IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 20 June 2025, accepted 26 July 2025, date of publication 15 August 2025, date of current version 22 August 2025.

Digital Object Identifier 10.1109/ACCESS.2025.3599357

RESEARCH ARTICLE

Theory Inspires, but Examples Engage: A Mixed-Methods Analysis of Worked Examples From CoderBot in Programming Education

RENATO DE SOUZA GARCIA[1], JOÃO EMILIO ANTONIO VILLA[1], ANDRE LUIZ MENDES MIRANDA[1], GILLEANES THORWALD ARAUJO GUEDES[1], ANA CAROLINA ORAN[2], PAULO SILAS SEVERO DE SOUZA[1], RICARDO FERREIRA VILELA[3], PEDRO HENRIQUE DIAS VALLE[4], AND WILLIAMSON SILVA[5]

[1]Federal University of Pampa, Alegrete 97546-550, Brazil
[2]Federal University of Amazonas, Manaus 69067-005, Brazil
[3]State University of Campinas (UNICAMP), Campinas 13083-970, Brazil
[4]University of São Paulo, São Paulo 05508-220, Brazil
[5]Federal University of Cariri, Juazeiro do Norte 63048-080, Brazil

Corresponding author: Williamson Silva (williamson.silva@ufca.edu.br)

This work was supported in part by CAPES, Brazil, under Finance Code 001, and in part by the CAPES Transformative Agreement.

ABSTRACT Programming has become increasingly important in our society. However, the learning process presents significant challenges, particularly for novice students of introductory courses. From the students' perspective, programming concepts are often perceived as complex and challenging to understand. Chatbots have emerged as promising and effective pedagogical agents, offering continuous support and personalized feedback throughout the programming learning process. In this paper, we present CoderBot, a pedagogical agent grounded in Example-Based Learning designed to assist novice students in comprehending programming concepts using correct and erroneous practical examples. To evaluate the self-efficacy and acceptance of CoderBot in the classroom, we conducted an exploratory study involving 103 undergraduate students from several regions of our country, all of whom were enrolled in introductory programming courses. The quantitative findings highlight the ease of use associated with CoderBot, along with noticeable improvements in students' understanding of programming concepts and increased levels of motivation and self-confidence. Moreover, the qualitative results indicate that CoderBot holds the potential to be an effective pedagogical agent for supporting programming instruction, particularly in terms of clarity, accessibility, and ongoing assistance. However, the findings also suggest the need for further expansion of the available examples and improvements in the clarity of responses to realize the tool's educational potential fully. These results offer valuable insights into integrating chatbots within academic environments, underscoring the role such tools can play in enhancing the learning experience for programming students.

INDEX TERMS Programming education, pedagogical agents, coderbot, example-based learning, empirical study, self-efficacy, acceptance.

I. INTRODUCTION

Coding skills have become indispensable for technology professionals and individuals seeking to enhance their technical competencies [1]. This trend is reflected in the increasing number of students enrolling in Science,

The associate editor coordinating the review of this manuscript and approving it for publication was Ka Wai Gary Wong.

Technology, Engineering, and Mathematics programs [2]. However, learning to program often presents significant challenges, particularly in the early stages of the course [1]. Consequently, approval rates in introductory programming courses are historically low [3], contributing to high retention and dropout rates [4]. Instructors report difficulties in effectively teaching these concepts [5], which may stem from students' lack of prior exposure to programming, limited experience with practical coding tasks, lack of motivation, and insufficient individualized support [6].

In response to these challenges, conversational agents like chatbots have emerged as promising educational tools, particularly in programming education. Chatbots offer students the opportunity to progress more autonomously, resolve specific doubts [7], [8], and receive immediate feedback — a critical advantage in reducing their reliance on direct instructor support [9], [10]. This instant feedback enhances the learning process by facilitating real-time correction and reflection. Additionally, chatbots provide personalized, motivational feedback, enabling students to engage in programming practice through interactive exercises and learning at their pace [11].

Despite these benefits, many chatbots currently used in programming education suffer from notable limitations, including outdated or inappropriate datasets, which can compromise the accuracy of the information provided [12]. Excessive formality in language and a lack of pedagogical focus often create barriers for novice learners [13]. Furthermore, many chatbots prioritize technical productivity over educational value, providing ready-made solutions rather than fostering the development of logical reasoning and computational thinking [4]. This emphasis on syntax over conceptual understanding can demotivate students as they struggle with the rigid demands of programming language structure [14]. Thus, there is an apparent demand for more adaptable and intuitive educational chatbots that prioritize the learning needs of beginners and support their programming skills development in a more accessible and didactic manner [4].

In this context, this paper introduces CoderBot, an educational agent designed to support the teaching of programming concepts through Example-Based Learning (EBL). CoderBot employs correct and erroneous practical examples to enhance students' comprehension of programming tasks. EBL was selected as the theoretical foundation of CoderBot due to its alignment with Cognitive Load Theory, which posits that learners require structured instructional models to effectively internalize and understand new content [15]. CoderBot, therefore, emerges as an innovative educational technology that promotes novel teaching and learning methodologies for both instructors and students while improving the quality and efficiency of the learning experience.

The study described in this paper involved 103 undergraduate students from several regions of our country, all enrolled in introductory programming courses. The study evaluated both the acceptance of the CoderBot technology and the students' perceived self-efficacy. The Technology Acceptance Model (TAM) [16] was adopted to assess technology acceptance, while self-efficacy was measured using a domain-specific self-efficacy questionnaire [17]. In addition to quantitative measures, students provided open-ended responses, which were analyzed using coding procedures to interpret the qualitative data. This mixed-methods approach allowed for a comprehensive understanding of the students' perceived challenges and benefits, demonstrating CoderBot's potential as an effective and accessible tool for supporting programming education.

This paper is organized as follows. Section II presents the main concepts associated with this paper and the related work. Section III describes our proposal. Section IV describes the planning of the exploratory study. Section VI discusses our lessons learned and the new version of our proposal. Finally, Section VIII presents the study's final considerations.

II. BACKGROUND AND RELATED WORK

As technology advances and the demand for diversified learning options grows, the need for innovative educational tools such as chatbots has become increasingly apparent [18]. As valuable educational resources, Chatbots allow students to regulate their learning pace while providing continuous access to study materials. By empowering students to take ownership of their learning journey, these tools enhance their learning experience [11]. Recent research underscores that these technologies support the learning process, improve effectiveness, and foster active student engagement [11], [19]. Chatbots integrated into applications, websites, and messaging platforms assist users in performing tasks, clarifying concepts, and offering tailored support, guidance, and solutions based on individual needs [11]. In addition to providing constructive feedback, chatbots facilitate self-assessment and enhance the development of skills and competencies, making learning more intuitive and collaborative [11], [20]. However, despite these numerous advantages, it is essential to emphasize that chatbots were designed to aid — rather than replace—instructors, who remain indispensable in the educational process [11].

In this context, several studies have investigated the transformative role of chatbots in education, highlighting their potential to enhance both learning experiences and student engagement. For example, Hobert [21] developed and evaluated a chatbot named Coding Tutor, which provides individualized support by offering explanations, tips, and feedback on students' source code. A usability study involving 40 undergraduate students of an information systems course revealed that the chatbot was perceived as a valuable complement to traditional programming instruction, mainly when instructor assistance was not readily available.

Similarly, Iqbal Maliket et al. [22] developed a chatbot to assist programming students with their tasks and learning processes. Their study analyzed the influence of the chatbot on student performance, measured by final course grades.

Results indicated that the chatbot effectively aided students in grasping fundamental programming concepts and in identifying common semantic and syntactic errors.

Carreira et al. [23] introduced Pyo, a chatbot to support novice programmers focusing on the Python programming language. Pyo provides personalized assistance to help students better understand programming concepts and identify mistakes in their code. The authors reported that students found Pyo highly beneficial in complementing their learning process, particularly in providing individualized support, facilitating conceptual understanding, and helping them resolve coding errors. The authors stated that positive interactions with Pyo helped students overcome common difficulties faced by novice programmers.

Finally, Kasinathan et al. [24] developed TicTad, a chatbot designed to support students with learning difficulties or those interested in learning the C# language. TicTad provides an interactive and practical approach to memorizing concepts, reducing the time and effort required compared to traditional learning methods. Tested by 30 beginner students, TicTad received positive feedback, with participants describing it as both fun and educational. The study concluded that TicTad met its target users' needs by creating an engaging and supportive learning environment for learning C#.

Although prior studies highlight the potential of chatbot programming education, a significant gap remains in how pedagogical theories, such as EBL and Cognitive Load Theory are systematically integrated into their design. Most existing tools emphasize functionality, code generation, or basic guidance but lack structured pedagogical strategies to scaffold novice learners' understanding.

In this context, we propose the CoderBot, a pedagogical agent that aims to bridge this gap by explicitly embedding EBL and Cognitive Load Theory into its architecture. It employs correct and incorrect worked examples, structured via pedagogically grounded templates, to support step-by-step reasoning and error reflection—techniques known to foster deep conceptual understanding while managing cognitive load. To illustrate how CoderBot differs from other tools in the field, Table 1 compares key features of CoderBot with representative educational chatbots (e.g., PyO, Coding Tutor, TicTad) and generative AI-based assistants (e.g., GPT, Copilot, Gemini).

The comparison covers dimensions such as:

- **Pedagogical Model**: Describes the underlying theoretical and pedagogical framework that guides the tool's instructional design for programming education;
- **Use of Cognitive Load Theory**: Specifies whether the tool explicitly incorporates principles from Cognitive Load Theory to reduce extraneous load and optimize learners' mental effort during task execution;
- **Correct vs Incorrect Examples**: Indicates whether the tool provides both correct and incorrect code examples to promote learning through guided error analysis and reflection;
- **Personalization**: Assesses the tool's ability to tailor its content, feedback, or learning pathways to individual student profiles, considering prior knowledge, progress, or learning preferences;
- **Handling of Complex Questions**: Evaluates the tool's capacity to address complex or open-ended programming problems that extend beyond basic or predefined instructional scenarios;
- **Feedback Type**: Characterizes the form and depth of feedback provided by the tool, such as contextualized error messages, step-by-step explanations, or natural language dialogue;
- **Open-ended Input Support**: Determines the extent to which the tool allows students to input queries in natural language, as opposed to using only fixed options or structured commands; and,
- **Platform**: Specifies the deployment environment or interface through which the tool is accessed, such as web applications, desktop software, IDE extensions, or cloud-based APIs.

Table 1 highlights that, despite the increasing availability of educational chatbots and AI-powered assistants in programming education, most existing tools prioritize technical support over pedagogical intentionality. These tools often lack structured instructional design and fail to explicitly incorporate learning theories that scaffold novice learners' conceptual development. In contrast, CoderBot offers pedagogically grounded scaffolding aligned with students' cognitive processes, explicitly operationalizing Example-Based Learning and Cognitive Load Theory. Rather than functioning as a general-purpose assistant, CoderBot emerges as a purpose-built educational ally—designed to foster meaningful, theory-driven learning experiences in programming instruction. This comparative analysis reinforces CoderBot's unique position within the current landscape, addressing unmet pedagogical needs in a domain still dominated by function-oriented tools.

III. CODERBOT

The CoderBot is a pedagogical agent developed to assist students in learning programming by enhancing both instructional efficiency and engagement. We designed CoderBot to benefit both instructors and students. CoderBot facilitates the understanding of programming content through a practical and interactive approach. We grounded CoderBot on principles of EBL [25], [26], [27]. CoderBot streamlines instruction by minimizing information overload, allowing students to concentrate on the problem's essential aspects [27].

EBL, when aligned with the principles of Cognitive Load Theory, effectively leverages demonstration to guide students in mastering specific tasks or skills [26]. By observing a task successfully executed, students develop confidence in their ability to replicate similar outcomes, which fosters a positive belief in their capabilities [25]. This approach helps reduce cognitive load by providing a structured framework

TABLE 1. Comparison of CoderBot, Educational Chatbots, and generative AI Tools in programming education.

Feature / Tool	**CoderBot**	**PyO** [23]	**Coding Tutor** [21]	**TicTad** [24]	**Generative AI (GPT, Gemini, Copilot)**
Pedagogical Model	EBL with worked examples	Vygotskian perspective	ICAP Framework (Interactive, Constructive, Active, Passive) and Scaffolding	Expert system and Active	Generative (few-shot prompting)
Use of Cognitive Load Theory	Explicit, via structured templates	Acknowledged, but not central	Not addressed	Not addressed	Implicit only
Correct vs Erroneous Examples	Yes, both provided	Yes, provides correct answers and guidance for errors	Yes, provides feedback on errors and solution guidance	Yes, provides correct code examples and creation guidance	Ad hoc (no pedagogical structure)
Personalization	Planned via worked examples templates	Limited, future work	Adaptive scaffolding	Based on user history an learning styles	Adaptive through context and memory
Handling of Complex Questions	Limited (template-based)	Introductory focus	Medium (can respond to open questions and guide step-by-step)	Basic structured prompts	High — natural language
Feedback Type	Localized and explanatory	Textual explanations and examples	Conversational feedback (text and compiler errors)	Text, voice (text-to-speech), code examples, images	NL explanation (varied)
Open-ended Input	No (menu-based)	Yes	Yes	Yes	Yes
Platform	Web-based	Web-based	Web-based	Desktop application (Verbot-based)	API / Cloud-based tools, IDE extensions

focused on achieving the expected result, which benefits comprehension and understanding of the content taught in the classroom.

While students learn programming, they can leverage CoderBot as an active support to assist them in solving exercises and developing critical programming skills. In addition, CoderBot, as a pedagogical agent, provides both practical correct and incorrect code examples, demonstrating effective strategies for tackling specific programming problems. By examining the correct examples, CoderBot guides the students in developing computational thinking, helping them understand the underlying logic of tasks and the steps needed for problem-solving [28], [29].

This interactive, example-driven approach—central to EBL — greatly aids students in bridging theory with practice, facilitating a clearer understanding of programming concepts [29]. Employing correct examples offers numerous benefits: it reduces cognitive load, strengthens content retention, and enhances students' confidence. Additionally, these examples enable students to focus intensively on each concept, fostering self-sufficiency in problem-solving.

Erroneous examples, conversely, play an important role in inviting students to think more deeply about content. When provided with erroneous code, students must diagnose the problems, try to understand them, explain them, and fix them appropriately [30], which cultivates critical thinking and analytical skills. As a result, students develop a set of thinking and scoring skills and have the potential to learn better and enjoy a more stimulating—and thus more effective—learning environment. According to studies conducted by [31], presenting erroneous examples is effective because it forces students to focus on why such a code doesn't work. It further engages students in developing a deeper understanding of programming basics.

In this context, we integrated CoderBot into a Web portal, enabling the presentation of correct code examples with detailed steps and erroneous ones, challenging students to identify problems in the code, and providing immediate feedback. Below, we will present more details about CoderBot and its use. An essential part of this integration was the incorporation of a standardized Worked Example (WE) template into CoderBot. We develop the template based on the needs observed among instructors for structured and consistent teaching materials [32]. The close collaboration between the template's designers and CoderBot's developers was key to ensuring an effective integration. Through this process, we adapted the template to suit the interactive and dynamic nature of CoderBot, providing clear instructional elements such as problem descriptions, expected outcomes, reflective prompts, testing strategies, and guided feedback. This incorporation allows CoderBot to offer a cohesive and pedagogically sound learning experience. The structured template ensures consistency across different topics. At the same time, we ergonomically designed the selection of elements to display, aiming to avoid visual clutter for students, and were informed by recommendations from instructors involved in the Worked Examples Template for Programming Education study [32]. By combining the strengths of the WE template with CoderBot's interactive features, the platform not only supports more effective lesson planning for instructors but also fosters deeper student engagement with the material.

CoderBot's functionalities are structured to address the needs of both students and instructors. CoderBot offers students a selection of topics and content areas, presenting

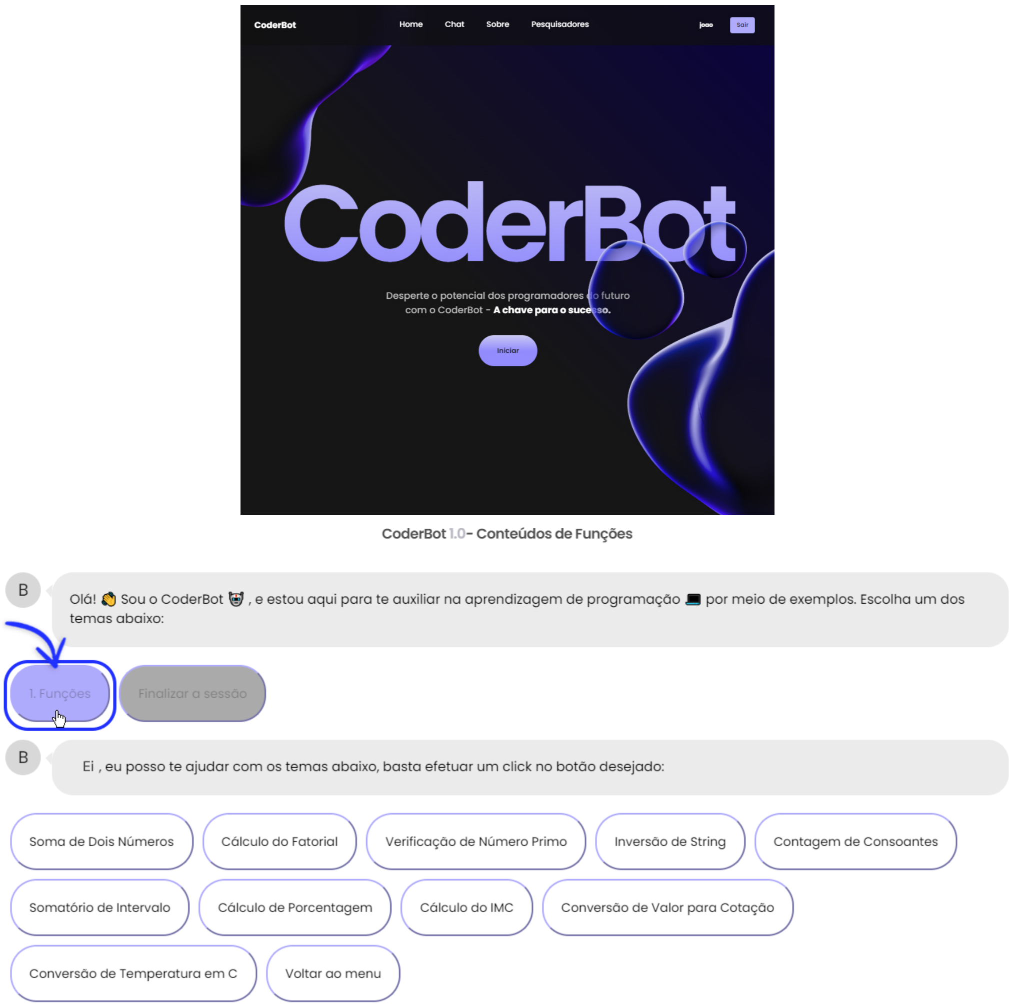


FIGURE 1. **(a) Home screen e (b) CoderBot chat interface.**

correct and incorrect code examples aligned with the chosen content. This approach allows students to study self-directed, reinforcing their understanding of programming concepts through immediate, practical examples. For instructors, CoderBot includes a customizable interface that enables them to adapt examples according to their classes' unique requirements, enhancing the tool's pedagogical relevance. This study focuses specifically on the student experience, emphasizing the design of CoderBot's interface to provide an intuitive, accessible, and engaging learning environment.

On the Home screen (Figure 1), students are presented with a brief introduction to CoderBot and an overview of its available content. This introductory layout provides a clear entry point for students to familiarize themselves with CoderBot's functionalities and navigate its resources effectively. Upon accessing CoderBot's chat screen (Figure 1), students are greeted with a welcome message and provided with two options: "Functions" and "End Session." When the "Functions" button is selected, a set of code examples is presented (for example, "Sum of two numbers" or "Factorial calculation"), enabling students to choose the topic they wish to explore.

Once a topic is selected, students encounter an exercise statement with options to view a correct or erroneous example. When choosing a correct example, they are shown accurate code with a step-by-step guide on how to reach the solution (Figure 2). By viewing the correct examples, we expected students to develop computational thinking, understand the logic behind the questions, and follow structured steps to solve the problems [28], [29].

If they choose an erroneous example, Students are prompted to identify the line containing the mistake from multiple options. Selecting the correct line triggers a congratulatory message, a clear explanation of the error, and the correct solution (Figure 3).

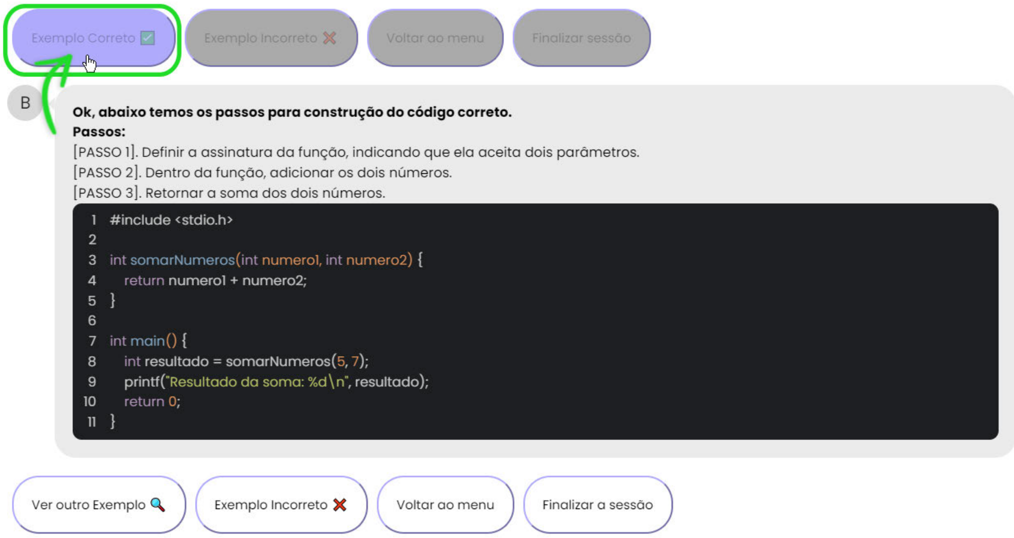


FIGURE 2. Correct example provided by CoderBot.

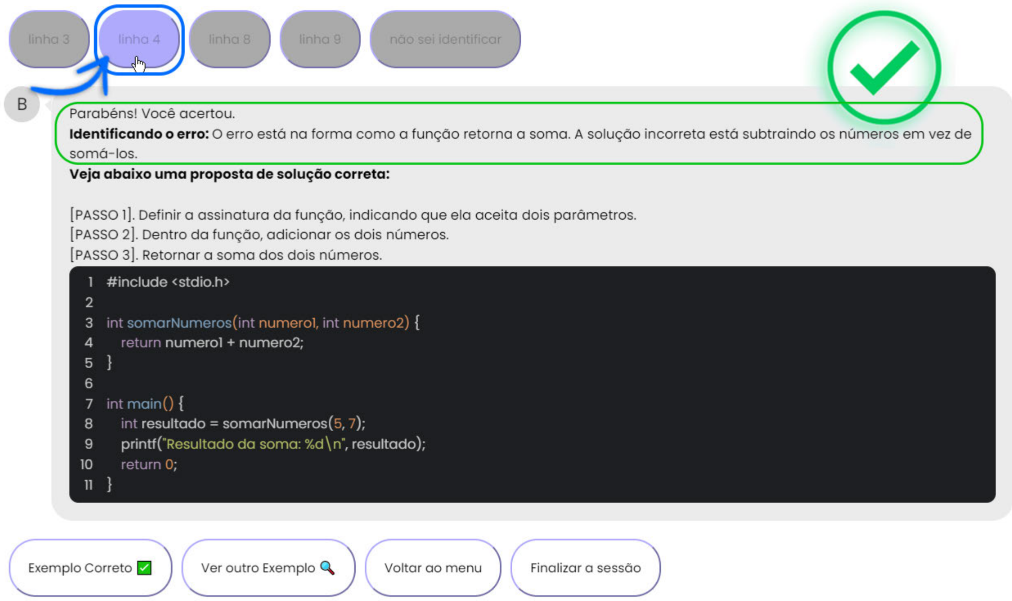


FIGURE 3. Selecting the ***Correct*** option in an erroneous example in CoderBot.

If a student chooses an incorrect line, a message highlights the problem, explains the error, and provides the corrected code along with a step-by-step solution (Figure 4). This process, grounded in reflective practice, encourages students to analyze errors critically, understand why they occur, and effectively correct them [30]. This enhances student learning and provides a more stimulating and effective environment for teaching. Thus, CoderBot promotes a reflective and participatory approach to the learning process. Additionally, CoderBot offers flexibility for both instructors and students. Instructors can customize materials based on course requirements, while students can engage with content at their preferred pace. This structure fosters a participatory and reflective approach, reinforcing the learning process through structured guidance and active engagement.

IV. EXPLORATORY STUDY

We performed an empirical study to evaluate CoderBot and to gain insight into students' perceptions of its role as a pedagogical tool in teaching programming.

A. SUBJECTS

This study involved 103 students enrolled in introductory programming courses at several higher education institutions (HEI) in our country, more specifically at: Federal University

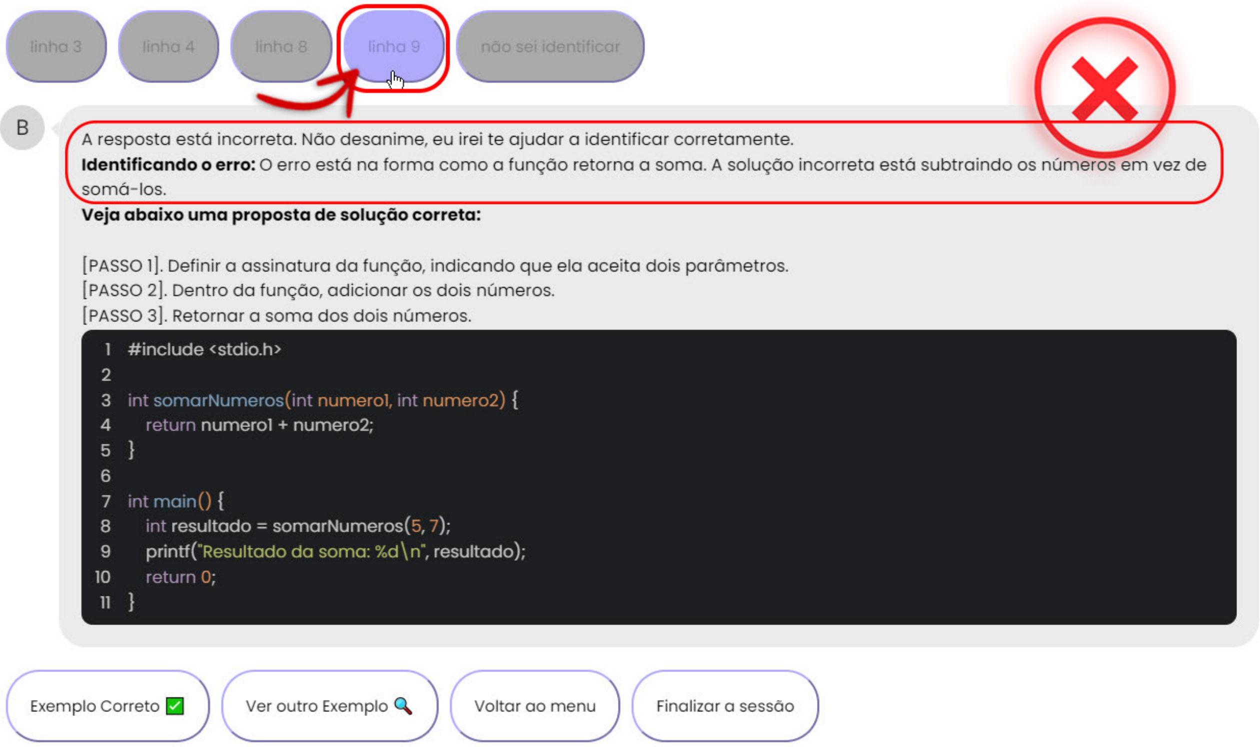


FIGURE 4. Selecting the *Incorrect* option in an erroneous example in CoderBot.

TABLE 2. Overview of study participants.

Inst.	Major	Instr.	Course Name	Concepts	#Students
UNIPAMPA	Computer Science	D1	Algorithms and Programming for Computing	Functions	8
UNIPAMPA	Software Engineering	D2	Algorithms and Programming	Arrays	11
IFMS 2	Technology in Systems Analysis and Development	D3	Computer Programming	Lists	15
IFMS	Technology in Systems Analysis and Development	D4	Programming Language I	Arrays	18
UFAM	Software Engineering	D5	Algorithms and Data Structures 1	Functions	51

of Pampa (UNIPAMPA), Federal University of Amazonas (UFAM), Federal Institute of Pará (IFPA), and Federal Institute of Mato Grosso do Sul (IFMS). Table 2 presents more specific demographic details. To gather representative data, we focused on students from foundational programming courses. We contacted the instructors responsible for these courses to coordinate the study's implementation. In total, we engaged five instructors, reaching out to them to understand the content they were covering, the current stage of their course, and their availability for scheduling the experiment.

B. PLANNING

To facilitate the execution of the study, we employed Google Workspace tools. The instruments developed for the experiment included: (i) a consent form, a key instrument in the experiment, designed to ensure confidentiality and participant anonymity; (ii) a characterization questionnaire aimed at gathering detailed information about the student's knowledge and background in programming; (iii) study documentation, including the experimental script, the CoderBot link, a list of exercises, and detailed instructions for the experiment; and (iv) a post-use evaluation questionnaire featuring open-ended questions to capture the students' perceptions of CoderBot. To ensure validity, two independent researchers peer-reviewed all study artifacts.

C. EXECUTION

We initially conducted a pilot study with two students to confirm that the experimental design met its objectives. The pilot's results were satisfactory, and the team found that no significant changes were needed for the artifacts.

We emailed the course instructors, outlining the study's objectives and providing comprehensive guidelines. Once the instructors agreed to participate, one of the researchers coordinated directly with them to schedule the experiment. The email correspondence emphasized the importance of instructor involvement, ensuring they actively assisted students during the experiment.

On the scheduled day, we conducted the study with the participating students. We integrated the experiment into the practical assessment activities in the course syllabus. During the session, the instructors acted as moderators, guiding the students through the tasks. Initially, students were asked to sign a consent form, agree to participate in the study, and allow their data to be analyzed. All participants willingly consented and signed the forms.

Next, students completed a characterization form, which included questions about their previous programming experience. The majority indicated that they had limited or no prior experience as they were in the early stages of their coursework. The moderators then provided training on CoderBot, covering its features and usage. We instructed the students to use CoderBot as a support tool for their programming tasks. Afterward, we distributed the programming exercises, and students used CoderBot to assist in solving the problems. The average time to complete the exercises was 136 minutes, with a minimum time of 40 minutes and a maximum time of 150 minutes, demonstrating the feasibility of using CoderBot in a classroom setting.

Upon completing the activities, students responded to a survey to measure their acceptance of CoderBot and perceived self-efficacy. We based our acceptance questionnaire on the Technology Acceptance Model (TAM) [16] (Table 3), which evaluates three indicators: perceived usefulness (the extent to which the student believes that CoderBot improves academic performance); perceived ease of use (the degree to which students feel CoderBot can be used without difficulty); and perceived intention to use (the likelihood that the student will continue to use CoderBot in the future). Responses were recorded on a five-point Likert scale, ranging from "Strongly Disagree" to "Strongly Agree", with a neutral option available.

TABLE 3. TAM Questionnaire items.

TAM Items	**Perceived Usefulness**
	PU1 – Using CoderBot improves my performance when solving programming problems.
	PU2 – Using CoderBot improves my productivity when solving exercises.
	PU3 – CoderBot makes my job easier while learning to program.
	PU4 – I find CoderBot useful in helping me learn programming.
	Perceived Ease of Use
	PEU1 – CoderBot was clear and easy to understand for me.
	PEU2 – Using CoderBot didn't require much mental effort.
	PEU3 – I think CoderBot is an easy to use chatbot.
	PEU4 – I find it easy to remember how to perform tasks using CoderBot.
	Perceived Intention to Use
	PIU1 – Assuming I have access to CoderBot, I plan to use it in the future to support my learning to code.
	PIU2 – Given that I have access to CoderBot, I foresee that I would use it in the future to learn programming.
	PIU3 – I plan to use CoderBot to help me solve exercises next month.

The self-efficacy questionnaire was also administered based on the framework proposed by [17]. This questionnaire assessed the following indicators (Table 4): satisfaction (perceived pleasure in using and usefulness of CoderBot); usability (perceived ease of using CoderBot); benevolence (user's perception of CoderBot's accurate and intentional actions, considering the user's interests); and credibility (perception of CoderBot's ability and experience). The students recorded their responses on a five-point Likert scale, ranging from "Very Dissatisfied" to "Very Satisfied."

TABLE 4. Self-efficacy questionnaire items.

Self-Efficacy Items	**Satisfaction**
	S1 – How much did you enjoy using the CoderBot?
	S2 – How useful was the CoderBot to you in assessing the risk of recidivism?
	S3 – How would you rate your overall satisfaction with the CoderBot?
	Usability
	U1 – How easy was the CoderBot for you to use?
	U2 – How understandable did the CoderBot provide the answers?
	U3 – How acceptable is the time spent asking the CoderBot questions?
	Benevolence
	B1 – Did you feel the CoderBot correctly understood your questions?
	B2 – Did you feel that the answers provided by the CoderBot were clear?
	B3 – Was the interaction with the CoderBot pleasant?
	Credibility
	C1 – Should the CoderBot be integrated into training practices?
	C2 – Should the CoderBot be mandatory for use in training?
	C3 – Did you feel the CoderBot was credible?

D. ANALYSIS OF RESULTS

The data collected were analyzed using both quantitative and qualitative methods. We tabulated the questionnaire responses for the quantitative analysis and utilized descriptive statistics to calculate maximum, minimum, mean, and standard deviation values. We also used R Studio software to generate stacked bar charts, which facilitated the visualization and interpretation of the results.

We also performed a specific analysis of student comments collected from the questionnaires. We followed a structured coding procedure to interpret these qualitative data. The qualitative analysis aimed to code, categorize, and synthesize data to identify the difficulties and benefits students perceived after using CoderBot. We adopted a four-step qualitative analysis procedure designed by [33].

We read all the student comments in the **first step**. This was an important step in obtaining a broad view of the data provided by students during the CoderBot experimentation period. We also performed a filtering process to remove comments with no answers or out-of-context answers that addressed aspects unrelated to CoderBot. In the **second step**, we performed open coding, in which we created codes (concepts relevant to understanding the perception of CoderBot) from the participants' responses (quotes). In the **third step**, we performed axial coding, grouping the codes according to their properties and forming concepts representing subcategories and categories. Finally, in the

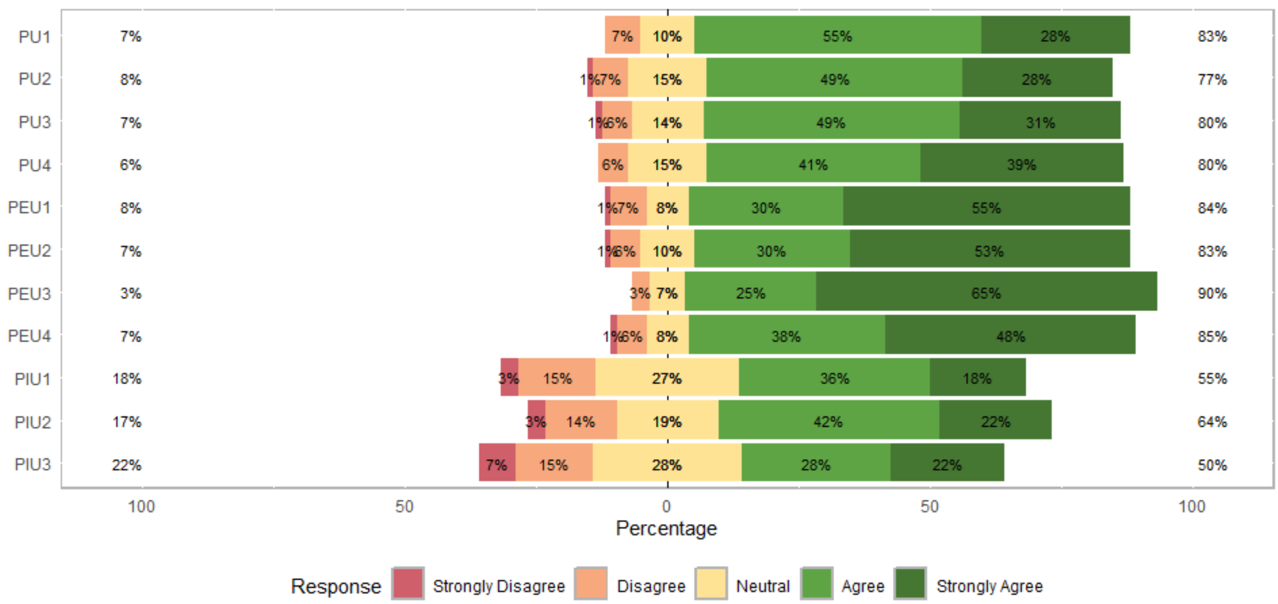


FIGURE 5. **Acceptance perceived by students.**

fourth step, we performed a complete evaluation of the final analysis to verify the consistency of the results. One researcher conducted the qualitative study and subsequently discussed it with other researchers who were experienced in qualitative methods. This collaborative approach helped to mitigate bias and improve the reliability of the findings. The insights gained from this analysis contributed to refining CoderBot for future iterations.

V. RESULTS

A. QUANTITATIVE ANALYSIS

The results of students' perceptions of CoderBot, evaluated through the TAM, are presented in Figure 5.

Findings related to **Perceived Usefulness** (PU1, PU2, PU3, and PU4) indicated that 83% of the students agreed that CoderBot improves their performance in solving programming problems (PU1). Student E34 noted: "*CoderBot helped because it showed examples of how to answer quick questions that could take hours to solve.*" Additionally, 80% of the students agreed that CoderBot could be an educational tool in learning programming (PU4). Student E83 remarked: "*It helps in learning programming because it brings practical examples seen during classes, making the student understand the programming logic proposed by the exercises.*" This evidence indicates that CoderBot was well received by the students as a valuable educational tool, reinforcing its perceived effectiveness and utility as a support for programming instruction.

The analysis of **Perceived Ease of Use** (PEU1, PEU2, PEU3, and PEU4) revealed an agreement of over 85%, underscoring that most students found CoderBot accessible and intuitive to use. For example, in item PEU3, E17 stated: *"using CoderBot was easy, helping to solve simple exercises and, at times, exercises of medium difficulty."* Therefore, including CoderBot in teaching does not present significant difficulties for students. This positive perception is reflected in other items, especially PEU4, which obtained 85% agreement among students. Students E40 and E52 also reported that CoderBot is intuitive. Therefore, incorporating CoderBot into programming education does not pose significant usability challenges for students.

Regarding **Perceived Intention to Use** (PIU1, PIU2 and PIU3), while over 56% of students responded positively, a considerable proportion remained neutral (24.7% on average) or disagreed (19% on average).

Student P66 stated: "*the range of data for demonstration examples is small, but it showed the initial path to solving the questions.*" Similarly, E33 suggested: "*for learning purposes, it would be important to add comments to the codes to help understand them.*" These insights reveal that, although CoderBot has gained general acceptance, there are opportunities for improvement in its functionality and breadth. The observed variability in responses suggests that the current limitations may be due to CoderBot not yet being in its final version, highlighting potential areas for further refinement.

Figure 6 presents the results of students' Perceived Self-Efficacy when using CoderBot.

For the **Satisfaction** indicator (S1, S2, and S3), we observed that a majority of students (average of 68%) expressed satisfaction with CoderBot. For example, E81 reinforced that the CoderBot "*helped to give direction to each question, which saves time, that's great.*" In addition, 68% of students agreed that they liked CoderBot, with E18 commenting: "*even though the exercise doesn't require exactly the function that CoderBot has, it's easy to use as a basis to adapt to the real problem of the question.*"

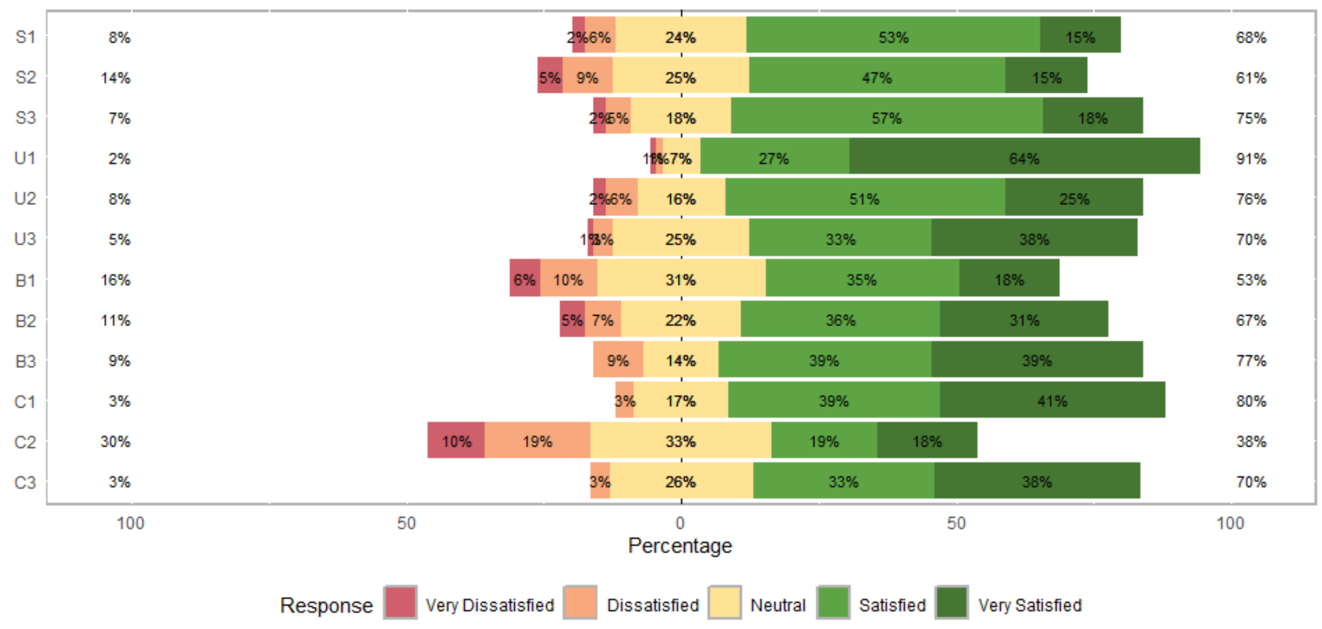


FIGURE 6. **Self-efficacy perceived by students.**

Regarding **Usability** (U1, U2, and U3), this indicator received an average agreement of 79%, the highest among the self-efficacy indicators. Notably, item U1 obtained an agreement from 91% of students, demonstrating the ease with which students could navigate and utilize CoderBot effectively. These findings collectively suggest that CoderBot provides a good user experience, contributing to positive perceptions of satisfaction and usability.

In the **Benevolence** indicator, positive agreement averaged 65%. Item B3, which assessed whether interaction with CoderBot was pleasant, achieved the highest results, with 77% agreement. For example, E23 stated: "*using CoderBot, I was able to improve my understanding of programming codes. Its interface is intuitive and simplifies interaction, making it easy to use.*" However, item B1, which evaluated whether CoderBot accurately understood the questions, showed a significant level of neutrality (31%). E01 highlighted a limitation, commenting that "*although CoderBot has been very helpful, its current state is considerably limited, especially with more complex questions*."

The **Credibility** indicator received the lowest overall agreement (average 62.5%). Item C1, which queried whether CoderBot should be integrated into programming instruction, was met with 80% agreement. E18 pointed out that "*CoderBot helped solve some exercises due to the simplicity of the explanation and the structured examples.*" However, item C2 showed significant neutrality (33%), with E48 commenting that: "*CoderBot is very understandable and practical; however, it doesn't explain what beginners need to know, which is why I hesitate to rely on it solely for classroom content.*" These insights indicate that enhancing explanations of code logic within CoderBot could improve its instructional effectiveness, especially for beginners.

B. QUALITATIVE ANALYSIS

In our qualitative analysis, we identified four main categories reflecting various perspectives on CoderBot's impact: (i) CoderBot's Contribution to Learning Improvement, (ii) Support from Examples in CoderBot for Conceptual Understanding, (iii) Limitations in CoderBot's Learning Support, and (iv) Suggested Enhancements for CoderBot.

The first category, **CoderBot's Contribution to Learning Improvement**, addresses how CoderBot facilitated the learning process and highlights specific support areas.

Within the subcategory CoderBot Enhanced Language Comprehension, several students commented on how CoderBot aided their understanding of the programming language used in the exercises. For instance, E01 shared a positive experience: *"CoderBot was instrumental in completing exercises and deepening my understanding of the C language."* Similarly, E31, who had experience in another programming language, remarked: *"I am familiar with another programming language, which is not C, and CoderBot helped me to get a quick start with C syntax and structure."*

Students also noted that CoderBot's impact on the problem- solving skills. Regarding this, E08 expressing: *"It simplified the exercises by structuring my thought process. The well-written code clarified the logic behind each question"*. Further, E83 highlighted that CoderBot enhanced their understanding of programming logic: *"CoderBot aids in learning programming by providing practical examples in Introduction to Programming and Data Structures courses. This helps students grasp the programming logic behind each exercise."*

The subcategory CoderBot's Role in Problem-Solving Support captured feedback on how CoderBot facilitated problem-solving and exercise development. E16 shared, *"CoderBot offers insights into tackling problems,"*) while E14 observed, *"CoderBot could help me solve problems, even though it was not the exact problem I was having, but the tool showed me step-by-step how to solve the bigger problem"*. Additionally, we noted that CoderBot helped beginners in programming, with students highlighting the support provided to those just starting to learn programming. As E67 noted *"CoderBot is an invaluable resource for those new to programming."* E60 agreed, stating: *"It helped with fundamental questions, and for beginners, it's quite beneficial"*.

Under the subcategory Didactic Clarity in CoderBot's Explanations, students highlighted the clarity and instructional effectiveness of CoderBot's explanations. E57 commented that CoderBot provides a *"clear and direct content that makes it easy to follow. The step-by-step breakdown of code execution helped me understand each line."* E73 echoed this sentiment: *"It was much clearer to visualize the code, and in terms of learning, it helped me a lot; with the step-by-step instructions, it is easier to understand each line."* This shows that CoderBot's clarity facilitated learning, particularly in step-by-step guidance.

Lastly, CoderBot as a Tool for Reinforcement and Recall is the subcategory highlighting how CoderBot supports students in recalling previously learned material. E82 mentioned: *"Using CoderBot helped me improve; I was able to remember how to use Array (I learned it last semester), and I confess that it clarified many things for me"*. E15 also reflected: *"It was beneficial for some questions, for example, when I needed to recall some algorithms that I had already done on the list, like the summation ones"*.

The second emerging category was **CoderBot's Use of Examples to Enhance Understanding**. This category includes feedback from students who reported that the examples provided by CoderBot facilitated their comprehension of programming exercises.

A prominent subcategory was CoderBot's Use of Correct and Erroneous Examples. Based on the results, we observed that this approach enriched students' understanding of problem-solving processes. As student E49 noted, CoderBot's presentation of *"a correct and erroneous way to do the exercise"* helped them learn *"which path not to take when developing the code."* This insight highlights the pedagogical value of exposing students to both correct and erroneous examples during the explanation of an exercise, allowing them to understand correct and erroneous approaches clearly. Another important subcategory that we identified was CoderBot presents Detailed Step-by-Step Solutions. Students highlighted that the detailed solution of the exercises by CoderBot helped them better understand how to solve the questions. Student E57 mentioned that *"the step-by-step code execution topics helped me understand how each command line works."* Similarly, E73 noted: *"With the step-by-step, it was easier to understand each line, it was much clearer to visualize the code, and it helped me a lot in learning."*

The comments contained in this third subcategory report that thanks to the simplified way the exercise solutions were presented in the explanation CoderBot presents a simple explanation of code examples so they could better understand and absorb the content in question. This is exemplified by the comment from E18, who said he could solve the exercise codes due to the clear and simplified explanation provided by CoderBot: *"CoderBot helped solve some exercises due to the explanations' simplicity and the examples' structure."*

Finally, there were reports that the examples helped to start the code, indicating that the examples helped students take the first step in solving the exercises. For example, E76 mentioned that CoderBot helped start to solve the question, providing an idea of how to start, which motivated the student to develop the question: *"CoderBot was useful in starting the code by giving a brief idea of how the code could be made."* Student E66 suggested that more examples of each exercise could be made available. However, it acknowledged that with the examples present, he could find an initial path to solving the questions: *"The range of data for examples in the demonstration is small, but it showed the initial path to solving the questions."*

Our analysis found that while many students positively perceived CoderBot, some encountered challenges in using it effectively. The emerging category, **Challenges in Learning with CoderBot**, highlights why certain students found the tool less helpful than anticipated.

One of the perceived difficulties was that CoderBot required excessive reading, which was burdensome for some learners. Student E02 expressed: *"my initial impression is that the CoderBot requires a lot of reading and little typing"*, highlighting that they could use other available tools to achieve the same function, suggesting CoderBot's text-heavy interface felt limiting. This perception implies that excessive reading requirements may hinder students who prefer a more hands-on approach.

Another challenge was that effective use of CoderBot seemed to require prior programming knowledge. Student E37 commented: *"If the user has little programming knowledge, it would not help much with solving the problem."* Similarly, E52 added: *"It is intuitive and straightforward for those who already know a little about programming; as I am still in the beginning stages, I need help understanding the intersection, list, pointers, and more specific features."* These remarks indicate that students at the initial stage of learning may struggle with the exercises proposed by CoderBot due to insufficient foundational knowledge.

Students also reported that CoderBot sometimes complicated the resolution of activities instead of assisting, leading to increased confusion. E05 stated: *"in some questions in the exercise, it ended up complicating the process of developing the question."* S75 corroborated: *"sometimes, it made the simplest logic more confusing."* These comments suggest that, for some learners, CoderBot introduced additional complexity rather than clarity. In this regard, E38 remarked: *"it did not help much because when I did not understand the logic,* *I needed an explanation,"* implying that CoderBot's content alone was insufficient to clarify the underlying programming logic. Adding to this, E88 commented: *"Coderbot provides ready-made codes with few comments. It should focus more on the logic behind the questions, as understanding programming logic is more important than the coding itself."* These observations indicate a need for more detailed and explanatory examples. Complementing these points, P71 mentioned: *"It helped in some cases. However, it was not so clear in code in others."* Thus, due to the lack of clarity in the examples presented, some students experienced difficulties when attempting to develop the exercises.

The final emerging category, **Suggested Improvements for CoderBot**, explores and captures students' recommendations for enhancing feedback from students who proposed enhancements to make CoderBot more effective.

One of the main points highlighted was the need for more practical examples to assist in learning and exercise development. Student E29 expressed that CoderBot should display more programming examples with possible solutions: *"I think it should have more examples and show more ways to solve the same question. I do not remember which, but in some of the questions, it used Boolean, and others did not require it. So, I think having at least three examples of each concept would be appropriate if it is more complex."* E77 added: *"the availability of examples could be higher for more diverse problems; Coderbot will fall behind other chatbots."* These comments suggest that a greater variety of examples could improve the tool's utility, usability, and versatility.

Another point raised was the limitation of CoderBot's features. E01 commented: *"As much as it has been beneficial, its current state is considerably limited, especially regarding more complex questions."* E41 added that the tool was helpful but limited: *"It helped with basic questions, but it is still a limited system."* E46 proposed adding a button to copy and paste the example code displayed in CoderBot to save students' time: *"A button to copy the code would save time and let you edit only what's needed."* This feedback indicates the need for additional features to increase flexibility in using CoderBot.

Students also emphasized the need for more detailed comments within the code examples. E88 observed that *"CoderBot provides ready-made code with minimal comments. It should focus more on explaining the logic behind the examples since understanding logic is more critical than the syntax itself."* Including detailed comments could enhance students' comprehension of the code examples, particularly for beginners.

Another aspect discussed was the addition of new features to improve the user experience and usability of CoderBot. P85 suggested improvements to the chat interface and a feature to change the response language: *"the chat interface could be better, and an option to switch response language would be useful."* E72 also emphasized the need for descriptions of commands adopted in the examples: *"Some commands lack explanations. It would be nice if there were a description of the commands. If it is for beginners, each command should at least have a description."* These recommendations indicate that improvements to CoderBot's interface and features could make it more accessible and valuable for users at all levels.

VI. DISCUSSIONS

We describe an exploratory study to identify students' perceptions of CoderBot for programming education, specifically facilitating novice learners' understanding of programming fundamentals. Grounded in Example-Based Learning principles, CoderBot adopted correct and erroneous code examples to guide students in developing accurate code and recognizing potential errors in the code, functioning as a structured learning facilitator. Through personalized and structured examples, CoderBot supports cognitive map construction, promotes reflective learning, and fosters an interactive, participatory approach to programming education.

Our qualitative student feedback analysis reveals strengths and challenges, offering insights that extend previous research on educational chatbots. The study revealed four categories of perceptions toward CoderBot. In the first category (CoderBot's Contribution to Learning Improvement), students highlighted how CoderBot supported their programming studies. Reports indicate that CoderBot facilitated understanding of the C programming language (adopted by one instructor during the experiment), improving comprehension of syntax, structure, and logical flow in programming tasks. The clarity in the presentation of the exercises helped students understand the logic of the questions. CoderBot provided foundational support for some students and enhanced their grasp of key concepts through clear, step-by-step explanations. Moreover, CoderBot proved beneficial in helping students revisit and reinforce previously learned content, thus consolidating their understanding of critical concepts.

The second category underscored the role of CoderBot's examples in enhancing the comprehension of programming exercises. Coderbot provides correct and incorrect examples and clear steps for problem-solving, helps promote critical thinking, and encourages learning from mistakes. Students noted that the detailed explanations and simplified language improved content assimilation, while the initial examples encouraged student engagement in the exercises. In addition, we realized that a clear and practical explanation of the examples, with step-by-step explanations, was significant for understanding the exercises. These results highlight the importance of correctly integrating a clear, complete, and varied pedagogical approach in teaching programming.

Contrasting with the first category, the third category (Challenges in Learning with CoderBot) indicated that, despite CoderBot's benefits, it also presents some challenges for students. In this sense, students' feedback identified excessive reading requirements, the need for prior knowledge, and insufficient clarity, in some examples, as significant barriers. Also, some students reported that CoderBot sometimes complicated rather than facilitated their activity completion. These results highlight the need for improvements in CoderBot, specifically to make it more accessible and efficient for all levels of education and to improve its effectiveness.

Students also provided constructive feedback and suggestions for improving CoderBot, particularly highlighting its current limitation in addressing complex, open-ended programming questions. Compared to more advanced conversational agents like ChatGPT, students perceived CoderBot as less adaptable and more restricted in its responses — a limitation also attributed to the need for more diverse and context-rich examples. This constraint primarily stems from CoderBot's current architecture, which relies on a predefined, template-based dataset and rule-based interactions, limiting its ability to interpret nuanced queries or generate dynamic explanations. To address these limitations, students suggested enhancements such as expanding the variety of examples per topic, including explanatory comments within code, enabling copy-to-clipboard functionality, supporting language switching, and offering command-specific explanations. These insights point to the importance of making CoderBot more flexible, responsive, and pedagogically rich. In response to this feedback, our development roadmap includes the integration of Large Language Models (LLMs) in future iterations of CoderBot. This enhancement is expected to increase its capability to process and respond to complex, context-dependent queries through natural language generation. The integration will combine the pedagogical scaffolding of the current template-driven structure with the adaptability and semantic reasoning enabled by LLMs—striking a balance between instructional control and conversational flexibility. This evolution will make CoderBot a more intelligent and versatile educational companion, capable of delivering structured guidance while supporting richer and more personalized learning experiences.

The qualitative evidence suggests that CoderBot has strong potential as an educational pedagogical agent, particularly regarding clarity, accessibility, and ongoing support. From this, we can affirm that CoderBot emerges as an emerging educational technology, which demonstrates that it is possible to integrate pedagogical approaches, such as Example-Based Learning, during the design process of a chatbot as a conversational agent. This leads to further research on AI-driven education since the content taught will be communicated to students more clearly and accurately. Besides that, the results of this study help to further tailor educational research approaches in computer science education on a set of feedback mechanisms promoting personalized and adaptive instruction of computer science for learners by addressing the preferred learning modality of the students we studied in this paper. This is because we require notable educational chatbots 'designed to be plastic' in terms of richness and differentiation of feedback for heterogeneous classrooms, where students have different backgrounds and vastly different prior knowledge. In our view, this trend renders educational chatbots in their modern form an adjunct pedagogical tool and a central pedagogical resource in the instruction of STEM courses.

CoderBot's architecture and findings provide a foundation for exploring adaptive learning algorithms, dynamic content delivery, and real-time error correction in computer science education. This study highlights the value of integrating structured feedback mechanisms and customizable instructional content in chatbots, setting a precedent for designing tools that can support novice programmers in both formal and informal learning environments. Moving forward, this work will inform research on the role of chatbots in computer science education, particularly in creating accessible learning tools that can adapt to various student needs and cognitive preferences, fostering a more inclusive and practical programming education experience.

VII. THREATS TO VALIDITY

As in all empirical studies, some threats could affect the validity of the results. In this section, we discuss the threats to the validity of our findings [34], [35].

Internal Validity. The time available to the students could influence the results. However, we controlled this threat using exercises that could be constructed in the stipulated period. Each session lasted for two classes (duration of 1h40 per class). The exercises used to carry out the activities could have affected the study if the students did not understand the scenario. This threat was minimized using exercises based on actual problems. Also, the requirements of this scenario were explicit, such as simulating some exercises carried out in the classroom.

External Validity. As researchers, we know student behavior during activities could result in participant reactivity. However, in this case, the students understood that they were performing a graded practical assignment, which was already part of the course syllabus, as performed by Hay et al. [36]. So, this bias may not have influenced our results.

Conclusion Validity. The number of participants could be better statistically. However, sample size is a known problem in studies of Computer Science Education [33], [37]. We conducted our study in a specific teaching context. Therefore, it may not explain the whole reality. However, this exploratory study is essential as initial evidence regarding the CoderBot in programming education. If other researchers have the same teaching context, they can replicate this empirical study. According to Carver et al. [38], when studies are replicated and achieve the same or similar results as the original study, it gives greater validity to the findings.

Construct Validity. Regarding the application of a questionnaire used to collect student perceptions, we emphasize that this threat cannot be considered a risk to the validity of results since these questionnaires have already been validated and used in other studies [16], [17]. We also capture the students' perceptions using open-ended answers. Open-ended answers are usually more challenging when collecting subjective data. However, students are more comfortable providing real insights on a particular topic [35]. There is a risk of not recording all the relevant information. We avoided this by reviewing the answers with three more experienced researchers in programming education. Besides, we conducted a pilot to assess if the script would reach its goal.

VIII. FINAL CONSIDERATIONS

This paper introduces CoderBot, a pedagogical agent grounded in EBL that helps novice programmers learn through well-structured and personalized examples. To evaluate CoderBot's effectiveness, we conducted an exploratory study with undergraduate students from introductory programming courses. Findings reveal that CoderBot was well-received by students, with feedback indicating that it is an engaging and user-friendly tool for learning programming. Also, high levels of agreement in the Satisfaction and Usability indicators. The Perceived Self-Efficacy suggests that CoderBot successfully supports students in their educational journey. Nonetheless, variability in responses related to the Credibility indicator reveals areas for improvement, with students noting a need for greater diversity and depth in examples and additional explanatory comments in the code.

In addition to the improvements mentioned by students, further research endeavors could enhance CoderBot's role as a comprehensive programming education tool. Future work will also focus on integrating Artificial Intelligence driven by Large Language Models into CoderBot to enable a more sophisticated interpretation of student struggles and adaptable interaction patterns. We will plan additional experimental studies with instructors to evaluate their perception of CoderBot's impact on teaching practices, and with students using an updated version of CoderBot to assess their performance and comprehension across varied pedagogical settings.

ACKNOWLEDGMENT

The authors acknowledge the use of Grammarly and ChatGPT-4o to support the writing and revision process of this manuscript. ChatGPT-4o was employed to improve spelling, grammar, vocabulary, and overall style, as well as to assist in drafting and refining sections of the text. They also utilized ChatGPT 4.o to speed up the writing of Python code to create the graphics. They carefully examined, tested, and often corrected all suggestions, whereby we take full responsibility for the form and content of the article.

REFERENCES

[1] C. G. B. Morais, "Ensino e aprendizagem de programação: Estudo de caso no ensino superior," Tech. Rep., 2022.
[2] C. Papakostas, C. Troussas, A. Krouska, and C. Sgouropoulou, "A rule-based chatbot offering personalized guidance in computer programming education," in *Proc. Int. Conf. Intell. Tutoring Syst.*, 2024, pp. 253–264.
[3] G. Alves, A. Rebouças, and P. Scaico, "Coding dojo como prática de aprendizagem colaborativa para apoiar o ensino introdutório de programação: Um estudo de caso," in *Proc. Anais do Workshop sobre Educação em Computação (WEI)*, Jul. 2019, pp. 276–290.
[4] J. Penney, J. F. Pimentel, I. Steinmacher, and M. A. Gerosa, "Anticipating user needs: Insights from design fiction on conversational agents for computational thinking," in *Proc. Int. Workshop Chatbot Res. Design*, 2024, pp. 204–219.
[5] A. V. Robins, "Novice programmers and introductory programming," in *The Cambridge Handbook of Computing Education Research*, 2019, pp. 327–376.
[6] L. G. Dantas, "Um protótipo de um sistema para fornecer dicas para tarefas de programação em disciplinas de programação introdutória," Tech. Rep., 2020.
[7] F. Clarizia, F. Colace, M. Lombardi, F. Pascale, and D. Santaniello, "Chatbot: An education support system for student," in *Proc. Int. Symp. Cyberspace Saf. Secur.*, 2018, pp. 291–302.
[8] S. Ruan, L. Jiang, J. Xu, B. J.-K. Tham, Z. Qiu, Y. Zhu, E. L. Murnane, E. Brunskill, and J. A. Landay, "QuizBot: A dialogue-based adaptive learning system for factual knowledge," in *Proc. CHI Conf. Hum. Factors Comput. Syst.*, May 2019, pp. 1–13.
[9] P. Smutny and P. Schreiberova, "Chatbots for learning: A review of educational chatbots for the Facebook messenger," *Comput. Educ.*, vol. 151, Jul. 2020, Art. no. 103862. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0360131520300622
[10] P. K. Bii, "Chatbot technology: A possible means of unlocking student potential to learn how to learn," *Educ. Res.*, vol. 4, no. 2, pp. 218–221, 2013.
[11] M. D. L. Roca, M. M. Chan, A. Garcia-Cabot, E. Garcia-Lopez, and H. Amado-Salvatierra, "The impact of a chatbot working as an assistant in a course for supporting student learning and engagement," *Comput. Appl. Eng. Educ.*, vol. 32, no. 5, p. 22750, Sep. 2024.
[12] J. Correia, M. C. Nicholson, D. Coutinho, C. Barbosa, M. Castelluccio, M. Gerosa, A. Garcia, and I. Steinmacher, "Unveiling the potential of a conversational agent in developer support: Insights from mozillas pdf.js project," Tech. Rep., 2024.
[13] A. P. Chaves, "Desenho de linguagem de chatbots: Influência da variação da linguagem na experiência do usuário com chatbot assistente de turismo," in *Proc. Anais Estendidos do XVIII Simpósio Brasileiro de Sistemas Colaborativos (SBSC Estendido)*, May 2023, pp. 42–47.
[14] J. Edwards, J. Ditton, D. Trninic, H. Swanson, S. Sullivan, and C. Mano, "Syntax exercises in CS1," in *Proc. ACM Conf. Int. Comput. Educ. Res.*, Aug. 2020, pp. 216–226.
[15] R. K. Atkinson, S. J. Derry, A. Renkl, and D. Wortham, "Learning from examples: Instructional principles from the worked examples research," *Rev. Educ. Res.*, vol. 70, no. 2, pp. 181–214, 2000.
[16] V. Venkatesh and F. D. Davis, "A theoretical extension of the technology acceptance model: Four longitudinal field studies," *Manage. Sci.*, vol. 46, no. 2, pp. 186–204, Feb. 2000.
[17] A.-P. Raiche, L. Dauphinais, M. Duval, G. De Luca, D. Rivest-Hénault, T. Vaughan, C. Proulx, and J.-P. Guay, "Factors influencing acceptance and trust of chatbots in juvenile offenders' risk assessment training," *Frontiers Psychol.*, vol. 14, Jun. 2023, Art. no. 1184016.
[18] G. Ilieva, T. Yankova, S. Klisarova-Belcheva, A. Dimitrov, M. Bratkov, and D. Angelov, "Effects of generative chatbots in higher education," *Information*, vol. 14, no. 9, p. 492, Sep. 2023.
[19] A. C. S. da Sousa and R. L. Fecchio, "Chatbots no apoio à educação superior: Revisão de literatura," Tech. Rep., 2021.
[20] J. F. Pane, E. D. Steiner, M. D. Baird, L. S. Hamilton, and J. D. Pane, *How Does Personalized Learning Affect Student Achievement?* Santa Monica, CA, USA: RAND Corporation, 2017.
[21] S. Hobert. (2019). *Say Hello to 'Coding Tutor'! Design and Evaluation of a Chatbot-Based Learning System Supporting Students to Learn to Program.* [Online]. Available: https://api.semanticscholar.org/CorpusID:209149824
[22] S. I. Malik, M. W. Ashfque, R. M. Tawafak, G. Al-Farsi, N. A. Usmani, and B. H. Khudayer, "A chatbot to facilitate student learning in a programming 1 course: A gendered analysis," *Int. J. Virtual Pers. Learn. Environ.*, vol. 12, no. 1, pp. 1–20, Sep. 2022.
[23] G. Carreira, L. Silva, A. J. Mendes, and H. G. Oliveira, "Pyo, a chatbot assistant for introductory programming students," in *Proc. Int. Symp. Comput. Educ. (SIIE)*, Nov. 2022, pp. 1–6.
[24] V. Kasinathan, A. Mustapha, S. Siow, and M. Hopman, "TicTad: A chatterbot for learning visual C# programming based on expert system," *Indonesian J. Electr. Eng. Comput. Sci.*, vol. 11, no. 2, p. 740, 2018.
[25] X. Huang, "Example-based learning: Effects of different types of examples on student performance, cognitive load and self-efficacy in a statistical learning task," *Interact. Learn. Environ.*, vol. 25, no. 3, pp. 283–294, Apr. 2017.

[26] T. Van Gog and N. Rummel, "Example-based learning," in *International Handbook of the Learning Sciences*. Evanston, IL, USA: Routledge, 2018. [Online]. Available: https://www.routledgehandbooks.com/doi/10.4324/9781315617572-20

[27] J. Sweller, J. J. G. van Merrienboer, and F. G. W. C. Paas, "Cognitive architecture and instructional design," *Educ. Psychol. Rev.*, vol. 10, no. 3, pp. 251–296, Sep. 1998.

[28] B. Jury, A. Lorusso, J. Leinonen, P. Denny, and A. Luxton-Reilly, "Evaluating LLM-generated worked examples in an introductory programming course," in *Proc. 26th Australas. Comput. Educ. Conf.*, Jan. 2024, pp. 77–86.

[29] O. Chen, E. Retnowati, B. K. Y. Chan, and S. Kalyuga, "The effect of worked examples on learning solution steps and knowledge transfer," *Educ. Psychol.*, vol. 43, no. 8, pp. 914–928, Sep. 2023.

[30] C. S. Große and A. Renkl, "Finding and fixing errors in worked examples: Can this foster learning outcomes?" *Learn. Instruct.*, vol. 17, no. 6, pp. 612–634, Dec. 2007.

[31] D. M. Adams, B. M. McLaren, K. Durkin, R. E. Mayer, B. Rittle-Johnson, S. Isotani, and M. van Velsen, "Using erroneous examples to improve mathematics learning with a web-based tutoring system," *Comput. Hum. Behav.*, vol. 36, pp. 401–411, Jul. 2014.

[32] A. L. M. Miranda, R. Garcia, G. M. Lunardi, R. Vilela, P. H. D. Vale, and W. Silva, "Projeto e Avaliação de um template de worked examples para o ensino de Programação," in *Proc. Anais do XXXIV Simpósio Brasileiro de Informática na Educação (SBIE)*, Nov. 2023, pp. 1673–1684. [Online]. Available: https://sol.sbc.org.br/index.php/sbie/article/view/26788

[33] W. Silva, I. Steinmacher, and T. Conte, "Students' and instructors' perceptions of five different active learning strategies used to teach software modeling," *IEEE Access*, vol. 7, pp. 184063–184077, 2019. [Online]. Available: https://ieeexplore.ieee.org/document/8765700/2

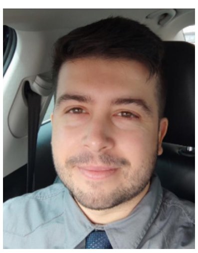
[34] K. Petersen and C. Gencel, "Worldviews, research methods, and their relationship to validity in empirical software engineering research," in *Proc. Joint Conf. 23rd Int. Workshop Softw. Meas. 8th Int. Conf. Softw. Process Product Meas.*, Oct. 2013, pp. 81–89.

[35] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslén, *Experimentation in Software Engineering*. Cham, Switzerland: Springer, 2012.

[36] L. R. Hay, R. O. Nelson, and W. M. Hay, "Methodological problems in the use of participant observers," *J. Appl. Behav. Anal.*, vol. 13, no. 3, pp. 501–504, Sep. 1980.

[37] W. A. F. Silva, I. F. Steinmacher, and T. U. Conte, "Is it better to learn from problems or erroneous examples?" in *Proc. IEEE 30th Conf. Softw. Eng. Educ. Training (CSEE&T)*, Nov. 2017, pp. 222–231. [Online]. Available: http://ieeexplore.ieee.org/document/8166706/

[38] J. C. Carver, N. Juristo, M. T. Baldassarre, and S. Vegas, "Replications of software engineering experiments," *Empirical Softw. Eng.*, vol. 19, no. 2, pp. 267–276, Apr. 2014.

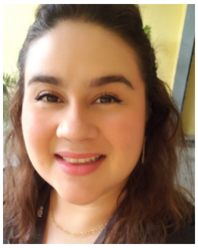


RENATO DE SOUZA GARCIA received the bachelor's degree in computer science from the State University of Mato Grosso do Sul (Dourados Campus) and the M.Sc. degree from the Graduate Program in Informatics, Federal University of Pampa (UNIPAMPA). He is currently pursuing the Ph.D. degree with the Institute of Mathematics and Statistics, University of São Paulo (IME-USP). He is a Professor at the Federal Institute of Education, Science, and Technology of Mato Grosso do Sul. He is a member of the NIPETI Research Group (Interdisciplinary Center for Research, Study, and Development in Information Technology). With previous experience as a Software Developer, as well as a strong academic background, he integrates research, teaching, and community outreach to drive advancements in information systems, software engineering, and software development. His research interests include computer science education, informatics in education, experimental software engineering, information systems, as well as conversational agents, chatbots, artificial intelligence, and machine learning.

JOÃO EMILIO ANTONIO VILLA received the bachelor's degree in software engineering from the Federal University of Pampa (UNIPAMPA), where he is currently pursuing the master's degree in software engineering. He is an Instructor at SENAC Alegrete, teaching Python programming and software development courses. He also has experience as a teaching and research fellowships in higher education. He has been actively involved in academic and outreach projects, including CoderBot, an educational chatbot developed to support programming education. This project resulted in empirical studies and scientific publications in the field of computing education. He has also contributed as a Volunteer in outreach programs, such as Programa C and Tramas at the Federal University of Pampa, aiming to promote educational initiatives and technology dissemination. With a solid technical background in front-end, UX design, Java, React.js, and Python, and an interest in interdisciplinary applications of technology, he integrates academic research, teaching, and software development to explore innovative approaches for enhancing programming education and user experience in intelligent systems. His research interests include software engineering, computing education, web development, chatbots, and usability and user experience (UX).

ANDRE LUIZ MENDES MIRANDA is currently pursuing the degree in software engineering with the Universidade Federal do Pampa (UNIPAMPA). He works at Venda ERP, a white label ERP company, where he began as a Full-Stack Intern, developing chatbots and solutions for the white label ecosystem. Later, he transitioned to 360Chat, a customer support platform, contributing to its development and enhancement. With a strong background in full-stack development, he has experience in automation scripts, API integrations, CI/CD pipelines, and gRPC-based microservices. His expertise includes React,.NET, Java, and Go, with a keen interest in optimizing software workflows and automation. Beyond his professional work, he is passionate about education and technology. He has participated in an extension program focused on teaching children programming logic, which sparked his interest in computing education. Since then, he has contributed to academic research and publications on the subject. He is involved in a startup project, working on an expanding application while exploring team organization strategies, Kanban methodologies, and team management. His research interests include software engineering, chatbots, and educational technology, aiming to integrate technical expertise with user-centered innovations to enhance software development practices and user experience.

GILLEANES THORWALD ARAUJO GUEDES received the bachelor's degree in informatics from the University Campaign Region (Universidade da Região da Campanha)—URCAMP, Bage Campus, and the M.Sc. and Ph.D. degrees in computer science from the Graduate Program in Informatics, Federal University of Rio Grande do Sul (UFRGS). Currently, he is an Adjunct Professor with the Federal University of Pampa (UNIPAMPA) and a member and the Coordinator of the Graduate Program in Software Engineering (PPGES-UNIPAMPA). His research interests include requirements engineering, software design, modeling with UML, software process, software verification and validation, domain-specific modeling languages (DSMLs), and agent-oriented software engineering (AOSE).

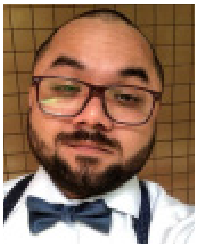


ANA CAROLINA ORAN received the bachelor's degree in systems analysis from the Centro de Ensino Superior Fucapi, the M.Sc. degree in computer science from the Federal University of Pernambuco (UFPE), and the Ph.D. degree in informatics from the Graduate Program in Informatics, Federal University of Amazonas (UFAM). Currently, she is a Professor with UFAM and a member of the Graduate Program in Informatics (PPGI-UFAM). She is part of the Usability and Software Engineering (USES) Research Group and actively contributes to research projects focused on enhancing software development practices and supporting the local software industry in the Amazon region. With previous experience as a Software Developer and a Requirements Analyst, as well as a strong academic background, she integrates research, teaching, and practical innovation to foster technological and human-centered advancements in software engineering. Her research interests include software engineering, requirements engineering, usability and user experience (UX), developer experience (DX), agile methodologies, and software testing.

PAULO SILAS SEVERO DE SOUZA received the master's and Ph.D. degrees in computer science from the Pontifical Catholic University of Rio Grande do Sul (PUCRS), Brazil. Currently, he is a Professor of software engineering with the Federal University of Pampa (UNIPAMPA). He has experience in software development, holding multiple software deposits at Brazilian National Institute of Industrial Property (INPI). Additionally, he has a strong research background, having published over 50 articles in prestigious venues, including IEEE COMMUNICATIONS LETTERS, *Future Generation Computer Systems*, IEEE ACCESS, *Measurement*, *Journal of Network and Computer Applications*, and *The Journal of Supercomputing*, where he also serves as a Regular Reviewer. His research interests include DevOps, cloud computing, and related paradigms.

RICARDO FERREIRA VILELA received the bachelor's degree in computer science from the Federal University of Gois (UFG), the M.B.A. degree in people management from ESALQ-USP, and the M.Sc. and Ph.D. degrees in computer science and computational mathematics from the University of São Paulo (ICMC-USP). Currently, he is an Assistant Professor with the School of Technology, University of Campinas (FT-UNICAMP). His research interests include software quality, experimental software engineering, software testing, search-based software testing, test automation, and chatbots. He is a member of Brazilian Computer Society (SBC).

PEDRO HENRIQUE DIAS VALLE received the bachelor's degree in computer science from the Federal University of Gois (UFG), the M.B.A. degree in project management from ESALQ-USP, and the M.Sc. and Ph.D. degrees in computer science and computational mathematics from the University of São Paulo (ICMC-USP). Currently, he is an Assistant Professor with the Institute of Mathematics and Statistics, University of São Paulo (IME-USP). His research interests include computer science education, informatics in education, software quality, experimental software engineering, software testing, software architecture, interoperability, educational games, conversational agents, chatbots, and artificial intelligence. He is a member of Brazilian Computer Society (SBC).



WILLIAMSON SILVA received the bachelor's degree in systems analysis and development from the Federal Institute of Education, Science, and Technology of Roraima (Boa Vista Campus) and the M.Sc. and Ph.D. degrees in informatics from the Graduate Program in Informatics, Federal University of Amazonas (UFAM). Currently, he is an Adjunct Professor with the Federal University of Cariri. With a solid academic background and an interdisciplinary approach, he actively contributes to the academic community, seeking advancements in the fields of computing education and software engineering. His research interests include computer science education, informatics in education, requirements engineering, software quality, experimental software engineering, information systems, usability and user experience, as well as conversational agents, chatbots, artificial intelligence, and machine learning. He was a member of the Steering Committee of the Special Commission on Information Systems (CESI) (2022–2023, 2023–2024, and 2024–2025) and the Steering Committee of the Special Commission on Informatics in Education (2024–2025), both committees of Brazilian Computer Society (SBC). He is also part of the Interest Group on Active Methodologies linked to the Special Commission on Computer Education.

Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) - ROR identifier: 00x0ma614