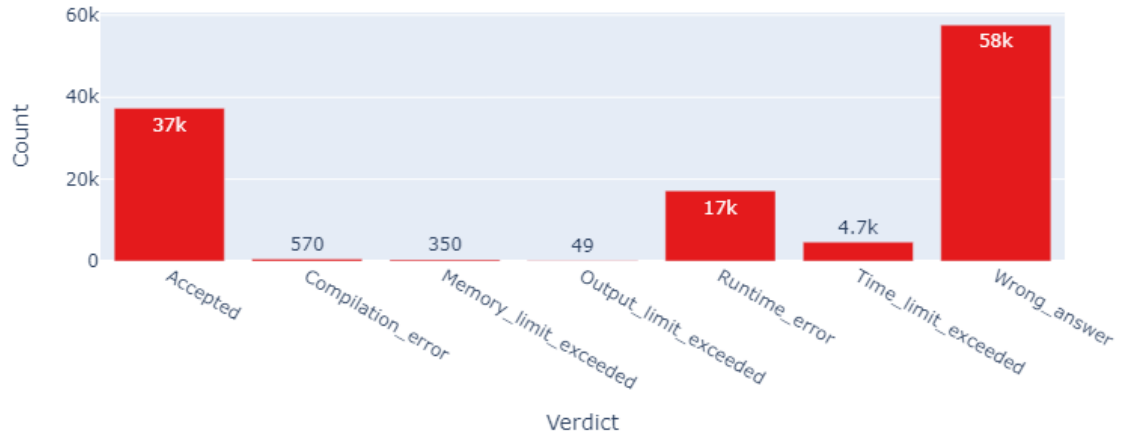

Figure 4. Exploratory Data Analysis: Verdicts obtained in the submissions

Furthermore, a correlation analysis was conducted with UNCode_grade as the dependent variable. Firstly, the Shapiro-Wilks normality test was performed on the academic performance data to determine the appropriate statistical test for calculating the correlations. The test resulted in a p-value > 0.05, indicating that normality cannot be assumed in the data. Therefore, Spearman's correlation coefficient was used as it does not require the samples to be normally distributed. Sex and Academic program variables were not considered in the analysis, as Spearman's coefficient is used to quantify correlations between non-categorical variables. Figure 5 shows the 29 measures and metrics that have a statistically significant correlation (p-value ≤ 0.05) with academic performance.

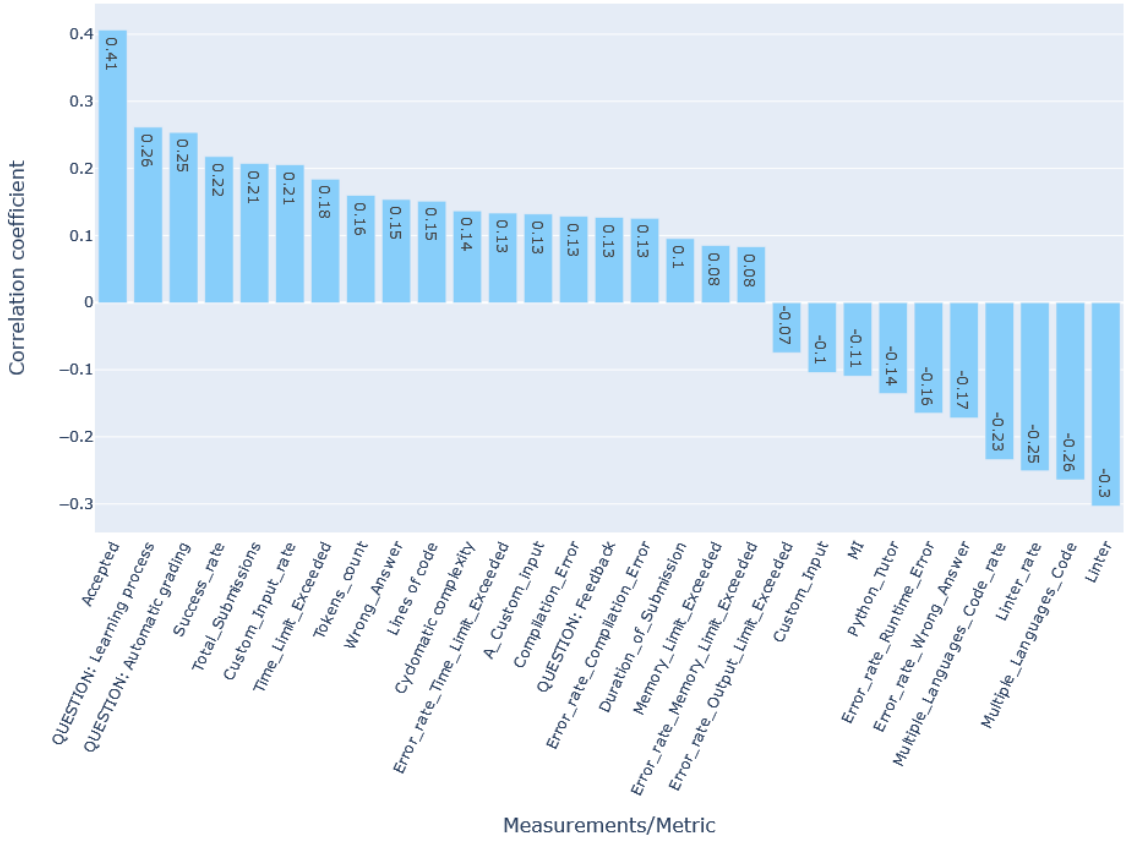

Figure 5. Variables with significant correlations with respect to academic performance

The variable with the highest positive correlation is "Accepted," with a coefficient of 0.41. This is expected, as students who answer more questions correctly are typically more successful at solving course exercises. The next two variables in order of magnitude are "Learning Process" and "Automatic Grading," with coefficients of 0.26 and 0.25, respectively. These results make sense, as students who find the platform useful for learning and appreciate the benefits of automatic grading are more likely to effectively use platform tools and improve their programming skills.

The variable "Success_rate" also has a positive correlation of 0.22, which is expected as students who answer a high percentage of assignments correctly demonstrate strong programming skills. Next, the positive correlations of "Total_Submissions" and "Custom_Input_rate" both with a correlation coefficient of 0.21. In the first case, this may indicate that some students submit many solutions until they get the correct one. In the second case, a student who is able to perform custom tests is likely more knowledgeable about programming and can construct and correct programs more effectively.

In contrast, the variables with the highest negative correlation are the number of accesses to "Linter," "Multiple_Languages_Code," "Linter_rate," and "Multiple_Languages_Code_rate," with values between -0.3 and -0.23. These negative correlations are unexpected, as these tools are designed to support the learning process of students.

The academic performance values were categorized into two groups: passing students (approved), with final grades equal to or higher than 3.0, and students who did not pass the subject (failed). This categorization was done to identify whether the correlations between measures and metrics changed between high-performing and low-performing students. Figure 6 presents the significant correlations (p-value ≤ 0.05) for both categories of students.

Focusing on the variables that turned out to be significant in both categories, we observed that six variables had positive correlation coefficients, while one variable had a negative correlation. The positive correlations were found in the variables Accepted, Success_rate, Time_Limit_Exceeded, Total_Submissions, Wrong_Answer, and Error_rate_Time_Limit_Exceeded.

In all cases, the positive correlations were stronger in the group of students who did not pass the course. Moreover, the variable with a negative correlation shared by both groups of students was the Error_rate_Runtime, which was higher in the case of failed students.

At this point in the research, after completion of the Phase I application, it is worth noting that relying solely on quantitative data may leave researchers with unanswered questions and gaps in their understanding of the research problem. For example, what are the underlying reasons for some of the relationships identified? Therefore, in the proposed mixed-methods framework for learning analytics, we emphasize the importance of incorporating qualitative analysis (Phase II) and discussion of the results (Phase III) to address the research questions more comprehensively.

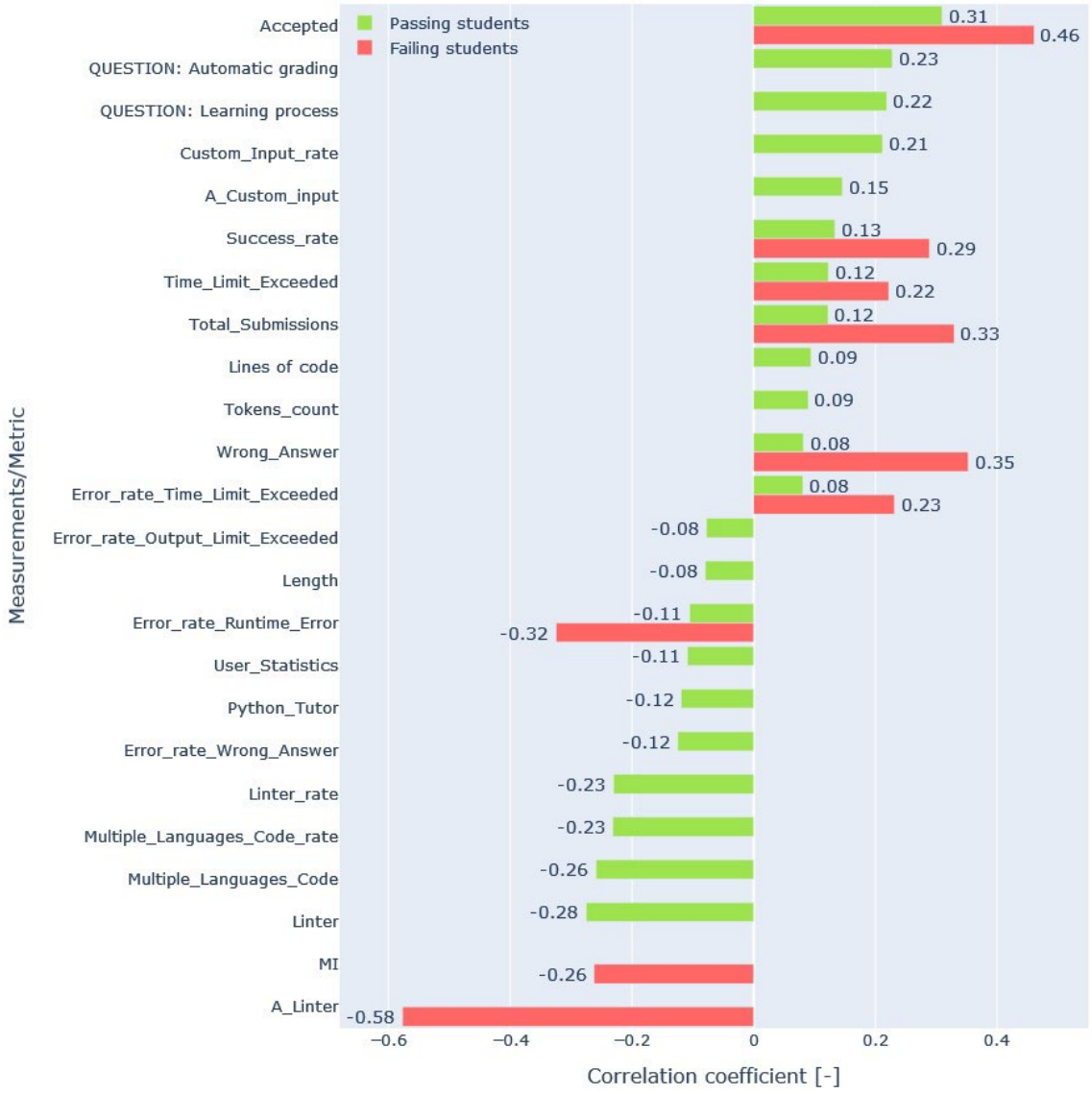

Figure 6. Variables with significant correlations with respect to the academic performance of students discriminated by academic performance (approved and failed)

PROPOSAL APPLICATION: QUALITATIVE DATA

During the qualitative analysis phase, the focus shifts to analyzing qualitative data collected from students' experiences in the computer programming course. Qualitative data may include responses from questionnaires, interviews, or open-ended questions that capture students' perceptions, feedback, and subjective experiences. The methods and analysis of qualitative data seek to find possible explanations for the findings of quantitative methods. The goal is to explain the findings of the first phase with the findings of the second phase, which helps to verify whether the behaviors identified by the quantitative data are confirmed or refuted by the qualitative data. This approach broadens the scope of the results of the quantitative phase and generates clarifications of the behaviors found from a qualitative perspective. By incorporating the qualitative dimension, educators can gain richer insights into students' perspectives, which can inform targeted interventions and improvements in the computer programming course.

Data Preparation

In the quantitative phase, the sample of participants was defined as a subset of the data set from 17 out of 24 Computer Programming courses, covering the period from the second semester of 2019 to the second academic period of 2020 (the *Collection* stage). Perception questionnaires were conducted among students on the use of UNCode in the subject, using Google Forms, and the responses were stored in a spreadsheet for each semester. Only the answers to the open-ended questions of the questionnaire were used in this phase, as they were qualitative in nature. The open questions asked students to explain the reasons behind their agreement or disagreement with the statements of the closed questions considered in the quantitative phase (Table 2). The responses collected from the spreadsheet files were grouped and coded by subject matter to homogenize the data set (*Consolidation and Cleaning* stages). The final data set contains responses from 349 students who participated in the perception questionnaire. The responses were related to open questions on the usefulness of the UNCode platform to enhance learning in computer programming (named ANSWER: Learning process), automatic grading (named ANSWER: Automatic grading), and feedback (named ANSWER: Feedback). The textual answers provided by students in response to each of these three open questions are identified as the basic unit of analysis.

Data Transformation

The stage of *theme exploration* begins the process of content analysis of the qualitative data. This analysis was carried out with the support of the NVivo computational tool. A preliminary review of the basic units of analysis was carried out, identifying recurrent, preconceived, and/or emerging themes.

For the *open coding* stage, the first level of content analysis coding involves assigning one or more codes and categories to each basic unit of analysis. Keep in mind that the open and axial coding process is not a strictly sequential process, and the generation of codes and categories may overlap with the identification of general themes. Additionally, some units may lack detail, and thus only be placed in general themes without category or code. For each of the three questions, the identified categories and codes are listed and described below.

Learning process question

Regarding the learning process question, we generated a total of 5 themes, 21 categories, and 15 codes related to the usefulness of the platform in students' learning process. Table 5 specifies the number of units of analysis grouped by code, category, and general theme, which are classified by students' academic performance. The columns "approved", and "failed" refer to the grouping of responses from the categories previously made during the correlation analysis according to the academic performance of the students. In this way, the column "approved" refers to the number of responses from students who successfully completed the programming course, "failed" represents the number of responses from students who did not meet the requirements of the course, and "total" indicates the total number of responses from students independently of their academic performance. The total sum of references in the table does not correspond to the total number of responses considered, as each basic unit of analysis may be labeled with more than one code, category, or theme. The following categories and codes were assigned:

1. Test cases: References from students highlighting that test cases integrated in the platform help to obtain feedback on the submitted program and identify errors.
2. Formative tips: Mentions of the usefulness of the formative feedback offered by the platform, which provides suggestions about the code construction process in terms of syntax, semantics, efficiency, and maintainability aspects.
3. Knowing the errors: References on how the platform feedback allows students to identify specific errors in the programs built, facilitating the correction and refinement process of the designed solution.

4. Online availability: Benefit for students of the platform working through a web browser, as opposed to programs running in local environments, generating a learning environment where the knowledge acquired can be tested outside the classroom space.
5. Workspace: Platform offers a specific space for the organization of activities, work, and developments made in the course, as a positive aspect.
6. Ease of use: Perceptions about the simplicity of the platform's operation, including the convenience of building and modifying programs directly from the platform.
7. Programming languages: References to UNCode's functionality for selecting various programming languages such as Python, C++, and Java.
8. Constant practice: Mentions of how the platform allows frequent exercise practice, allowing students to strengthen and consolidate the knowledge acquired in class.
9. Custom input: Mentions to the tool that allows performing customized tests by the students.
10. Linter: References to the tool for highlighting syntactic errors and source code style.
11. Python Tutor: Tool integrated into the platform that allows step-by-step visualization of the execution of the programs.
12. Autonomous learning: Mentions of how the platform allows students to expand their knowledge and skills in programming without requiring the direct intervention of the teacher or instructor.
13. Stimulating exercises: Category containing references where UNCode programming problems are characterized as exercises sufficiently demanding to test and strengthen acquired skills and knowledge, without demotivating students due to the level of difficulty.
14. Optimized evaluation: Category assigned to the mentions on how the platform makes the evaluation of the codes submitted by the students much simpler, faster, and more objective. The following four codes are identified in this category: immediate grading, including opinions highlighting the immediacy of the grading, offered by the platform, of the solutions sent in the course activities; objective grading, which mentions the objectivity of the grading obtained in the platform, since the subjectivity of a manual grader is avoided; problem and exercise approach, including mentions on how the platform simplifies the construction of exercises, facilitating the understanding of the context and instructions of the programming problems posed; and submission of academic activities, grouping opinions on how the platform simplifies the process of uploading and submitting solutions to the exercises.
15. Programming skills: Category that contains references on how the use of the platform favors the development of computer programming skills, which transcend from the technical handling of programming tools or languages to relevant long-term skills. Among the skills mentioned by the students, algorithmic thinking, and the understanding of programming logic as a sequential and systematic process stand out.
16. Problem-solving: Category assigned to mentions the development of skills to find the desired solution of computer programming exercises, by means of verification tests together with the comparison of the solution obtained with expected results.
17. UNCode general failures: Category that contains the references about the problematic aspects of the platform that can interfere with the learning process of the students. The following codes are identified within this category (this enumeration is in accordance with the one made in Table 5): (7) Inefficient grading: Reports on errors or inconsistencies between the numerical rating obtained and the quality of the constructed program. (8) Failures in test cases: Reports on the incorrect execution of the test cases of the exercises presented in the platform, which does not allow students to obtain formative feedback in an effective manner. (9) Tools unavailable: Perceptions about how failures in the functioning of the platform tools, hinder the construction and correction of the code. (10) Incompatibility: Mentions of incompatibility of programs developed in UNCode with other code verification platforms. (11) Loss of information: Reports about occasional loss of information within the platform. (12) Platform down: Opinions where the failure of the platform servers is identified as the main inconvenience, preventing student access. (13) Registration: Perceptions about lack of

clarity and inconveniences in the registration process as a user to use the platform. (14) Processing speed: Mentions about significant delays in the processing of the files uploaded by the students to the platform. (15) Visualization: Reports on failures in the visualization interface of both the executed program and the test cases, which do not allow the acquisition of relevant information for the learning process.

18. Inflexibility of the validations: Category assigned mentions the excessive rigorousness of the platform when validating the solutions built by the students. Specifically, it refers to cases where the platform qualifies as incorrect some programs that meet the objective of the exercise but have minor errors of form.
19. Failure in educational objective: Category containing perceptions that state that the platform is not a meaningful tool for the process of learning and acquiring programming skills.
20. Insufficient feedback: Category assigned to mentions the insufficiency in objectivity and detail of the platform feedback. In this sense, some students mention having a perspective of UNCode as a confusing and unreliable tool.
21. Replaceable tool: Category containing references to the possibility of replacing UNCode's functionalities with other available tools or programs, which may even fulfill the platform's objectives more effectively.

Table 5. Number of units of analysis grouped by codes, categories, and themes of the learning process question

THEME	CATEGORY	CODE	UNIT OF ANALYSIS		
			Approved	Failed	Total
Platform Environment			13	1	14
Benefits of the Platform			6	0	6
	Test cases		9	1	10
	Formative tips		21	0	21
		Guides implementable improvements	13	0	13
	Knowing the errors		80	2	82
	Online availability		8	1	9
	Workspace		4	0	4
	Ease of use		32	0	32
		Ease of writing and correcting code	13	0	13
	Programming languages		5	0	5
	Constant practice		17	3	20
UNCode Tools			33	0	33
	Custom input		30	1	31
	Linter		29	0	29
	Python Tutor		69	4	73
Pedagogical Achievements			1	0	1
	Autonomous learning		13	0	13
	Stimulating exercises		39	1	40
	Optimized evaluation		10	0	10
		Immediate grading	22	0	22

THEME	CATEGORY	CODE	UNIT OF ANALYSIS		
			Approved	Failed	Total
		Objective grading	15	0	15
		Problem and exercise approach	15	0	15
		Submission of academic activities	26	5	31
	Programming skills		30	2	32
	Problem solving		66	1	67
Areas of improvement for programming learning	UNCode general failures		45	0	45
		Inefficient grading	4	0	4
		Failures in test cases	5	0	5
		Tools unavailable	6	1	7
		Incompatibility	5	0	5
		Loss of information	3	0	3
		Platform down	14	2	16
		Registration	0	1	1
		Processing speed	5	2	7
		Visualization	5	1	6
	Inflexibility of the validations		13	2	15
	Failure in educational objective		6	5	11
	Insufficient feedback		10	1	11
	Replaceable tool		1	1	2

Automatic grading question

A total of 4 themes, 8 categories, and 7 codes were generated in response to the question regarding the platform's usefulness for automatic grading of student solutions. Table 6 presents the number of units of analysis grouped by code, category, and general topic, and classified by academic performance. The following categories and codes were assigned:

1. Teacher Involvement: Category assigned to units where students recommend that teachers be included in the platform use, specifically in the academic performance evaluation process.
2. Little use of the platform: Category assigned to a few units that report insufficient experience with the platform, as it is possible that few activities have been developed with UNCode in some groups.
3. Autonomous Learning: Category assigned to units that highlight the platform's ability to promote student learning with minimal intervention from the teacher or monitor. The definition of this category is homologous to that established in the question related to the learning process. However, differences are evident with respect to the identified codes, which allows identifying a greater number of components that contribute to autonomous learning. Firstly, a group of references is identified where it is stated that the use of the platform allows students to self-assess their level of knowledge and skills in the subject, which are grouped with the Code of Evaluation Skills. Secondly, there is Formative Feedback, which refers to the provision of quality information to identify errors, improve, and evaluate the quality of programs developed by students. Finally, there is Grade Tracking, where students highlight UNCode tools that allow statistical control of the grades obtained during the semester in the activities carried out.

4. Optimized evaluation: Category where references related to the characteristics and aspects of the platform that allow for effective and appropriate evaluation of the solutions proposed by students are grouped. The codes generated within the category correspond to Immediate grading, Objective grading, and Immediate feedback.
5. Validation failures: Category assigned to comments that report errors in the validation process of developed programs, as they are marked as wrong despite meeting exercise requirements. Some students specify that the grading and evaluation criteria applied by the platform are too strict and inflexible, ignoring small writing errors and semantics, and resulting in poor grades. These references are grouped with the code Inflexibility in checks.
6. General failures: Category assigned to responses where platform failures and problems are highlighted during use. The definition of this category is similar to the UNCode general failure category in the question related to the learning process. However, the errors reported in this question tend to be less specific.
7. Incomprehensible: References that highlight difficulties in clearly understanding the purpose of automatic grading, specifically its functionality or the information it generates.
8. Insufficient Feedback: References that state that the information provided to students when submitting a solution lacks content and explanation, which does not allow for a full understanding of the provided feedback.

Table 6. Number of units of analysis grouped by codes, categories and themes of the automatic grading question

THEME	CATEGORY	CODE	UNIT OF ANALYSIS		
			Approved	Failed	Total
Implementation difficulties			3	0	3
	Teacher involvement		3	0	3
	Little use of the platform		2	0	2
Ease of use			5	0	5
Pedagogical achievements	Autonomous learning	Evaluation of skills	7	0	7
		Formative feedback	63	1	64
		Grade Tracking	7	0	7
	Optimized evaluation		19	0	19
		Immediate grading	73	2	75
		Objective grading	71	6	77
		Immediate feedback	22	1	23
Areas of improvement for automatic grading	Validation failures		20	0	20
		Inflexibility in checks	20	3	23
	General failures		12	0	12
	Incomprehensible		3	0	3
	Insufficient feedback		2	1	3

Feedback question

Regarding the question on the platform's usefulness for providing automatic feedback on student solutions, a total of 4 themes, 5 categories, and 3 codes were identified. Table 7 shows the number of units of analysis grouped by code, category, and general theme, and classified by academic performance. The categories and codes assigned were:

1. Hidden test cases: Units that highlight instances where the difference between the obtained and expected results is not visible, making it difficult for students to identify errors.
2. Insufficient guidance: Units where it is highlighted that the feedback obtained is not sufficient, since in some cases the information acquired does not allow to specifically identify errors or the way to correct the program. Some causes of this include minor errors that go unnoticed by the platform and lack of clarity in explanations and instructions. The consequence implies an autonomous obligation in the process of correcting the developed programs by the students.
3. Comparison with expected outputs: Units that reference the usefulness of comparing the output generated by the student's program with the expected output to identify errors and correction strategies.
4. Correcting errors: Units where students affirm that clear identification of errors in their code is crucial to understanding the type of mistake made and the most appropriate correction strategies.
5. Specific feedback: Units where feedback generated by the platform is described as highly detailed and specific, aiding in timely problem-solving in programming.

Table 7. Number of units of analysis grouped by codes, categories, and themes of the feedback question

THEME	CATEGORY	CODE	UNIT OF ANALYSIS		
			Approved	Failed	Total
Clear initial conditions			6	2	8
Areas of improvement in feedback			2	0	1
	Hidden test cases		12	0	12
	Insufficient guidance		35	0	35
		Minor details	7	2	9
		Lack of clarity	47	3	50
		Autonomous error identification	12	0	12
Problem solving			157	0	157
	Comparison with expected outputs		33	2	35
	Correcting errors		51	3	54
	Specific feedback		60	2	62
Pointing out errors			89	3	92

During the *axial coding* stage, categories were grouped into general themes for each of the three questions, as can be seen from Tables 5 to 7 (first column). For the learning process question, five general themes were generated including platform environment, benefits of the platform, UNCode tools, pedagogical achievements, and areas for improvement. For the automatic grading question, four general themes were generated including implementation difficulties, ease of use, pedagogical achievements, and areas for improvement. For the feedback question, four general themes were generated including clear initial conditions, areas for improvement in feedback, problem-solving, and pointing out errors. The categories with the highest number of records were identified for each general theme, indicating areas where students had the most positive or negative perceptions of the platform. Overall, the findings suggest that the platform is useful for learning computer programming, but there are areas for improvement in terms of teacher support, clarity of feedback, and the operation of the platform.

Data Analysis and Results

Based on the results of the open and axial coding, the *selective coding* stage was carried out. First, it is found that the UNCode toolset, especially Python tutor, Custom input, and Linter, is the practical basis of the benefits of the platform. In other words, the strengths, and possibilities of UNCode that contribute to the learning of computer programming are represented through the platform's own options. These benefits allow students to obtain pedagogical achievements that students believe are achieved due to the use of UNCode. These pedagogical achievements can be divided into three groups: development of programming skills, autonomous learning, and optimized evaluation.

There is a reciprocal association between the first two achievements. By promoting the development of important programming skills, students acquire capabilities, knowledge, and confidence, which fosters learning processes with little or no intervention from teachers and assistants. UNCode provides students with the means to enhance their programming abilities. The platform's Python tutor tool, for instance, offers a practical environment for practical coding by means of interactive visualizations. This hands-on experience helps students improve their understanding of programming principles and techniques. Additionally, autonomy in learning allows students to consolidate knowledge such as logical thinking and program construction. By allowing students to independently explore programming concepts and experiment with coding, the platform enables them to develop their problem-solving skills and gain a deeper understanding of how to construct effective programs. The third academic achievement relates to the optimization of the evaluation of the programs submitted by students and is represented in two aspects: (1) the simplification of the process of submission of course activities, and (2) the objectivity and speed in the grading of the solution submitted by the student. The objective and immediate grading has become a distinctive feature of UNCode, providing reliability and efficiency to the operation of the platform.

However, there are also aspects that could be improved, which can significantly affect and hinder the student's experience, deteriorating the overall perception and assessment of UNCode. General malfunctions, such as platform crashes when there is a high volume of users connected concurrently, can directly hinder participation in academic activities, affecting the optimization of evaluation, identified as the platform's benefit. Another aspect to improve is the perceived inadequacy of the guidance offered by UNCode. The lack of information or clarity in the feedback can hinder or slow down the processes of autonomous learning. In some cases, insufficient instruction may compromise students' ability to solve programming problems as they do not obtain sufficient information to identify strategies to solve errors. The third aspect to improve is the inflexibility in the validation process. If the platform rates programs as erroneous despite meeting exercise requirements but having minor errors, it can create a sense of failure among students. This perception of harsh grading may discourage learners and undermine their confidence, even when they have made significant progress in their programming skills. Improving these aspects can enhance the student experience and address potential barriers to effective learning. Ensuring platform stability, providing clear and informative feedback, and adopting a more flexible validation process that recognizes and acknowledges students' efforts would contribute to a more positive and supportive learning environment within UNCode.

Finally, a systematic comparison was made between the responses of those who approved (passed) the course and those who did not (failed). This was aimed at evaluating the hypothesis about the effect of passing or failing the course on the perception of the use of UNCode. However, no differences attributable to belonging to either group were found in any of the themes, categories, or codes. At first, it could be stated that the difference between groups is not evident, due to the imbalance in the number of members of each group, but it can be observed that at the discursive level, there are no substantial differences either. Therefore, it can be concluded that the perception and valuation of the platform appear to be independent of the course outcome, suggesting that factors other than course performance influence how students perceive and evaluate UNCode.

PROPOSAL APPLICATION: DISCUSSION

INTEGRATION OF PHASES

The *integration of phases* stage consisted of analyzing together significant correlations results presented in the quantitative data analysis phase with content analysis results described in the qualitative data analysis phase. The measures and metrics can be divided into five categories according to the type of data they represent: obtained verdicts, solution attempts, tool usage, closed questions in the perception questionnaire, and software metrics.

The study found that the feedback generated by verdicts has a positive effect on students' learning process. Students obtained relevant information through verdicts that helped them know errors and develop programming problem-solving skills and promote autonomous learning. They perceived the UNCode platform as an objective and efficient tool for validating constructed programs. Positive correlations between verdicts and academic performance can also be linked to other platforms' benefits, such as formative tips, constant practice, and ease of use. Regarding automatic grading, error verdicts correlated with academic performance may be linked to some platform's pedagogical achievements such as objective grading, immediate grading, and formative and immediate feedback.

Areas of improvement identified in the qualitative phase include minor syntax details, lack of clarity and insufficient guidance, malfunctioning, inflexibility, insufficient feedback, incomprehensible, and validation failure. These categories are similar across different questions and are related to incorrect responses due to minor formatting errors, incomplete or not useful verdicts, and visualization and test case execution issues. Moreover, it is important to consider that these areas for improvement identified through the qualitative analysis can inform the development of new metrics to be considered in the quantitative analysis of future studies that capture aspects that may influence or hinder the user experience in a timely manner.

We also analyzed the relation between the number of solution attempts made by students and various factors of the UNCode platform. The results showed a positive correlation between the total number of attempts made and the academic performance of students. This might be related to the platform's benefits identified by students such as the possibility to practice constantly, online availability, stimulating exercises, workspace, and ease of use. Students who perceived UNCode as an easy-to-use tool tended to use the platform actively by sending a high number of solutions. The platform's constant availability also generated an exclusive workspace for the student, which allowed for constant practice of exercises even outside of class, resulting in a high number of registered attempts. However, the study also identified aspects that some students considered should be improved within the platform, which negatively affected the number of solutions sent. For example, general failures, inflexibility of the validations, and insufficient feedback were identified as obstacles to sending solutions. Incompatibility with programs developed in other external development environments meant that students opted for external tools for program development, evaluation, and correction, and used UNCode only to submit the final program, which limited the number of attempts registered in UNCode.

Regarding the tool usage and the correlation between it and students' academic performance, seven measures show a significant correlation with student performance, with custom input rate having a positive correlation while the rest have a negative correlation. The rate of custom input usage might be related to the references of custom input in the questionnaire; this indicates that students who perceive the option to evaluate programs built with custom tests as a useful tool tend to prefer to use this tool, as they have the skills to design tests to debug the proposed solution and obtain good academic performance. However, the negative correlation found for the other tools and academic performance is opposed to results found in the qualitative analysis, where we found positive students' perceptions with respect to UNCode tools, especially those related to Python tutor and Linter.

The quantitative phase of the study also shows that some students' responses to the closed-ended perception questions in the questionnaire have a significant positive correlation with their academic performance. Specifically, the questions related to the usefulness of UNCode in the learning process, automatic grading, and feedback all had positive correlations. The qualitative phase supports these findings, as most students identified the positive aspects of the platform in their open-ended responses. In particular, more than half of the surveyed students recognized the benefits of using UNCode in programming learning, pedagogical achievements, and promoting problem-solving. The use of custom input, identified as a tool in the learning process question, also had a significant positive correlation with academic performance.

In the final analysis of the software metrics, the quantitative phase showed that three metrics – token count, lines of code, and cyclomatic complexity – had a positive correlation with academic performance, while the maintainability index (MI) had a negative correlation. It is possible that this positive correlation is due to students who developed longer programs in terms of tokens, lines of code, and the number of possible paths within the program execution. However, the findings from the qualitative analysis did not provide such technical details in relation to software metrics, making it difficult to integrate them with the quantitative results.

Interpretation

The research question of this proposal application on how mixed research methods applied in learning analytics can enhance the understanding of the relationships between variables generated throughout the learning process and the academic performance of students in computer programming can be answered through the integrated results summarized below.

Our findings suggest that students' academic performance is positively correlated with the number of accepted programs (correct responses), success rate, the amount of exceeded memory limit errors, compilation errors, verdicts, and exceeded time limit rates. Considering the perceptions about the platform as a source of formative feedback, it is possible to conclude that these verdicts not only permitted students to identify errors but also provided guidance for correcting the constructed program, which generated problem-solving skills and autonomous learning. This indicates that students possibly acquire sufficient knowledge to successfully solve course activities, which is reflected positively in academic performance. These results support previous research findings that the accumulated percentage of correct exercises has a significant correlation coefficient of 0.67 with student academic performance (Azcona et al., 2019). Additionally, our study found that the positive correlation of the number of incorrect responses (Wrong_Answer) might be related to test case references and comparison with expected outputs, indicating that the use of standardized tests for automatic program evaluation is effective as formative feedback, benefiting student academic performance.

Regarding the use of UNCode's tools, despite the negative results found in the correlation analysis regarding academic performance, these negative correlations are refuted by references in the questionnaire responses that identify Python tutor, custom input, and linter as contributing elements within the platform and as benefits of the platform. These results are also in line with previous findings of studies conducted by Restrepo-Calle et al. (2020) and Ramírez-Echeverry et al. (2022), where students' perceptions of UNCode's use as a learning support platform were also analyzed. Within these investigations, it is evident that students recognize the visualization tool of code execution (Python tutor) as an added value of the platform, which is associated with the identification and correction of errors. Additionally, students highlight the tools for verification of good programming practices (Linter) and tests with customized inputs (Custom input). Moreover, the use of user statistics has a non-significant correlation, which is consistent with research conducted by Zacharis (2015), and Macfadyen and Dawson (2010), where the number of accesses to the grading tool does not show a significant correlation with the grade. This result can be related to a small percentage of responses that indicate monitoring grades as part of their pedagogical achievements, suggesting that the

majority of students is not aware of monitoring their academic achievements and therefore has no noticeable impact on the learning process.

Furthermore, the total number of attempts made by students has a positive correlation with their final grade, which can be attributed to positive aspects perceived by students, such as stimulating exercises, ease of use, constant practice, online availability, workspace, and platform environment. The relationship between these results suggests that high-performing students may use the platform as a source of feedback to improve their solutions by making multiple attempts at the same activity. The platform provides a workspace that facilitates the presentation of academic activities, is user-friendly, allows for the creation of stimulating exercises, and encourages constant practice since it is available even outside the classroom. This result is consistent with Zacharis' (2015) research, which found a positive correlation (0.2 to 0.39) between the number of activities submitted during the course and the final grade.

Moreover, the positive correlation of time between attempts can be associated with the group of students who highlight the immediate feedback and the writing and correction aspect as positive characteristics of UNCode. These results are consistent with the findings of Andergassen et al. (2014), who obtained a positive correlation of 0.18 between the average time difference between repetitions (i.e., attempts) of exercises and the final exam grade. These results indicate that due to the speed of the evaluation process on the platform, students with good academic performance possibly invest most of their time in building and correcting the solution between each submission. Another of the results obtained is the positive correlations of perceptions in the three closed questions (QUESTION: Learning process, QUESTION: Automatic grading, and QUESTION: Feedback), which are corroborated by the answers to the open questions, where most students identify pedagogical achievements and the benefits of the platform. This probably indicates that users who have a positive experience with the platform tend to identify and take advantage of its benefits, achieving good academic performance.

Our findings of a positive correlation between academic performance and the cyclomatic complexity metric contradict the results of Vahdat et al. (2015), which found a negative correlation between the two variables. Furthermore, previous works in this context have shown no significant correlation between these variables (Castellanos et al., 2017). Therefore, it is necessary to further explore these relationships to improve our understanding of this situation.

The results of this research provide insights into how automatic formative feedback can be beneficial to the learning process for students. Nevertheless, some students highlight that this type of feedback needs to be complemented with instructor guidance to achieve their objectives. Furthermore, allowing students to design personalized tests appears to be a useful approach for constructing correct solutions. The study also found that high-scoring students tend to make the most attempts and use the majority of their time correcting their programs. Additionally, it emphasizes the importance of ensuring that students understand the platform's utility in the class methodology to increase the likelihood that they will take advantage of the tool and improve their academic performance.

In terms of answering the research question posed, the results obtained show that the use of mixed methods allows the results of the quantitative phase to be complemented by observations from the qualitative phase. In this sense, in most cases, the qualitative data, which correspond to the students' perceptions, corroborate, and expand upon the results of the quantitative data. The agreement between the results of both phases allows for generating several hypotheses about the underlying reasons for the observed behaviors and the learning processes of the students, which are based on both quantitative and qualitative results. In other cases, the mixed approach reveals contradictions between the results of both phases (e.g., results of tool use), which allows for identifying topics of interest beyond the scope of the research and generating new questions that can be addressed in future works. In other words, the application of the framework presented in this paper demonstrated that a mixed

methods approach to understanding the study question was superior to the use of a quantitative methodology alone.

Finally, it is worth noting that the application of the framework presented has some limitations, such as an imbalance in the dataset used in both the quantitative and qualitative phases. Specifically, the number of students with grades above the minimum passing grade is much higher than the number of students who fail the course, which could affect the magnitude of the correlations obtained differentiated by passing and failing categories. Moreover, there is a limitation associated with the high dispersion of the data on the total number of attempts made and the average time between attempts, due to the variety of activities carried out in the different groups of the programming course. Some instructors propose more hands-on workshops, reinforcement exercises, or projects with flexible deadlines, while others focus on assessing students' knowledge through short tests and exams, which usually have a limited time frame. This means that the behaviors and strategies that students use during their learning process can vary significantly depending on the type of activity they are exposed to. Finally, the students provided suggestions for improving the formative feedback, which should be considered to enhance the functionality and usability of the tool.

Hypothesis Generation

Based on the integration of results from both phases of the mixed methods approach used in the research and their interpretation, ideas for possible future work arise that can expand the discoveries of the current study. Firstly, it is possible to hypothesize that UNCode as a course tool may have a significant impact on the average final grades of students using the tool, particularly in a programming course. A comparative analysis between students using the UNCode platform and those enrolled in a similar course where the platform is not used could validate this hypothesis. To investigate this further, a quasi-experimental study design could be implemented with an experimental group consisting of students who use the tool and a control group consisting of students who do not use the tool.

On the other hand, the significant correlations evidenced can promote the design and execution of educational interventions within the course, corresponding to the final stage of the cyclic learning analytics methodology proposed by Carter et al. (2019). The development of interventions consists of making decisions in the studied educational context, where information, guidance, or feedback is shared with the students with the aim of positively influencing their behavior (Carter et al., 2019). In this context, it is plausible to hypothesize that an intervention designed to increase the visibility of error verdict descriptions, accompanied by additional instructions for error correction, could significantly improve the perceptions of students who perceive the feedback they receive as insufficient. Furthermore, suggesting the use of specific tools based on their functionality, such as recommending the use of Python Tutor to address runtime execution errors, may effectively encourage students to engage with platform tools, resulting in improved perceptions of the feedback process and possibly even improved academic performance. The impact of these interventions can also be evaluated through an experimental design that seeks statistical differences between a group that implements one of the interventions and a control group.

In the context of introductory programming education, AI/large language models have the potential to revolutionize teaching by enhancing the learning experience, providing personalized support, and enabling more efficient assessment and feedback mechanisms. Future research in this area is to implement the proposed framework on data from an introductory programming course using these models.

Conclusions and Future Works

This article proposes a sequential explanatory mixed-methods design for learning analytics, consisting of three main phases: (1) preparation, transformation, and analysis of quantitative data; (2) collection

and content analysis of qualitative data; and (3) integration of results from both phases and discussion/interpretation of the findings. This framework was applied to historical quantitative and qualitative data from students' use of an automated feedback and evaluation platform for programming exercises in a programming course at the National University of Colombia. The answer to the research question posed corresponds to the fact that the results obtained demonstrate that the mixed methods effectively complement quantitative and qualitative data. Qualitative data, representing students' perceptions, generally support and extend the quantitative data. The consistency between the two phases allows hypotheses to be generated about student behavior and learning processes based on both types of data.

Specifically, the relationship between students' use of the programming tool and their academic performance was examined. Results indicate that students who expressed the highest level of agreement with the tool's usefulness for learning and who appreciated the ability to automatically evaluate their programs and receive feedback (qualitative data) tended to have better academic performance (quantitative data). This suggests that the formative feedback allowed students to identify errors and provided guidance for correcting the constructed program, which generated problem-solving skills and autonomous learning that enabled students to successfully complete course activities, which was positively reflected in academic performance. In addition, students who emphasized the benefits of the tool (qualitative data) achieved better academic performance (quantitative data). First, they found it valuable for identifying errors in their programs and providing corrective feedback. Second, the tool gave them the opportunity to practice programming autonomously and develop their problem-solving skills. The exercises provided were found to be challenging and stimulating, which motivated the students to learn and increased their curiosity. Students also appreciated the tool's objective and immediate grading system. All of this suggests that users who have a positive experience with the platform are more likely to recognize and take advantage of its benefits and achieve good academic performance.

The main contribution of this work is the proposed methodological framework for the application of learning analytics in computer programming courses, which is based on mixed methods and specifies activities from data collection, both quantitative and qualitative, to results integration and discussion. It is worth noting that the methodological activities are described in a general manner, providing a reference for future research in similar contexts. The use of mixed methods allows for the complementation, corroboration, or refutation of quantitatively evidenced results with qualitative data, and the generation of hypotheses about possible causes or explanations of students' behaviors. Specifically, the framework used in this approach helped to formulate hypotheses that describe different aspects of the learning processes that occur in a computer programming educational environment. Based on these hypotheses, several future research projects and works were proposed.

Although a limitation of the presented study is that the framework was only demonstrated in the context of its use for learning computer programming, we suggest that future research implement the proposed framework in different educational contexts and populations to strengthen the obtained results, complement the proposed methodology, or address identified issues. Potential lines of research to continue this work include (1) implementing the proposed mixed-method learning analytics methodology on a different population sample, such as students from other universities; (2) using techniques to correct unbalanced data sets, such as fine-tuning algorithms, resampling, or random over/under sampling in learning analytics studies; (3) analyzing students' interactions with the UNCode platform and their academic activities, such as exams, quizzes, workshops, projects, and assignments, in correlation with their activity grades rather than their final course grade; and (4) using the findings on students' behaviors and perceptions of the UNCode platform to design interventions that positively affect their academic performance, with an experimental design to statistically evaluate the effect.

REFERENCES

Aissa, M., Al-Kalbani, M., Al-Hatali, S., & BinTouq, A. (2020). Novice learning programming languages in Omani higher education institution (Nizwa University) issues, challenges and solutions. In A. Al-Masri, & Y. Al-Assaf (Eds.), *Sustainable development and social responsibility, Volume 2* (pp. 143–148). Springer. https://doi.org/10.1007/978-3-030-32902-0_18

Aljohani, N. R., Daud, A., Abbasi, R. A., Alowibdi, J. S., Basheri, M., & Aslam, M. A. (2019). An integrated framework for course adapted student learning analytics dashboard. *Computers in Human Behavior*, *92*, 679–690. https://doi.org/10.1016/j.chb.2018.03.035

Andergassen, M., Mödritscher, F., & Neumann, G. (2014). Practice and repetition during exam preparation in blended learning courses: Correlations with learning results. *Journal of Learning Analytics*, *1*(1), 48–74. https://doi.org/10.18608/jla.2014.11.4

Arnold, K. E., & Pistilli, M. D. (2012). Course signals at Purdue: Using learning analytics to increase student success. *Proceedings of the 2nd International Conference on Learning Analytics and Knowledge* (pp. 267–270). Association for Computing Machinery. https://doi.org/10.1145/2330601.2330666

Azcona, D., Hsiao, I.-H., & Smeaton, A. F. (2019). Detecting students-at-risk in computer programming classes with learning analytics from students' digital footprints. *User Modeling and User-Adapted Interaction*, *29*, 759–788. https://doi.org/10.1007/s11257-019-09234-7

Baker, R. S., & Inventado, P. S. (2014). Educational data mining and learning analytics. In J. Larusson, R. White (Eds.), *Learning analytics* (pp. 61–75). Springer. https://doi.org/10.1007/978-1-4614-3305-7_4

Barber, R., & Sharkey, M. (2012). Course correction: using analytics to predict course success. *Proceedings of the 2nd International Conference on Learning Analytics and Knowledge* (pp. 259–262). Association for Computing Machinery. https://doi.org/10.1145/2330601.2330664

Bryman, A. (2015). Mixed methods research: Combining quantitative and qualitative research. In A. Bryman, *Social research methods* (pp. 634-660). Oxford University Press.

Carter, A., Hundhausen, C., & Olivares, D. (2019). Leveraging the integrated development environment for learning analytics. In S. Fincher, & A. Robins (Eds.), *The Cambridge handbook of computing education research* (pp. 679-706). Cambridge University Press. https://doi.org/10.1017/9781108654555.024

Castellanos, H., Restrepo-Calle, F., González, F. A., & Ramírez Echeverry, J. J. (2017, October). Understanding the relationships between self-regulated learning and students source code in a computer programming course. *Proceedings of the IEEE Frontiers in Education Conference, Indianapolis, IN, USA*, 1–9. https://doi.org/10.1109/FIE.2017.8190467

Clow, D. (2012). The learning analytics cycle: Closing the loop effectively. *Proceedings of the 2nd International Conference on Learning Analytics and Knowledge* (pp. 134–138). Association for Computing Machinery. https://doi.org/10.1145/2330601.2330636

Coffrin, C., Corrin, L., de Barba, P., & Kennedy, G. (2014). Visualizing patterns of student engagement and performance in MOOCs. *Proceedings of the Fourth International Conference on Learning Analytics and Knowledge* (pp. 83–92). Association for Computing Machinery. https://doi.org/10.1145/2567574.2567586

Corbin, J. M., & Strauss, A. (1990). Grounded theory research: Procedures, canons, and evaluative criteria. *Qualitative Sociology*, *13*, 3–21. https://doi.org/10.1007/BF00988593

Creswell, J. W. (2014). The selection of a research approach. In J. W. Creswell, *Research design: Qualitative, quantitative and mixed methods approaches* (pp. 3-24). Sage Publications.

Elia, G., Solazzo, G., Lorenzo, G., & Passiante, G. (2019). Assessing learners' satisfaction in collaborative online courses through a big data approach. *Computers in Human Behavior*, *92*, 589–599. https://doi.org/10.1016/j.chb.2018.04.033

Hernández-Sampieri, R., Fernández-Collado, C., & Baptista-Lucio, P. (2014). Proceso de la investigación cualitativa [Qualitative research process]. In R. Hernández-Sampieri, C. Fernández-Collado, & P. Baptista-Lucio, *Metodología de la investigación* (pp. 355-466). McGraw-Hill.

Hilliger, I., Ortiz-Rojas, M., Pesántez-Cabrera, P., Scheihing, E., Tsai, Y.-S., Muñoz-Merino, P. J., Broos, T., Whitelock-Wainwright, A., & Pérez-Sanagustín, M. (2020). Identifying needs for learning analytics adoption in Latin American universities: A mixed-methods approach. *The Internet and Higher Education*, *45*, 100726. https://doi.org/10.1016/j.iheduc.2020.100726

Ihantola, P., Vihavainen, A., Ahadi, A., Butler, M., Börstler, J., Edwards, S. H., Isohanni, E., Korhonen, A., Petersen, A., Rivers, K., Rubio, M. A., Sheard, J., Skupas, B., Spacco, J., Szabo, C., & Toll, D. (2015). Educational data mining and learning analytics in programming: Literature review and case studies. *Proceedings of the 2015 ITiCSE on Working Group Reports* (pp. 41–63). Association for Computing Machinery. https://doi.org/10.1145/2858796.2858798

Kizilcec, R. F., Pérez-Sanagustín, M., & Maldonado, J. J. (2017). Self-regulated learning strategies predict learner behavior and goal attainment in Massive Open Online Courses. *Computers & Education*, *104*, 18–33. https://doi.org/10.1016/j.compedu.2016.10.001

Kop, R., Fournier, H., & Durand, G. (2017). A critical perspective on learning analytics and educational data mining. In C. Lang, G. Siemens, A. F. Wise, & D. Gašević (Eds.), *The handbook of learning analytics* (pp. 319–326). Society for Learning Analytics Research. https://doi.org/10.18608/hla17.027

Kumar, V. S., Kinshuk, Somasundaram, T. S., Boulanger, D., Seanosky, J., & Vilela, M. F. (2015). Big data learning analytics: A new perspective. In Kinshuk, & R. Huang (Eds.), *Ubiquitous learning environments and technologies* (pp. 139–158). Springer. https://doi.org/10.1007/978-3-662-44659-1_8

Ladias, A., Karvounidis, T., & Ladias, D. (2022). Forms of communications in scratch and the SOLO taxonomy. *Advances in Mobile Learning Educational Research*, *2*(1), 234–245. https://doi.org/10.25082/AMLER.2022.01.007

Lagus, J., Longi, K., Klami, A., & Hellas, A. (2018). Transfer-learning methods in programming course outcome prediction. *ACM Transactions on Computing Education*, *18*(4), Article 19. https://doi.org/10.1145/3152714

Lazarinis, F., Karatrantou, A., Panagiotakopoulos, C., Daloukas, V., & Panagiotakopoulos, T. (2022). Strengthening the coding skills of teachers in a low dropout Python MOOC. *Advances in Mobile Learning Educational Research*, *2*(1), 187–200. https://doi.org/10.25082/AMLER.2022.01.003

Long, P., & Siemens, G. (2011). Penetrating the fog: Analytics in learning and education. *EDUCAUSE Review*, *September/October*, 31–40. https://er.educause.edu/articles/2011/9/penetrating-the-fog-analytics-in-learning-and-education

Lonn, S., Aguilar, S. J., & Teasley, S. D. (2015). Investigating student motivation in the context of a learning analytics intervention during a summer bridge program. *Computers in Human Behavior*, *47*, 90–97. https://doi.org/10.1016/j.chb.2014.07.013

Lu, O. H. T., Huang, J. C. H., Huang, A. Y. Q., & Yang, S. J. H. (2017). Applying learning analytics for improving students engagement and learning outcomes in an MOOCs enabled collaborative programming course. *Interactive Learning Environments*, *25*(2), 220–234. https://doi.org/10.1080/10494820.2016.1278391

Macfadyen, L. P., & Dawson, S. (2010). Mining LMS data to develop an "early warning system" for educators: A proof of concept. *Computers & Education*, *54*(2), 588–599. https://doi.org/10.1016/j.compedu.2009.09.008

Mangaroska, K., & Giannakos, M. (2017). Learning analytics for learning design: Towards evidence-driven decisions to enhance learning. In É. Lavoué, H. Drachsler, K. Verbert, J. Broisin, & M. Pérez-Sanagustín (Eds.), *Data driven approaches in digital education* (pp. 428–433). Springer. https://doi.org/10.1007/978-3-319-66610-5_38

Margulieux, L. E., Morrison, B. B., & Decker, A. (2020). Reducing withdrawal and failure rates in introductory programming with subgoal labeled worked examples. *International Journal of STEM Education*, *7*, Article 19. https://doi.org/10.1186/s40594-020-00222-7

Monllaó Olivé, D., Huynh, D. Q., Reynolds, M., Dougiamas, M., & Wiese, D. (2020). A supervised learning framework: Using assessment to identify students at risk of dropping out of a MOOC. *Journal of Computing in Higher Education*, *32*, 9–26. https://doi.org/10.1007/s12528-019-09230-1

Pistilli, M. D., Willis, J. E., & Campbell, J. P. (2014). Analytics through an institutional lens: Definition, theory, design, and impact. In J. Larusson, & B. White (Eds.), *Learning analytics* (pp. 79–102). Springer. https://doi.org/10.1007/978-1-4614-3305-7_5

Ramírez-Echeverry, J. J., Restrepo-Calle, F., & González, F. A. (2022). A case study in technology-enhanced learning in an introductory computer programming course. *Global Journal of Engineering Education*, *24*(1), 65–71. http://www.wiete.com.au/journals/GJEE/Publish/vol24no1/10-Restrepo-Calle-F.pdf

Restrepo-Calle, F., Ramírez-Echeverry, J. J., & Gonzalez, F. A. (2018, July). UNCode: Interactive system for learning and automatic evaluation of computer programming skills. *Proceedings of the 10th International Conference on Education and New Learning Technologies*, *Palma, Spain,* 6888–6898. https://doi.org/10.21125/edulearn.2018.1632

Restrepo-Calle, F., Ramírez-Echeverry, J. J., & González, F. A. (2020). Using an interactive software tool for the formative and summative evaluation in a computer programming course: An experience report. *Global Journal of Engineering Education*, *22*(3), 174–185. http://www.wiete.com.au/journals/GJEE/Publish/vol22no3/06-Echeverry-J.pdf

Rienties, B., & Toetenel, L. (2016). The impact of 151 learning designs on student satisfaction and performance: Social learning (analytics) matters. *Proceedings of the Sixth International Conference on Learning Analytics & Knowledge* (pp. 339–343). Association for Computing Machinery. https://doi.org/10.1145/2883851.2883875

Salguero, A., Griswold, W. G., Alvarado, C., & Porter, L. (2021). Understanding sources of student struggle in early computer science courses. *Proceedings of the 17th ACM Conference on International Computing Education Research* (pp. 319–333). Association for Computing Machinery. https://doi.org/10.1145/3446871.3469755

Shen, H., Liang, L., Law, N., Hemberg, E., & O'Reilly, U.-M. (2020). Understanding learner behavior through learning design informed learning analytics. *Proceedings of the Seventh ACM Conference on Learning @ Scale* (pp. 135–145). Association for Computing Machinery. https://doi.org/10.1145/3386527.3405919

Siemens, G. (2013). Learning analytics: The emergence of a discipline. *American Behavioral Scientist*, *57*(10), 1380–1400. https://doi.org/10.1177/0002764213498851

Tempelaar, D. T., Rienties, B., & Giesbers, B. (2016). Verifying the stability and sensitivity of learning analytics based prediction models: An extended case study. In S. Zvacek, M. Uhomoibhi, & M. Helfert (Eds.), *Computer supported education* (pp. 256–273). Springer. https://doi.org/10.1007/978-3-319-29585-5_15

Vahdat, M., Oneto, L., Anguita, D., Funk, M., & Rauterberg, M. (2015). A learning analytics approach to correlate the academic achievements of students with interaction data from an educational simulator. In G. Conole, T. Klobučar, C. Rensing, J. Konert, & E. Lavoué (Eds.), *Design for teaching and learning in a networked world* (pp. 352–366). Springer. https://doi.org/10.1007/978-3-319-24258-3_26

Wu, Y., & Wu, W. (2018). A learning analytics system for cognition analysis in online learning community. In L. H. U, & H. Xie (Eds.), *Web and big data* (pp. 243–258). Springer. https://doi.org/10.1007/978-3-030-01298-4_21

Zacharis, N. Z. (2015). A multivariate approach to predicting student outcomes in web-enabled blended learning courses. *The Internet and Higher Education*, *27*, 44–53. https://doi.org/10.1016/j.iheduc.2015.05.002

AUTHORS

Edna Johanna Chaparro Amaya obtained her M.Sc. in Systems and Computing Engineering from Universidad Nacional de Colombia in 2023, following her undergraduate degree in Environmental and Chemical Engineering from Universidad de los Andes, Colombia, in 2017. Her research interests include engineering education and learning analytics.

Felipe Restrepo-Calle received his Ph.D. (cum laude) from the University of Alicante, Spain, in 2011. He worked as a postdoctoral researcher at the University of Seville, Spain, in 2012 and 2013. Since 2014, he has been working at the Department of Systems and Industrial Engineering at the National University of Colombia (Universidad Nacional de Colombia), Bogotá, Colombia, where he is an associate professor and head of the Programming Languages and Systems (PLaS) research group. His research interests include programming languages, dependable design in embedded systems, and engineering education.

Jhon Jairo Ramírez-Echeverry received his Bachelor's degree in Electronics Engineering from the Universidad Nacional de Colombia, Manizales (Caldas), Colombia, the M.Sc. degree in Telecommunications Engineering from the Universidad Nacional de Colombia, Bogotá, Colombia, and the Ph.D. degree (cum laude) in Engineering of Projects and Systems from the Universitat Politècnica de Catalunya, BarcelonaTech, Spain, in 2017. He is currently an Associate Professor in the Department of Electrical and Electronic Engineering at the Universidad Nacional de Colombia, Bogotá, Colombia. His research interests are engineering education (self-regulated learning) and electronic telecommunication systems.