UNIVERSIDADE FEDERAL DO AMAZONAS
INSTITUTO DE COMPUTAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA

Maria Ivanilse Calderon Ribeiro

CollabProg: Collaborative Web Platform to Support Instructors in the Adoption of Active Methodologies in Programming Education

Manaus
August 2025

Maria Ivanilse Calderon Ribeiro

CollabProg: Collaborative Web Platform to Support Instructors in the Adoption of Active Methodologies in Programming Education

Defense of Doctoral Dissertation submitted to the Graduate Program in Informatics of the Institute of Computing at the Federal University of Amazonas as a requirement for obtaining the degree of Doctor in Informatics.

Advisor: Prof. Dr. Eduardo Feitosa..

Co-advisor: Prof. Dr. Williamsom Silva.

Manaus
2025

Ficha Catalográfica

Elaborada automaticamente de acordo com os dados fornecidos pelo(a) autor(a).

R484c Ribeiro, Maria Ivanilse Calderon
CollabProg: Collaborative Web Platform to Support Instructors in the Adoption of Active Methodologies in Programming Education / Maria Ivanilse Calderon Ribeiro. - 2025.
202 f. : il., color. ; 31 cm.

Orientador(a): Eduardo Luzeiro Feitosa.
Coorientador(a): Williamson Sila.
Tese (doutorado) - Universidade Federal do Amazonas, Programa de Pós-Graduação em Informática, Manaus (AM), 2025.

1. Teaching programming. 2. Active learning methodologies. 3. Computer programming. 4. Educational Tool. 5. CollabProg. I. Feitosa, Eduardo Luzeiro. II. Sila, Williamson. III. Universidade Federal do Amazonas. Programa de Pós-Graduação em Informática. IV. Título

---

Ministério da Educação
Universidade Federal do Amazonas
Coordenação do Programa de Pós-Graduação em Informática

FOLHA DE APROVAÇÃO

"COLLABPROG: UM REPOSITÓRIO COLABORATIVO ABERTO PARA APOIAR NA ADOÇÃO DE METODOLOGIAS ATIVAS NO ENSINO DE PROGRAMAÇÃO"

MARIA IVANILSE CALDERON RIBEIRO

Tese de Doutorado defendida e aprovada pela banca examinadora constituída pelos professores:

Prof. Dr. Eduardo Luzeiro Feitosa - **Presidente**

Profa. Dra. Ana Carolina Oran Rocha - **Membro Interno**

Profa. Dra. Elaine Harada - **Membro Interno**

Prof. Dr. Pedro Henrique Dias Valler - **Membro Externo**

gov.br Documento assinado digitalmente
PEDRO HENRIQUE DIAS VALLE
Data: 10/10/2025 16:30:15-0300
Verifique em https://validar.iti.gov.br

Profa. Dra. Taciana Pontual da Rocha Falcão - **Membro Externo**

Manaus, 12 de setembro de 2025.

sei! assinatura eletrônica

Documento assinado eletronicamente por **Elaine Harada**, **Professor do Magistério Superior**, em 09/10/2025, às 13:33, conforme horário oficial de Manaus, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **Eduardo Luzeiro Feitosa**, **Professor do Magistério Superior**, em 09/10/2025, às 13:52, conforme horário oficial de Manaus, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015 .

Documento assinado eletronicamente por **Ana Carolina Oran Rocha**, **Professor do Magistério Superior**, em 09/10/2025, às 15:15, conforme horário oficial de Manaus, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015 .

Documento assinado eletronicamente por **Taciana Pontual da Rocha Falcão**, **Usuário Externo**, em 10/10/2025, às 11:47, conforme horário oficial de Manaus, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

A autenticidade deste documento pode ser conferida no site https://sei.ufam.edu.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **2838498** e o código CRC **9F0215FA**.

Avenida General Rodrigo Octávio, 6200 - Bairro Coroado I Campus Universitário Senador Arthur Virgílio Filho, Setor Norte - Telefone: (92) 3305-1181 / Ramal 1193
CEP 69080-900, Manaus/AM, coordenadorppgi@icomp.ufam.edu.br

Referência: Processo nº 23105.037979/2025-65 SEI nº 2838498

Acknowledgments

First and foremost, I thank God, who has guided me to this point with health, strengthening my belief in human persistence and resilience when pursuing a purpose.

I am deeply grateful to my parents, who have always supported me and taught me to persevere and strive to be the best human being and woman in a society that has historically imposed barriers upon them. I also thank my family for their strength and for understanding my need for absence and isolation throughout this journey.

I thank Professor João Marcos Bastos Cavalcanti, who believed in my interest in pursuing a doctorate and initially supervised me in the Graduate Program in Informatics (PPGI). I also acknowledge all the professors who have been part of my journey, especially those from PPGI who inspire and motivate us to continually learn and contribute to the scientific community. I am also grateful to the technical staff of the Secretariat, who have always been helpful and efficient, facilitating the academic processes.

To the friends I have made along this path, who remain present in my life, I thank you for being part of my formation, not only academically but also personally.

A special thanks to Josiane Rodrigues, my first "best" on campus, who taught me how to reach ICOMP on a hot afternoon bus ride to UFAM. She taught me to focus on my studies, set goals, and carefully plan each step to achieve them. Moreover, she generously shared her family with me, providing many joyful and instructive moments.

To my friend and sister Ana Carolina Oran, who has always been by my side, caring for me and giving me a reality check when needed. Through her, I also gained her family, who welcomed me in Manaus and ensured I never felt alone.

I have also had the pleasure of gaining new families along the way, such as Alice Adativa's and Bruno Ábia's families, two examples of scholars and researchers who have always generously helped me understand my difficulties and provided wisdom to keep moving forward.

I am also grateful to Joyce Miranda, a brilliant and kind person, always present with thoughtful and supportive words when I needed them most. To Diego Rodrigues, whose irreverent and humorous nature, combined with his problem-solving skills, was of great help during an important phase of my journey.

Finally, my deepest gratitude goes to my advisor, Professor Eduardo Feitosa, whose generosity, sincerity, and dedication were fundamental in overcoming the most challenging phases of the doctorate, especially at UFAM, an institution that is demanding in many respects. He made it possible for me to reach this moment, the completion of this work.

I am equally grateful and pleased to have been co-advised by Professor Willianson Silva, who taught me much about conducting research, interpreting results, and the importance of sharing knowledge with the scientific community. To them, I dedicate all the work of the past four years and eight months, which resulted in twelve publications, including four articles in high-impact journals.

Looking back, I realise it was worth every stumble, every tear, every moment of anxiety and emotional imbalance. Perhaps it was a tense journey, I do not know, but what I will carry with me and be most proud of is the result we achieved together, supported by this wonderful team of people whom God placed in my life.

"*Eu sou mais forte do que eu*"

(Clarice Lispector)

Abstract

Background: Teaching programming is a challenging task, as it requires instructors to guide students in developing complex skills such as real-world abstraction, problem-solving, and logical reasoning. However, the traditional teaching approach is often ineffective in achieving these objectives. Evidence suggests that Active Learning Methodologies (ALMs) can provide a more conducive environment for skill and competency development. Nonetheless, instructors' adoption rate of ALMs remains relatively low due to various barriers and factors, particularly in programming education. **Goal**: The objective of this doctoral thesis is to support instructors in adopting active learning strategies in programming education. To achieve this goal, the research was guided by the Design Science Research (DSR) methodology, which enabled the definition of the research problem as well as the development, evaluation, and evolution of an artifact. **Method**: The DSR approach is an iterative process composed of three interconnected cycles: the Relevance Cycle, the Design Cycle, and the Rigor Cycle. During the Relevance Cycle, an analysis of the feasibility of the research topic was conducted. To this end, a Systematic Literature Mapping was carried out to understand the main challenges faced by instructors in adopting ALMs in programming education, as well as to identify the ALMs currently used by teachers to support this discipline. Furthermore, exploratory experimental studies were conducted to deepen the understanding of the ALMs identified in the literature from the instructors' perspective. The Design Cycle enabled the development, evaluation, and evolution of an artifact, which in this case is a repository called CollabProg (CollabProg: An Open Collaborative Repository to Support the Adoption of Active Methodologies in Programming Education). CollabProg provides specific guidelines to assist instructors in applying ALMs, as well as helping them identify the ALMs best suited to their teaching context. To evaluate and improve CollabProg, two design cycles were conducted in different educational institutions to assess its use and acceptance. The results showed that CollabProg effectively supported instructors in adopting ALMs in programming education, contributing to overcoming some of the barriers they face and reaching a maturity level suitable for adoption by other educators. Finally, the Rigor Cycle focused primarily on the generation and use of knowledge. The main foundations involve knowledge about programming education, the identified ALMs that support this process, the Systematic Literature Mapping, the experimental studies conducted, as well as the qualitative and quantitative analyses carried out during the research. **Findings**: Regarding knowledge generation, the main contribution to the knowledge base is CollabProg itself an innovative open repository that assists instructors in identifying the most appropriate ALMs for their specific teaching contexts in programming education. Additionally, the following contributions are noteworthy: (i) the process of using CollabProg in real-world settings, which serves as a reference for other instructors; (ii) the research conducted for the repository's development; (iii) the set of ALMs applicable to programming education; (iv) key considerations for implementing these strategies; and (v) the knowledge derived from analyzing the impact of these methodologies on the programming teaching process. Evidence demonstrates that CollabProg effectively supports instructors in adopting ALMs while identifying limitations and opportunities for improvement. It was also found that the repository helps instructors select the most suitable ALMs tailored to their teaching context and specific classroom needs. The guidelines provided by CollabProg proved to be useful and highly practical for lesson planning involving these methodologies. **Implications**: The adoption of CollabProg highlights the critical importance of implementing effective support strategies for instructors teaching programming, especially to enhance student engagement and motivation. Given the complexity of programming concepts,

providing educators with tools that facilitate the selection and application of active learning methodologies is essential. This need becomes even more pronounced in collaborative learning environments where social interaction and peer engagement play a pivotal role in the learning process. CollabProg's adaptability and comprehensive support for diverse teaching contexts, including collaborative settings, constitute a key factor in promoting successful instructional practices and improving learning outcomes.

Keywords: Teaching programming, Active learning methodologies, Computer programming, Educational Tool, CollabProg.

List of Figures

Contents

Chapter 1

Introduction

This chapter introduces the context of the study by presenting the background that motivated the research, the guiding research question, and the methodology adopted to address it.

1.1 Contextualization

Computer programming requires both cognitive and metacognitive skills. Students must not only understand the syntax and semantics of a specific programming language, but also apply their creativity to solve complex problems (Raj et al., 2018). This process, therefore, combines the rigour of logical thinking with the flexibility of creativity (Eteng et al., 2022). Teaching and learning programming is particularly challenging, especially in foundational courses, which are often perceived as complex and demand a solid understanding of abstract concepts (Luxton-Reilly et al., 2018; Raj et al., 2018).

Typically, such courses require instructors to support students in developing a range of skills, such as real-world abstraction, problem-solving, and logical reasoning (Eickholt, 2018). Compounding this issue is the continued reliance on traditional teaching methods, which are primarily centred on teacher-to-student instruction. This often results in a "lecture-style indoctrination" approach, leading students to lose interest in learning. Within this context, it becomes evident that both teaching methodologies and content must be continually updated and/or adapted (Garcia et al., 2021).

In this context, acquiring the skills required for computer program development is one of the main challenges faced by computer science students. When unable to develop the necessary competencies (e.g., abstraction), students often drop out of courses and, in some cases, leave their degree programmes entirely (Sobral, 2021b). However, this scenario has been evolving, driven by advances in the development of approaches that support the teaching and learning process (Astrachan et al., 2002). Active Learning Methodologies (ALMs) have become increasingly important, as they foster greater student engagement, encourage critical thinking, and facilitate the acquisition of practical skills essential for success in programming (Moya, 2017).

In particular, ALMs have been widely adopted in developing strategies to address this issue (Sobral, 2021a). They integrate active student participation, experiential learning, and learning by doing, making learners more accountable for their own progress and fostering motivation

and satisfaction (Imbulpitiya et al., 2020). Compared to traditional approaches, ALMs promote effective engagement in constructing knowledge (Bacich and Moran, 2018) and encourage autonomy, which supports the development of problem-solving skills (Witt et al., 2018).

However, challenges remain in adopting ALMs for teaching programming in computer science (de Almeida et al., 2019). These include the variety of available methodologies, the need to adapt strategies to specific course contexts, and the fact that many instructors feel overwhelmed or anxious when required to adopt new pedagogical approaches (Kong et al., 2020). Despite positive evidence of their effectiveness, adoption rates remain low (Nguyen et al., 2021). Reported barriers include limited time for lesson planning, difficulty covering the full syllabus within an ALM, student resistance to unfamiliar strategies, doubts about their effectiveness in achieving learning objectives, and lack of clear guidance for implementation (Eickholt, 2018; Tharayil et al., 2018). Resistance to change among instructors, particularly those accustomed to traditional teaching, also hinders adoption (Calderon et al., 2022). These challenges underscore the need for targeted professional development and adequate resources to support the transition to ALMs. Addressing these barriers proactively can promote broader adoption and enhance the quality of programming education.

Furthermore, in many higher education institutions, ALMs are implemented either as part of the curriculum or in isolated courses (de Farias et al., 2018), which complicates the alignment of teaching practices with the preparation of future professionals. Although there is literature on the use of ALMs in this area, many reported strategies and tools are context-specific and not readily transferable to other educational settings (da Silva and Oliveira, 2019; de Almeida et al., 2019; Gonçalves et al., 2017; Moreno, 2019).

1.2 Research Question

The problem addressed in this research concerns improving the teaching of programming in Computer Science. In this context, the study is guided by the following research question: **How can instructors be supported in addressing the challenges of implementing Active Learning Methodologies in programming education within Computer Science courses?**

1.3 Objectives

The main objective of this doctoral thesis is to support instructors in the adoption of ALMs in programming education at the higher education level, specifically within Computer Science courses. To achieve this general objective, the following specific objectives have been defined:

- Identify the methodologies used by instructors and the difficulties and/or challenges encountered when applying them in the classroom.
- Identify evidence in the literature regarding strategies for implementing ALMs in Computer Science programming courses.
- Develop and adapt a toolset that offers a set of strategies for adopting ALMs in programming education, tailored to different teaching contexts.

1.4 Research Methodology

This subsection presents the application of the Design Science Research (DSR) method used to develop CollabProg. It provides a detailed description of the DSR cycles.

To achieve the objectives of this research, a methodology based on Design Science Research (DSR) was employed. DSR involves the design and investigation of artefacts with the purpose of interacting with the context of a problem and improving a specific aspect within that context (Wieringa, 2009). In DSR, changes or improvements are made according to the needs of those involved, in this case the instructors in the field. The aim is to address a problem through the iteration of design and investigation activities within a design cycle (Wieringa, 2009, 2014). The DSR process consists of three interconnected research cycles (Hevner and Chatterjee, 2010): the **Relevance Cycle**, the **Design Cycle**, and the **Rigor Cycle**. Figure 1.1 presents the methodology adopted in this research.

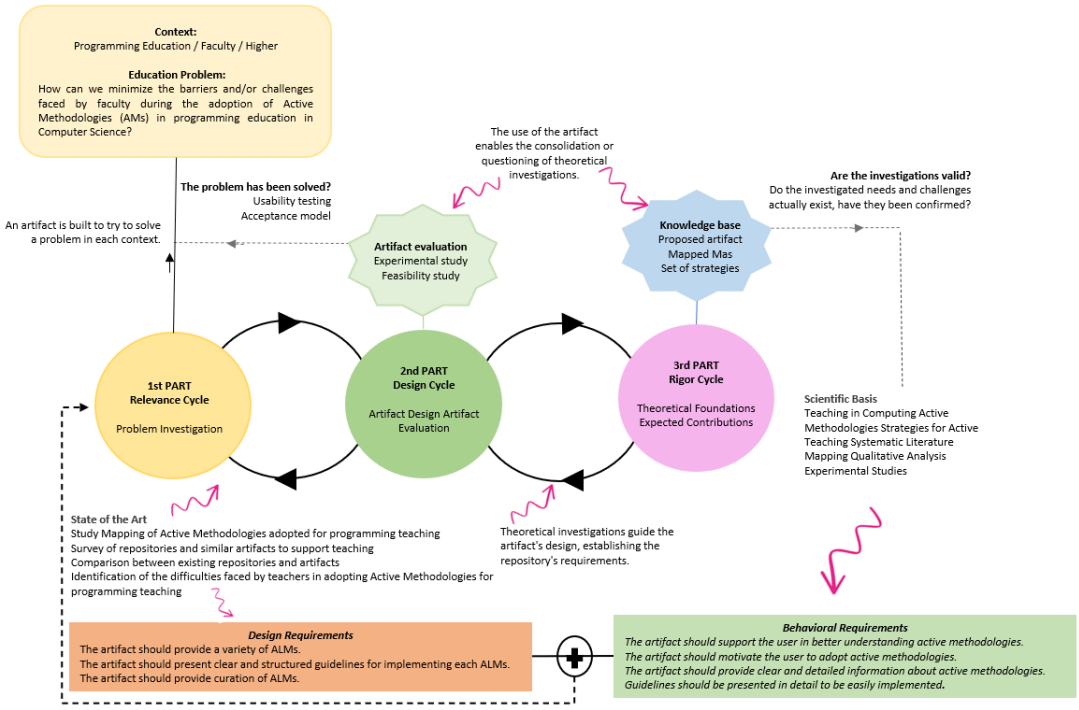


Figure 1.1: DSR adopted in the research

In the **Relevance Cycle**, we define the problem to be investigated, examine the context of the investigation, establish the motivation for addressing the problem, and set the acceptance criteria for the final evaluation of the research results. To support this process, we first conducted a Systematic Mapping Study (SMS) to summarise the types of ALMs and the experimental evidence related to the adoption of ALMs in programming education. Based on the results identified in the SMS, we defined two kinds of acceptance criteria for CollabProg: design criteria and behaviour criteria. These acceptance criteria were established to guide both the development and evaluation of the proposed solution, ensuring alignment with the evidence identified in the SMS. The acceptance criteria are detailed in the section 3.2.2.

In the **Design Cycle**, the solution proposed in this research, CollabProg, was developed, evaluated, and refined to ensure it addresses the problem and meets the defined requirements. The development was primarily based on the ALMs identified and selected from the SMS, to support instructors in adopting appropriate methodologies for teaching programming. CollabProg was evaluated through application to specific problems and contexts, allowing verification of whether the desired effects were achieved and whether further iterations of the Design Cycle were necessary. This practical evaluation involved instructors in the learning context. The results obtained from applying the artifact helped corroborate or challenge the validity of the defined requirements.

Finally, the **Rigor Cycle** refers to the generation and use of knowledge. It is grounded in research theories and methods, combined with the experience and understanding of the fundamentals guiding the research, and contributes to the expansion of the knowledge base (Hevner, 2007). Research methods were employed to document the steps carried out during the execution of the DSR cycles. Among these methods are the SMS and experimental studies, including qualitative analysis of the artefact based on instructors' perceptions. Research rigor is associated with credibility, reliability, precision and integrity, requiring theoretical and methodological rigour (Wieringa, 2014). Therefore, rigor is ensured when the researcher follows a previously established and validated method, preferably one widely recognised and accepted by the academic community. This requirement guided the use of solid theoretical foundations and existing technical knowledge in this research.

This research is based primarily on knowledge about ALMs, the barriers and difficulties faced by instructors in teaching, and the strategies they adopt. For the primary studies, we selected instructors teaching higher education courses in Computer Science, Information Systems, or Computer Engineering at educational institutions in Brazil. Questionnaires and the Technology Acceptance Model (TAM) were used as data collection instruments. TAM is a questionnaire designed to gather information about participants' perceptions regarding key factors that influence the acceptance or rejection of a given technology (Davis et al., 1989).

Therefore, we expected to present a technological support solution to assist instructors by consolidating in a single online portal strategies for adopting different ALMs in programming education. The platform would provide examples, activity suggestions, support options, tools used by the community, experiences with methodology adoption in various scenarios, results obtained by other instructors, and positive and negative aspects of the adopted ALM.

1.5 Organization of the Executive Summary

This document is presented in the form of an executive summary. The decision to structure the work in this way was motivated by the need to provide a clear and focused presentation of the main contributions, facilitating comprehension and dissemination of the results. An executive summary allows for a synthesis of the research process, highlighting key objectives, methodologies, and findings without extensive elaboration, which can be found in more detailed publications.

The executive summary is structured into chapters that correspond to different research stages and are related to scientific publications produced throughout the doctoral process. Table 1.1 presents the correspondence between each chapter and the respective publication(s).

Table 1.1: Mapping between chapters and scientific publications

Chapter	Objective	Reference
Chapter 2	Presents the theoretical background on programming education and the use of ALMs in the field of Computer Science. It also reviews related works that provide the foundation for this study	Eickholt (2018); Caceffo et al. (2018); de Castro and Siqueira (2019); Silva et al. (2020b); Lima et al. (2021); Ahshan (2021).
Chapter 3	Presents the results of the Relevance Cycle. It reports on a Systematic Mapping Study and a national survey conducted to investigate instructors' use of Active Learning Methodologies in programming education and the challenges encountered in their implementation.	Ribeiro and Passos (2020); Calderon et al. (2021); Calderon et al. (2024b); Calderon et al. (2024a); Calderon et al. (2025).
Chapter 4	Addresses the conception, evaluation, and refinement process of CollabProg. CollabProg was developed to mitigate instructors' practical difficulties in adopting ALMs in programming teaching by providing specific guidelines for their implementation. It is a collaborative and open repository designed to support instructors in the adoption of ALMs in programming education.	Ribeiro et al. (2021); Calderon et al. (2022); Calderon et al. (2023b); Calderon et al. (2022); Calderon et al. (2024c).
Chapter 5	Discusses the research rigor adopted in the study and highlights its main contributions, providing the reader with an overview of the methodological robustness and the value added by the findings	Ribeiro and Passos (2020); Calderon et al. (2021); Calderon et al. (2024b); Calderon et al. (2024a); Calderon et al. (2025); Ribeiro et al. (2021); Calderon et al. (2022); Calderon et al. (2023b); Calderon et al. (2022); Calderon et al. (2024c).
Chapter 6	Presents the main conclusions of the research, outlines directions for future work, discusses the implications of the findings, and addresses potential threats to the validity of the study.	Calderon et al. (2025); Calderon et al. (2024c)

Therefore, this thesis aims to contribute to programming education in Computer Science by supporting instructors in the adoption of active learning methodologies. By following a Design Science Research approach, the study proposes the development and evaluation of an artifact designed to address the challenges identified in the teaching context. The following chapters are organized to present the theoretical foundation, the methodological path, the design and validation of the proposed solution, and the main conclusions and contributions.

Chapter 2

Background and Related Works

This chapter presents the theoretical background on programming education and the use of ALMs in the field of Computer Science. It also reviews related works that provide the foundation for this study.

2.1 Teaching Programming in Computing Education

Programming is the core of computing technology. Teaching programming has become necessary due to the growing relevance of computing in daily life. Students must not only understand the syntax and semantics of programming languages but also apply creativity to solve complex problems, combining logical thinking with flexible problem-solving (Sharma et al., 2022). However, instructors face various challenges in this process. These challenges arise from the complexity of the subject matter, the diverse backgrounds of students, and the need to develop both technical skills and creative problem-solving abilities (Eickholt, 2018).

At the beginning of their courses, many students face difficulties designing and writing straightforward programs, and some hesitate to learn programming, perceiving it as a complex subject (Okonkwo and Ade-Ibijola, 2023). A lack of understanding of fundamental concepts is also a significant obstacle (Corritore and Love, 2020). Computer Science courses (CS) are often considered challenging because they require prior skills in logic, mathematics, and text interpretation (Freire et al., 2019). Students need prior knowledge of logic, mathematics, reading and text interpretation, abstraction, and other skills to succeed in these courses (Bigolin et al., 2020).

These challenges are reflected in high dropout rates in undergraduate computing programmes (Raj et al., 2018). Students majoring in CS, as well as those from other disciplines learning programming, face difficulties and often show signs of poor performance, frustration, and lack of engagement (Beaubouef and Mason, 2005). Some institutions report dropout rates of up to 50%, while the estimated average global pass rate for CS1 is around 68% (Penney et al., 2023). Considerable effort has been made to understand why learning programming remains consistently difficult (Penney et al., 2023). This is believed to be partly due to current instructional methods (Beaubouef and Mason, 2005), high expectations from instructors (Luxton-Reilly, 2016), and the perceived lack of support for beginner students (Luxton-Reilly, 2016). In summary, teaching programming courses in higher education is complex due to the broad range of skills required for student success (Denny et al., 2011).

2.2 Active Learning in Programming Education

Lecture classes, traditionally instructor-centred, involve students passively listening and absorbing the presented material, often supported by slide presentations. Although these lectures are necessary in some contexts, they often represent only a superficial change in teaching, merely replacing the blackboard with a projector (Caceffo et al., 2018). Technological advances have changed the means of delivering information but have not significantly altered how students learn. However, this method of instruction is limited because it does not foster higher-order thinking or advanced reasoning skills (O'grady, 2012).

On the other hand, Active Learning (AL), strongly influenced by Constructivism (Lima, 2016; Selçuk and Yilmaz, 2020), offers an approach where students actively construct their knowledge, assuming greater responsibility and control over their learning (Sasson et al., 2022). Constructivism is a learning theory that states that individuals actively construct their knowledge, determined by experiences (Travers et al., 1993). In this context, AL is actively constructed by the student, providing them with greater responsibility and control over their learning process. In this approach, the student not only passively absorbs knowledge but learns through practice and experience. This fundamental distinction motivates students to take responsibility for their learning (Yannier et al., 2021). In AL, instructor guide students to think, reflect, and cultivate their curiosity (Matsushita, 2018; Feyzi Behnagh and Yasrebi, 2020).

According to Parsons (2011), AL allows instructors to create learning situations in which students build knowledge, develop critical and reflexive skills, and explore personal attitudes and values. AL is a student-centred approach suitable for developing skills in independent study, self-determination, and collaborative work (Tutal and Yazar, 2022). Yannier et al. (2021) point to a growing consensus that humans learn more effectively when they are active rather than passive, engaged rather than distracted, when the content is meaningful rather than disjointed, and when learning takes place in socially interactive, iterative, and enjoyable environments.

The literature highlights the advantages of AL in the curricular structure of undergraduate courses (Pundak and Rozner, 2008; Eickholt, 2018; Feyzi Behnagh and Yasrebi, 2020), showing that active learning strategies tend to be more effective than traditional lectures in promoting various educational outcomes, such as improved learning performance (Silva et al., 2019). However, despite the favourable evidence, traditional teaching remains the predominant approach in university courses (Yannier et al., 2021).

2.3 Active Learning Methodologies in Programming Education

Programming involves coding step-by-step solutions and developing computational thinking (dos Santos et al., 2020). Thus, teaching programming requires selecting appropriate procedures, techniques, and tools to support student learning (Borges et al., 2018). Learning to program particularly during the initial stages of STEM (Science, Technology, Engineering, and Mathematics) educationcan be especially challenging for novice students. Many struggle with planning and writing code, often perceiving programming concepts as complex and difficult to master (Okonkwo and Ade-Ibijola, 2023). This steep learning curve is frequently attributed to a limited understanding of the fundamental principles required to build even simple programs (Corritore and Love, 2020).

Recent technological advances and innovative pedagogical approaches have transformed educa-

tion. One key development is the adoption of ALMs (Sobral, 2021a), which promote student-centred learning through active participation. These methodologies have become widespread in the fields of STEM, including computer science (Liao and Ringler, 2023). Rooted in Constructivist theory, ALMs emphasize that learners build knowledge through experience, assuming greater responsibility and control over their learning process (Lima, 2016; Arık and Yılmaz, 2020; Elliott, 1996; Sasson et al., 2022). These methods enhance student engagement and support the development of practical skills (Garcia et al., 2021), as ALMs are defined as any instructional approaches in which students participate in activities beyond passive listening (Duffany, 2017).

Adopting ALMs for teaching programming has practical and successful implications for educators who wish to implement AL, as they provide students with challenges they may face in the job market (Garcia et al., 2021). Hence, there is a variety of ALMs and strategies for their adoption that can mitigate the difficulties instructors face regarding implementing AL in programming education (Calderon et al., 2021). It's important to recognize that the successful implementation of ALMs in programming education is not a haphazard process. It requires a certain degree of knowledge and meticulous planning. Understanding the various ways, whether successful or not, of implementing different strategies for adopting ALMs is a key step in this process.

This knowledge can serve as a solid foundation for educators seeking to incorporate new methodologies and active learning strategies into their programming courses. Therefore, some ALMs have been implemented in programming education in undergraduate computing courses so that students can handle the challenges they may face in the job market or develop greater autonomy in solving proposed problems and improving communication (Garcia et al., 2021).

2.4 Related Work

Researchers have sought ways to improve the adoption of ALMs through new communication and instructional technologies. In the educational context, several studies focus on developing digital repositories to support teaching practices across various fields. This section presents a detailed review of related works that align with the aims of this research, highlighting key contributions and identifying gaps that motivate the current study.

The ALCASYSTEM (Figure 2.1), developed by de Castro and Siqueira (2019), is an online portal designed to incorporate ALMs into computing education. Its objective is to support educators in transforming teaching practices through information and communication technologies by facilitating access to active methodologies and promoting student engagement and the development of essential skills. The platform allows instructors to search, classify, and share ALMs used in their courses. It offers open access, enabling users to add comments, perform keyword searches, and consult works and experience reports shared by other educators. These features aim to assist in the implementation of more interactive and learner-centred teaching practices.

However, based on user feedback collected during evaluations, some limitations of the system were identified. Instructors reported that applying an active learning technique often requires reading the entire article, which can be time-consuming and hinder quick implementation. The homepage design was considered outdated and not engaging, potentially reducing user interaction. There is also an absence of concise summaries or case studies to facilitate quick understanding of techniques without reading full papers. Additionally, the platform lacks collaborative features such as discussion forums or spaces for educator interaction, which limits experience sharing and debate. Although a keyword search function has been implemented,

it is still under development and may not provide fully efficient results. While navigation is generally good, there is room for improvement in terms of intuitiveness and ease of use. Lastly, the database currently contains 285 works, which could be expanded to include more techniques and recent publications to ensure better coverage and up-to-date content.

These limitations suggest that, although the portal is a useful tool, improvements in design, usability, information access speed, and collaborative features could enhance its practical use in teaching. The system enables the inclusion, search, selection, classification, and recommendation of ALMs, organised by computing disciplines, specific active learning techniques, and includes a publication forum.

Figure 2.1: ALCASYSTEM website

To conclude, ALCASYSTEM represents an initial effort to centralise and disseminate ALMs within the computing education community. By integrating a curated repository of academic works with search, classification, and recommendation features, the platform supports educators in identifying and applying relevant methodologies. Although the portal has been positively evaluated for its potential to promote alternative teaching practices, its continued development is essential to address the identified limitations and to ensure broader adoption and pedagogical impact.

Another relevant initiative is OpenSMALS (Open Repository for Teaching Software Modeling from Active Learning Strategies), proposed by Silva et al. (2020a). This tool was developed to support instructors in applying ALMs in the context of software modelling, particularly in the teaching of UML diagrams. OpenSMALS seeks to address common challenges such as students' difficulties in understanding abstract modelling concepts and the continued reliance on traditional instructional approaches due to time and resource constraints. The repository provides structured pedagogical guidance, modelling scenarios, and assessment instruments to assist instructors in implementing active strategies. It also includes a recommendation feature

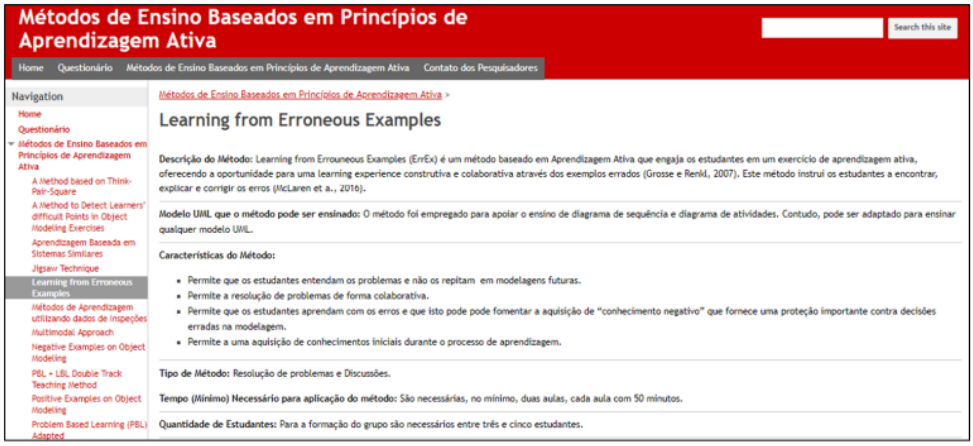


Figure 2.2: OpenSMALS website

that suggests appropriate methods based on questionnaire responses and enables educators to share and adapt strategies according to their teaching context.

Although the tool offers positive contributions—including support for contextualised application, availability of instructional materials, and potential to enhance student engagement—some limitations have been identified. These include the generality of some content, which may not meet specific instructional needs; increased complexity in lesson planning; and the requirement for instructors to be trained in the proposed methods. Additionally, its application has been primarily tested in small class settings, raising concerns about scalability to larger groups. The effectiveness of the tool also depends significantly on the instructor's prior experience and ability to adapt the strategies. Further research is necessary to validate its use in broader contexts and to improve its scalability. In summary, although the primary focus of OpenSMALS is on software modelling and instructional strategies for teaching UML diagrams, there is an indirect connection to programming education. By enhancing active learning strategies in modelling, the tool may also support the teaching of programming by facilitating the understanding of complex software development concepts.

Another relevant contribution is the preliminary guide proposed by Lima et al. (2021). Due to the length of its content, Figure 2.3 presents only steps 1 and 2 of the guide. Nevertheless, the guide is organised into four stages: step 1 identifies the student's learning style; step 2 diagnoses the students' soft skills; step 3 involves recognising the type and category of the ALM; and step 4 consists of selecting the most appropriate ALM. The guide aims to assist educators in choosing ALMs that align with students' learning styles and profiles, facilitating the integration of these methodologies into the teaching and learning process.

Developed using the DSR methodology, the guide underwent initial validation by experts, who assessed its clarity, ease of use, flexibility, and pedagogical suitability. It also incorporates strategies that connect academic learning with industry demands, promoting the development of both technical competencies and soft skills. However, the study does have some limitations. The initial validation was based on a convenience sample composed solely of academic specialists, which may restrict the external validity and overlook broader industry perspectives. The current version of the guide offers a preliminary overview but lacks in-depth information regarding the stages, benefits, and challenges of each active methodology, which could limit its practical

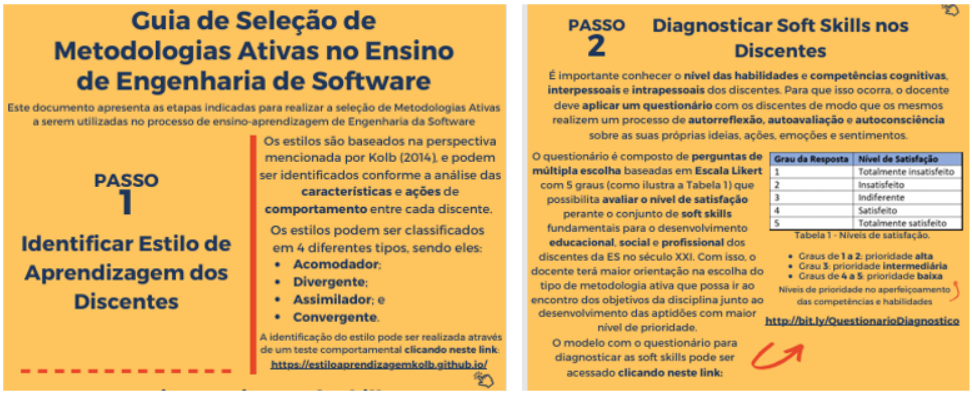


Figure 2.3: Guide for Assertive Selection of ALMs in ES

effectiveness.

Furthermore, although the guide supports both in-person and emergency remote teaching environments, it is not yet fully adapted to other modalities or more structured hybrid contexts. Lastly, the guide currently functions as a manual instrument requiring manual application and interpretation without integration into automated tools or software, which may impede scalability and ease of use in larger or more varied educational settings. Despite its contributions, the guide is not implemented as a web-based platform; instead, it is presented as an interactive PDF document. This format limits its accessibility, scalability, and potential for user interaction when compared to digital repositories or online systems. The absence of integrated digital features, such as searchable content, real-time updates, or collaborative tools, may restrict its practical adoption and long-term impact in broader educational contexts.

Another relevant work is the framework proposed by Ahshan (2021) to ensure active student engagement in remote and online teaching during the COVID-19 pandemic. This framework (Figure 2.4) is a conceptual model designed to guide educators in implementing activities and strategies that promote active engagement in remote learning environments. It combines adapted teaching pedagogy, educational technologies, and an e-learning management system to support interactive and participatory learning.

The model integrates ALMs, synchronous and asynchronous teaching methods, and content segmentation. It recommends the use of educational tools such as Google Meet, Jamboard, Google Chat, Breakout Rooms, Mentimeter, Moodle, and electronic writing devices. Moodle serves as the course management system. The framework begins with lesson bridging and a welcome message to engage students and connect lessons. It includes ConcepTest activities to assess and deepen students' understanding, facilitating interactions among students, content, and instructors. Learning objectives are clearly defined using action verbs based on Bloom's Taxonomy and linked to programme outcomes and graduate attributes to enhance engagement.

However, limitations include its focus on engineering courses, which may limit generalizability to other disciplines. The framework's dependence on specific educational technologies may restrict its applicability in environments with limited technological resources. Additionally, the study did not evaluate long-term learning outcomes or challenges in scaling the framework for larger groups. In summary, the framework provides a practical guide for educators to enhance

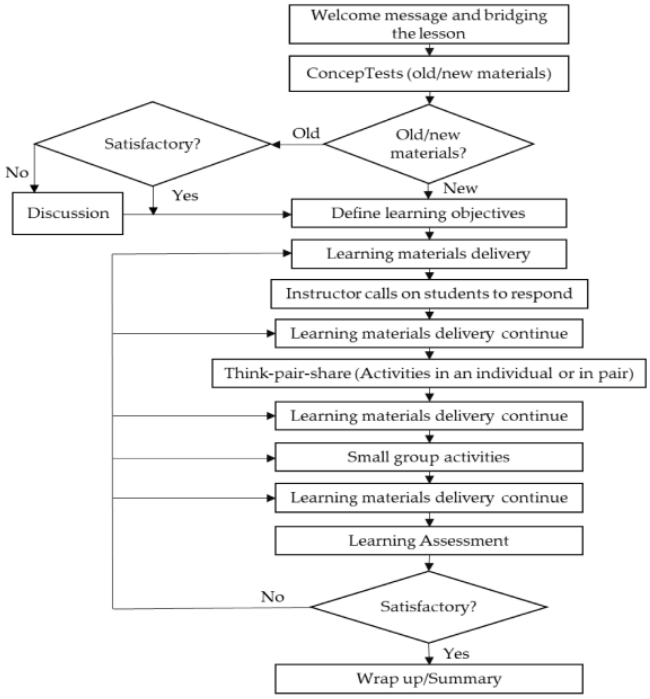


Figure 2.4: The framework layout

active student engagement in remote and online teaching contexts, especially relevant during the COVID-19 pandemic and beyond.

Recognizing the increasing demand for specialized support in programming education, it is important to highlight that existing resources, present in Table 2.1, although valuable for educators, tend to focus on specific domains such as software modeling, or offer broad repositories of articles without tools dedicated to programming instruction.

Table 2.1: Summary of related work

Author(s)	**Focus / Description**	**Key Results / Findings**
de Castro and Siqueira (2019)	ALCASYSTEM: online portal for ALMs in computing education	Supports access to methodologies, promotes engagement and skill development; limitations in usability, lack of collaborative features, database size
Silva et al. (2020b)	OpenSMALS: ALMs for software modeling and UML teaching	Provides pedagogical guidance, recommendation system, supports engagement; limitations include scalability, content generality, manual implementation
Ahshan (2021)	Framework for active engagement in remote learning	Integrates ALMs with e-learning tools, defines objectives with Bloom's taxonomy; limited to engineering courses, technology-dependent, scalability not tested
Lima et al. (2021)	Preliminary guide for ALM selection	Stepwise process for matching ALMs to students' learning styles and soft skills; manual, descriptive guide without digital integration
This study	CollabProg: curated repository of ALMs for programming education	Centralizes strategies for programming courses, optimizes instructor workflow, integrates multiple methods in one platform; addresses gaps of previous tools in accessibility, specificity, and ease of use

Consequently, this executive summary presents a novel contribution through CollabProg. Unlike other platforms, CollabProg centralizes a curated collection of strategies specifically developed

to facilitate the adoption of different active learning methodologies in programming education. This integrated repository aims to optimise educators' work by eliminating the need to search for information across multiple sources, thus enhancing the implementation of effective pedagogical practices. The following sections provide a detailed description of CollabProg and its potential impact.

2.5 Chapter Conclusion

This chapter provided an overview of the theoretical concepts related to programming education in Computer Science and the use of ALMs. It addressed the central role of programming education within computing technologies and highlighted the significant challenges faced by students, including difficulties with fundamental concepts and the prerequisite skills in logic and mathematics. The chapter also discussed the limitations of traditional teaching methods, which are often instructor-centred and focused on passive knowledge transmission, thereby failing to adequately foster logical reasoning and higher-order thinking skills.

Additionally, the chapter examined the application of ALMs in programming education, highlighting their potential to improve student engagement and learning outcomes. The review of related works revealed a range of pedagogical innovations, including ALCASYSTEM, which incorporates ALMs into computing disciplines, and OpenSMALS, a tool designed to implement active learning strategies in software modelling education. It also discussed the Preliminary Guide by Lima et al. (2021) and the Framework proposed by Ahshan (2021), both of which reinforce the importance of adapting teaching methodologies to diverse educational contexts and student profiles. Together, these discussions establish a comprehensive foundation for this study, highlighting both the challenges in programming education and the promising strategies offered by ALMs. This background and related works support the development of the proposals presented in the subsequent chapters.

Chapter 3

Relevance Cycle

This chapter presents the results of the Relevance Cycle. It reports on a Systematic Mapping Study and a national survey conducted to investigate instructors' use of Active Learning Methodologies in programming education and the challenges encountered in their implementation.

3.1 Studies Conducted

This research began in March 2020. To conduct experimental studies with the academic community, the research project was submitted to the Research Ethics Committee (CEP) of the Federal University of Amazonas (UFAM) and approved under opinion no. 4.694.031.

The first study was conducted as a Systematic Mapping Study (SMS), following the procedures outlined by Kitchenham (2012): planning, conducting, and analysing the results. The goal of the SMS was to summarise and characterise the ALMs employed in teaching computer programming in undergraduate computing courses. This study provides an overview of the current scenario and profiles the research adopting different ALMs in programming education. It also identifies the topics or course content, tools, programming languages, and metrics reported in the publications.The details of this part of the Relevance cycle can be seen in Appendices A, B and C.

Additionally, a survey was conducted. As stated by Kitchenham and Pfleeger (2008), surveys are suitable for capturing and summarising the views and experiences of a defined population. This survey targeted faculty members involved in teaching programming using ALMs. This approach was appropriate for identifying patterns and trends within a specific educational context. The survey aimed to explore instructors' perceptions regarding the use of ALMs in programming education, with particular attention to the challenges and difficulties encountered during implementation. The detailed conduct of the study is presented in Appendix D.

3.2 Systematic Mapping Study

By conducting the SMS, a deeper understanding of the research problem can be obtained. SMS is a method used to categorise and summarise existing information on a research question in an unbiased manner (Kitchenham and Charters, 2007). Based on the results of the SMS, it

is possible to identify the state of the art and research gaps, thereby enabling the suggestion of areas for further investigation. For this SMS, the following Research Question (RQ) was formulated: **How have instructors used active learning methodologies while teaching programming in undergraduate courses?**

3.2.1 Findings from the SMS

Responding to the research question, a total of 3,850 publications were identified through a meticulous search process. After rigorously applying the selection criteria, 81 publications were accepted for analysis. From these, 37 types of ALMs were identified. Among them, 17 publications reported the use of mixed methodologies, such as Flipped Classroom and Problem-Based Learning. The Flipped Classroom (FC) was cited in 14 publications, while Gamification-based Learning (GM) appeared in 11. Problem-Based Learning (PBL) was employed in eight publications, and Game-Based Learning (GBL) in five. Four publications described author-developed ALMs, and four others used Project-Based Learning (PjBL). Figure 3.1 presents the types of ALMs mapped in this study.

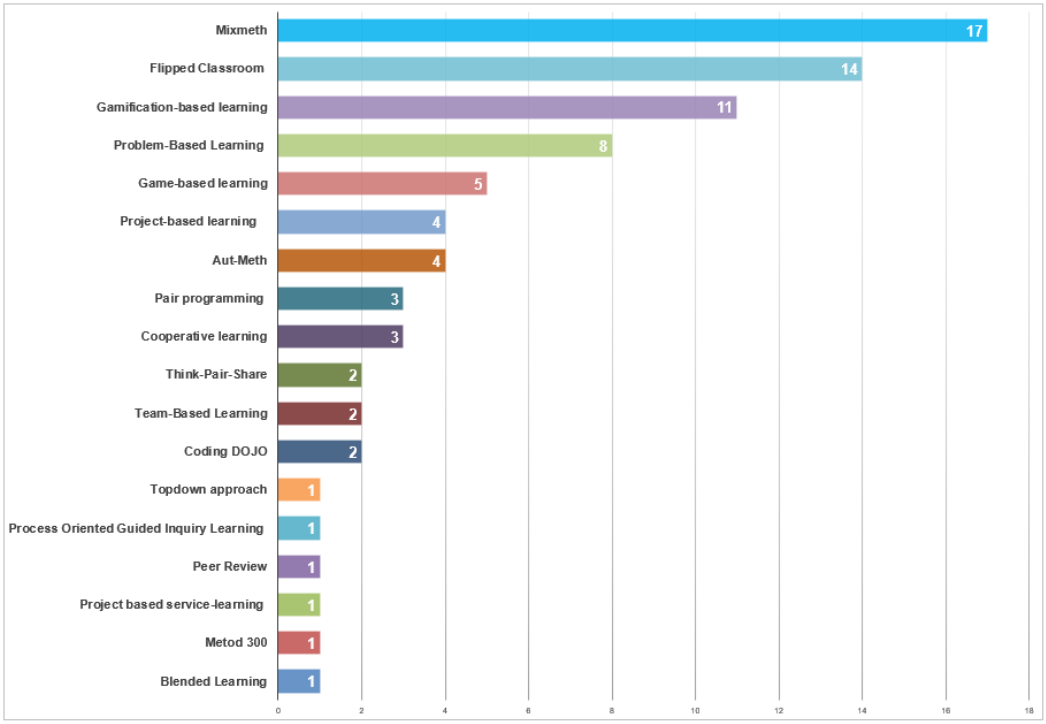


Figure 3.1: Types ALMs adopted for teaching programming.

Other identified methodologies include Cooperative Learning (CL) and Pair Programming (PP), each cited in three publications; Team-Based Learning (TBL), Think-Pair-Share (TPS), and Coding Dojo (Dojo), each cited in two publications; as well as Blended Learning (BL), Peer Review (PR), Project-Based Service Learning (PBSL), Method 300 (M300), Process-Oriented Guided Inquiry Learning (POGIL), and Top-Down Approach (TopD), each cited in one publication. Building on this categorisation, the ALMs reported in the publications were

analysed and classified by type, revealing 37 distinct ALMs adopted for teaching programming. According to Katona and Kovari (2016), numerous approaches have been developed in recent decades to enhance student learning outcomes through ALMs. This is especially relevant for programming courses, where regular practice is essential for mastery.

Among the ALMs mapped, 17 publications reported approaches that combined more than one ALM. These were classified as "Mixed Methodologies" (MixMeth), as presented in Table 3.1. Additionally, four publications proposing new instructional strategies were categorised as "Authors' Methodologies" (Aut-Meth), in which instructors adopted distinct teaching practices to incorporate active learning into their course planning. Mixed methodologies appeared in 20.9% (17) of the mapped publications, where authors integrated ALMs such as FC and PBL. Some studies also combined POGIL with PP for programming instruction. These combinations indicate an effort by instructors to explore alternative pedagogical strategies through the integration of multiple ALMs. Consequently, the studies reported diverse instructional experiences and expanded opportunities for promoting active student engagement.

Table 3.1: Methodologies adopted jointly

ID	ALM	#Publications
1	Flipped Classroom (FC) + Project-Based Learning (PBL)	S01
2	Mini-lecture + Live-coding + In-class coding (InCconding)	S03
3	Pair programming (PP) + Exercise-based learning (EBL)	S05
4	Flipped Classroom + Problem-based Learning	S12, S15
5	Animated Flowchart with Example Think-Pair-Share	S16
6	Project-based learning (PjBL) + SCRUM	S23
7	Student Ownership of Learning (SOL) + Flipped Classroom +	S26
8	Pairing-based pedagogy (Pairgogy) - Pairing-Based Approach (Pair programming + Blended Learning (BL)	S27
9	Flipped Classroom + Team-Based Learning (TBL)	S28
10	Process Oriented Guided Inquiry Learning (POGIL) + Pair Programming	S35
11	Process Oriented Guided Inquiry Learning + Pair Programming	S21
12	Game-based learning (GBL) + Problem-based learning	S43
13	Lecture-based Learning (LBL) + Problem-based Learning + Peer Instruction (PI)	S46
14	Flipped Classroom + Gamification-based learning (GM)	S65
15	Blended teaching + Problem-Based Learning + Task driven + Flipped classroom	S70
16	Learning by collaboration, flipped classroom, game-based learning	S73
17	Flipped Classroom, Peer Discussion, and Just-in-time	S76
18	Coding Dojo, Gamification, Problem-based Learning, Flipped Classroom and Serious Games	S81

The FC was reported in 17.5% (14) of the publications. In this methodology, in-person activities typical of traditional instruction are shifted to extracurricular settings, while theoretical content is accessed in advance through digital resources (Hendrik, 2019). GM appeared in 13.5% (11) of the analysed studies. According to Venter (2020), GM is considered a promising educational approach for this decade, as instructors worldwide acknowledge that the effective design of gamified activities can enhance students' productivity and creativity. PBL appeared in 9.8% (8) of the publications. This student-centred methodology encourages learners to conduct research, integrate theory and practice, and apply knowledge and skills to solve defined problems (Chang et al., 2020). GBL was reported in 6.1% (5) of the studies. GBL involves the use of educational games designed to balance the development of specific competencies with gameplay dynamics (Qian and Clark, 2016). It has been applied in various areas of computer science education, including software engineering, programming, and cybersecurity (Zhang-Kennedy and Chiasson, 2021).

Also, Aut-Meth was identified in 4.9% (4) of the publications, including studies S26 and S32. In these cases, the authors developed and implemented their own ALMs to promote collaboration and active learning in programming education. Project-Based Learning (PjBL) appeared in the

same proportion, in 4.9% (4) of the publications. PjBL is a student-centred approach in which learners construct their knowledge through the development of projects (Paristiowati et al., 2022). Lastly, 12 other types of methodologies were reported in fewer than four publications: CL and PP were identified in three studies; TBL, TPS, Dojo in two; and BL, PR, PBSL, M300, POGIL, and TopD in one publication each.

After analysing the data extracted from the publications selected for this research, the state of the art regarding the adoption of ALMs in computer programming teaching was characterised. This characterisation can support the development of new research by providing a foundation for selecting and improving different methodologies in teaching practice. Consequently, it facilitates the generation of knowledge and the construction of studies aimed at testing or creating methods to assist instructors in programming education.

Based on this analysis, the mapped ALMs were organised and categorised, enabling the development of CollabProg, an open and collaborative repository. This repository allows instructors to identify, select, adopt, discuss, comment on, evaluate, and collaborate on the use of both new and established ALMs in programming education. As a result, a set of step-by-step guidelines was developed and made available to support instructors in adopting these methodologies according to their specific teaching contexts. This practical resource reduces the need for instructors to consult multiple scientific articles or books when seeking appropriate ways to implement a given ALM in the classroom, thereby promoting more efficient and well-informed pedagogical decisions. The CollabProg repository and its features will be detailed in the next chapter of this work.

All details of the SMS results can be found in Calderon et al. (2023a). The study highlights the importance of using ALMs in programming education, demonstrating that these methodologies engage students in active, participatory, and contextualised learning. The analysed studies indicate that ALMs support students in applying knowledge to real-world problems, fostering the development of practical and critical skills relevant to programming. The variety of subjects addressed in combination with different ALMs reflects the interdisciplinary nature of programming and the need to prepare students for complex and diverse challenges.

3.2.2 Acceptance Criteria for CollabProg

Based on the results identified in the SMS, we defined two kinds of acceptance criteria for CollabProg: Design Criteria and Behavior Criteria. These acceptance criteria represent a set of expectations that guide both the development and the evaluation of the repository. The Design Criteria describe what the artefact should offer its users in terms of structure, content, and functionality. The Behavior Criteria, in turn, refer to the expected contributions of the artefact to programming teaching practices, particularly in fostering the understanding and adoption of active learning methodologies. Defining these criteria was essential to ensure that the design of CollabProg was evidence-based and aligned with the pedagogical needs identified in the literature. The criteria are detailed below.

The **Design Criteria** specify what CollabProg should offer its users:

- **DC1** – The artifact should provide a variety of ALMs, including detailed descriptions, application examples, and usage contexts.
- **DC2** – The artifact should present clear and structured guidelines for implementing each ALMs, covering aspects such as planning, execution, and evaluation.

- **DC3** – The artifact should provide curation of ALMs (the process of carefully selecting and organizing ALM content), including critical analyses, evidence-based recommendations, and feedback from other instructors who have already implemented these methodologies.

The **Behavior Criteria** are related to contributing to the teaching practices of programming and are as follows:

- **BC1** – The artifact should help the user deepen their understanding of ALMs by providing educational resources such as tutorials, case studies, and explanatory videos.
- **BC2** – The artifact should motivate instructors to adopt ALM by presenting evidence of effectiveness, observed benefits in other institutions, and success stories.
- **BC3** – The artifact should present clear and detailed information about ALMs, including pedagogical objectives, detailed implementation steps, and possible challenges with suggested solutions.
- **BC4** – The artifact should provide detailed and practical guidelines, using accessible language and concrete examples to facilitate implementation across different disciplines and levels of education.

The acceptance criteria defined for CollabProg serve as a reference for both its development and assessment. These criteria will be verified during the evaluation stages conducted throughout the design cycles. By analysing how well the tool meets the established Design and Behaviour Criteria, it is possible to identify strengths, limitations, and opportunities for improvement, ensuring that CollabProg effectively supports programming education through the use of ALMs.

3.3 Survey

To gather new insights from instructors regarding the adoption of ALMs in programming education, a survey was conducted with faculty members from Computer Science programs across Brazil. The survey aimed to explore their perceptions of using ALMs and to identify the challenges and difficulties encountered when implementing these methodologies in programming classrooms. For this purpose, the survey research method was employed, using a questionnaire as the primary tool to collect instructors' perceptions. According to Kitchenham (2012), a survey is a research method designed to summarise and understand the characteristics under investigation within a broad population. The target audience comprised higher education instructors experienced in teaching programming using ALMs. Further details on the survey planning are available in Calderon et al. (2024a).

3.3.1 Survey Design

For the construction of the survey, the guidelines suggested by Coelho et al. (2020) were followed, ensuring the questions were developed in a logical and coherent sequence. The questions are presented in Table 3.2. The survey was organised into different sections to investigate the adoption of ALMs in programming education. Initially, information about participants' profiles and experience was collected, including their current position and the length of time they have been adopting ALMs. Participants were then asked about their specific use of ALMs in teaching programming, followed by questions identifying the methodologies they have used and the subjects in which these were applied.

The survey also explored the types of tools and platforms used in the teaching process. Additionally, it investigated participants' motivations, perceived benefits, difficulties encountered, and challenges related to adopting these methodologies. This structure aims to provide a comprehensive understanding of the implementation and impact of ALMs within the specific context of programming education.

Table 3.2: Questions created for the survey.

ID	Question
Q01	Questions regarding the participants' profile (gender, state of teaching, academic qualifications, etc.) and experience (teaching computing, teaching programming courses, adopted programming languages).
Q02	Do you use any type of ALM for teaching programming?
Q03	How long have you been using ALM in programming teaching?
Q04	Which ALMs have you used in your programming classes?
Q05	What is your motivation for adopting ALMs for programming teaching?
Q06	What are the perceived benefits of adopting ALMs for programming teaching?
Q07	Have you encountered any difficulties in adopting ALMs?
Q08	What are the main challenges and drawbacks faced when using ALMs in programming teaching?

3.3.2 Findings from the Survey

The study gathered responses from 102 instructors across 21 states and the Federal District, covering a total of 22 federative units. The regional distribution of participants was as follows: North 37.2%, Northeast 14.9%, Southeast 11.%, South 9.7%, and Central-West 6.9%. The highest concentration of instructors was in the North region, with Amazonas (23.5%), Rondônia (21.6%), and Acre (7.8%) leading, followed by Minas Gerais (6.9%) and Alagoas (5.9%). Also, analyzing the profile of the participants, it was observed that the majority identified as male, comprising 62.7% (64) of the respondents, while 37.3% (38) identified as female. Regarding age distribution, most instructors fell within the 41 to 45-year-old range (28.4%), followed by those aged 46 to 50 years (26.6%) and 36 to 40 years (18.6%), indicating a predominance of professionals in the more advanced stages of their careers.

Regarding the type of institution, 77.5% (79) of the instructors work at public institutions, 19.6% (20) at private institutions, and 2.9% (3) at community-based institutions, reflecting the predominance of the public sector in educational provision. As for academic qualifications, 50% (51 instructors) hold a Master's degree, followed by 34.3% (35) with a PhD, Postdoctoral degree with 6.9% (7), 5.9% (6) with a Bachelor's degree, and 2.9% (3) with a specialisation. The high level of academic qualifications among the instructors is a positive indicator for the adoption of ALMs, as it is often associated with a greater interest in innovative pedagogical practices. Concerning classroom experience, most instructors reported 10 years of teaching experience (11.8%), followed by 11 years (6.9%), indicating a solid professional trajectory. In terms of experience specifically in teaching programming, 9.8% of instructors reported 5 years of experience, while 8.8% reported between 2 and 10 years, reflecting a diversity of experience levels within the sample.

As for the adoption of ALMs, the results show that 78.7% of instructors reported using or having used some type of ALM in programming education. Additionally, 21.2% of participants have been using ALMs for three years; 17.5% for four years; 16.% for five years; 15% for two years; and 8.8% for eight years. These findings indicate a growing trend in the adoption of ALMs in programming education, with many instructors having implemented these practices over a considerable period. The results also highlight that Brazilian instructors use ALMs as tools to facilitate learning, guiding and supporting students throughout their processes of discovery and knowledge construction.

Figure 3.2 illustrates the scenario regarding the ALMs employed by instructors in programming education (Q4). The ALMs **always** used in programming classes include: PBL (26), GM (19), PjBl (14), Dojo (19), and PR (5). The ALMs **almost always** used are: PBL (26), GM (21), PjB (20), PP (13), TBL (11), and Dojo (9). The ALMs **sometimes** adopted by instructors include: PjBL (14), TBL (12), GM (12), PBL (11), PR (8), Dojo (8), and PR (5). These results demonstrate a diverse and frequent use of ALMs in programming classes, indicating a significant shift in teaching strategies. This shift aims to enhance student engagement and performance through more participatory and collaborative learning approaches.

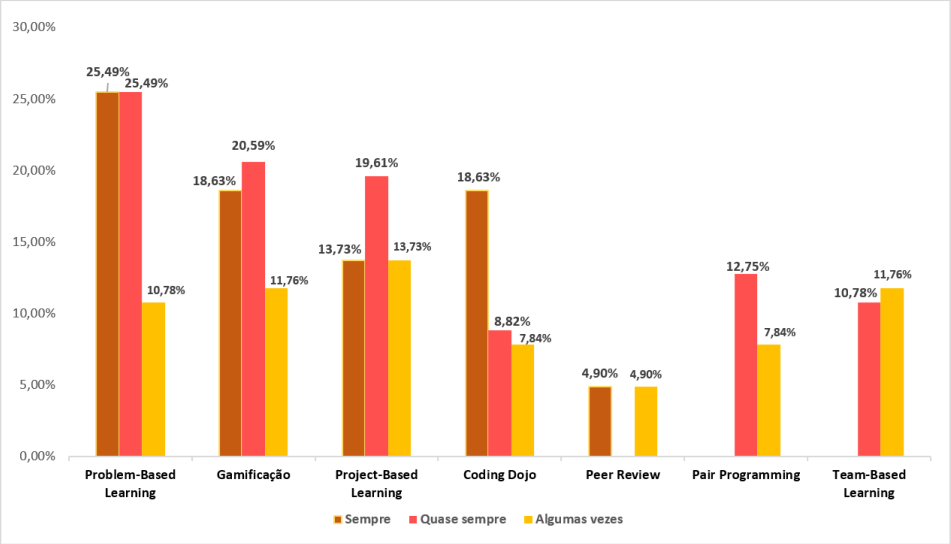


Figure 3.2: Frequency of ALM use in programming classes.

The analysis of the main motivations indicated by instructors for adopting ALMs in programming education reveals a range of factors influencing this decision (see motivations in Table 3.3). The key motivations highlighted by instructors include: increasing student engagement (56.3%), aligning content with real-world practice (55%), and the opportunity for innovation in teaching practice, as well as enabling students to create, adapt, and modify algorithms or code (each cited by 46.3%). These results reflect a pursuit of more effective and engaging teaching methods that offer students a more meaningful and relevant learning experience. These findings underscore the importance of practical teaching approaches that emphasise the application of knowledge in real-world contexts, preparing students not only to understand theoretical concepts but also to apply them effectively in professional settings.

Moreover, the motivations related to making classes more dynamic (40%), conducting short and frequent assessments (32.5%), and adapting to students' skills and needs (27.5%) underscore the importance of a personalised and flexible approach to programming education. This approach enables instructors to address diverse student needs and learning styles, fostering a more inclusive and effective learning environment. It is evident that instructors' motivations reflect a desire to promote a more engaging, practical, and relevant education, effectively preparing students for job market challenges and supporting their academic and professional development.

Table 3.4 presents the Positive Perceptions (PP) reported by instructors regarding the adoption of ALMs in programming education. The analysis of these positive aspects highlights a range

Table 3.3: Main motivations reported by instructor

ID	Motivation	Percentage (%)
M01	Increased student engagement in learning programming	56.3
M02	Alignment of content with their reality and teaching practice	55.0
M03	Opportunity for innovation in teaching practice	46.3
M04	Opportunity for students to create, adapt, and modify algorithms or code	46.3
M05	Replacement of traditional lecture-based classes	43.8
M06	Opportunity to develop skills for professional practice	41.3
M06	Active teaching practice that makes classes more dynamic	40.0
M07	Opportunity to conduct short and frequent assessments	32.5
M08	Adaptation to students' skills and needs	27.5
M09	Opportunity for collaborative knowledge construction	27.5
M10	Traditional teaching methodologies do not provide teachers with tools to improve content teaching in programming courses	26.3
M11	Connection between content and its application in students' daily lives	26.3

of perceived benefits for students and the overall classroom environment. Student motivation to learn the content (86.3%) was identified as the primary benefit, suggesting that ALMs can foster deeper and more meaningful interest in the subject matter. Additionally, student engagement in the classroom (78.8%) was the second most frequently cited benefit by instructors. Other well-rated benefits include improvements in students' ability to read code (70%) and to understand how programming instructions work (65%), indicating that instructors perceive ALMs as contributing to the development of key practical and technical programming skills. Collaboration among students during content learning (46.3%) and the resolution of challenges individually or in groups (33.8%) further underscore the role of ALMs in promoting teamwork and knowledge sharing among learners.

Table 3.4: Positive perceptions reported by instructor on adopting ALMs.

ID	Category	Percentage (%)
PP01	Motivation to learn content	86.3
PP02	Student engagement in the classroom	78.8
PP03	Improvement in code reading ability	70.0
PP04	Improvement in understanding programming instructions	65.0
PP05	Improvement in individual performance	51.2
PP06	Collaboration among students during learning	46.3
PP07	Improvement in class performance	45.0
PP08	Challenges solved individually or in groups	33.8
PP09	Improvement in interaction among students	33.8
PP10	Improvement in skill development	32.5
PP11	Students' ability to generate problem-solving alternatives	31.3
PP12	Improvement in student participation in class	30.0
PP13	Improvement in interaction between teacher and students	28.7
PP14	Students' ability to evaluate solutions	27.5
PP15	Students' ability to break problems into smaller modules	27.5
PP16	Knowledge sharing among students	25.0
PP17	Students' willingness to solve problems	21.3
PP18	Application of theory in practical activities	18.8
PP19	Students' ability to compare alternatives	15.0

Instructors also reported that ALMs improve individual student performance (51.2%) and overall class performance (45%), suggesting that these methodologies can contribute to better academic outcomes. Students' ability to develop skills, generate alternative solutions to problems, evaluate the solutions found, and break down problems into smaller modules indicates an enhancement of their cognitive and analytical competencies. The findings suggest that instructors broadly perceive the adoption of ALMs as beneficial, particularly in terms of fostering student motivation and engagement. These elements are essential for active and participatory learning, which are key to the effectiveness of programming education.

The results obtained from the survey conducted with 102 faculty members from various regions of Brazil provide a solid foundation for the development of CollabProg, an open and collaborative repository designed to support the adoption of ALMs in programming education. The data revealed that, despite the recognised benefits of ALMs such as promoting practical understanding, enhancing programming skills, and fostering essential competencies like teamwork and critical thinking—instructors still face significant challenges. These include a lack of specific training, insufficient technological support, and difficulties in implementing the various stages of ALMs.

These challenges highlight the urgent need for more resources and support to enable instructors to integrate these methodologies more effectively. The perspectives shared by faculty indicate the importance of institutional policies that promote continuous professional development, provide appropriate technological infrastructure, and ensure support for the successful implementation of ALMs. Based on the data collected, CollabProg can serve as a valuable tool to help address these barriers by offering practical guidelines, encouraging the sharing of experiences and resources among instructors, and supporting the more efficient adoption and application of ALMs.

3.4 Chapter Conclusion

This chapter presented the results of two complementary studies that contributed to advancing the use of ALMs in programming education. The first, a SLM, identified and categorised 37 distinct ALMs employed by instructors in programming courses. Based on an analysis of 81 publications, this study offered a broad overview of the methodologies currently adopted in the field, including prominent approaches such as FC, GM, PBL, and PjBL. The categorisation and organisation of these methodologies provided a foundational framework to initiate the research design cycle. Based on these results, the development of CollabProg was undertaken to support instructors in adopting ALMs in programming education. The second study, an exploratory evaluation of CollabProg 1.0, examined the perceptions of instructors regarding the platform's usefulness, ease of use, and intention to use. The findings confirmed the system's alignment with the defined acceptance criteria and highlighted its potential to facilitate the planning and reporting of ALM-based practices in programming courses. Together, these studies demonstrate the relevance and feasibility of using a structured repository to enhance the adoption of active methodologies in the computing education context.

The second study involved a survey with 102 instructors from different regions of Brazil, offering empirical insights into the practical application of ALMs in programming courses. The findings indicate a growing trend in the adoption of these methodologies, with the majority of instructors reporting current or previous use of ALMs. However, the data also revealed persistent challenges, including a lack of targeted training, limited access to technological infrastructure, and difficulties in operationalising the stages of ALMs. These constraints underscore the necessity for institutional support, encompassing ongoing professional development, the provision of adequate resources, and the implementation of policies that encourage and sustain the use of ALMs.

Together, these studies provide a robust foundation for the design and implementation of CollabProg, a tool conceived to address the challenges identified and promote broader adoption of ALMs in programming education. By offering practical guidance, encouraging collaboration among instructors, and facilitating the sharing of experiences and educational resources, CollabProg seeks to contribute to the improvement of teaching practices and student learning outcomes. The evidence gathered from both the SLM and the survey reinforces the relevance

of the CollabProg design cycle. Instructors highlighted the positive impact of ALMs on student motivation, engagement, performance, and the development of technical and collaborative skills. These insights are crucial for informing the design of more effective and inclusive teaching approaches, which will be detailed in the next chapter, where the features, functionalities, and implementation strategies of the CollabProg repository are presented.

Chapter 4

Design Cycle

This section addresses the conception, evaluation, and refinement process of CollabProg. CollabProg was developed to mitigate instructors' practical difficulties in adopting ALMs in programming teaching by providing specific guidelines for their implementation. It is a collaborative and open repository designed to support instructors in the adoption of ALMs in programming education.

4.1 First Design Cycle: Conception and Initial Validation

4.1.1 Organization of Knowledge about the Methodologies

The first design cycle involved organising the knowledge base on ALMs. The results obtained through the SMS enabled the identification and categorisation of the ALMs adopted by instructors, as well as the recognition of positive evidence regarding their application in programming education. Following the identification of these methodologies, we drew on the approaches proposed by Sobrinho et al. (2016) and Silva et al. (2020a) to structure the knowledge of each ALM in a conceptual model represented by a class diagram. To construct this model, we initially defined the domain and scope of the knowledge based on the results presented in Calderon et al. (2024b), which aimed to identify and categorise the types of methodologies adopted by instructors for teaching programming.

According to Sobrinho et al. (2016), the domain refers to the semantic representation and formalisation of teaching methodologies based on active learning principles, which are grounded in constructivist theory, emphasising that students construct knowledge through experience, interaction, and reflection. Doolittle et al. (2023) further explain that active learning places learners at the centre of the educational process by engaging them in contextualised tasks, collaborative interaction, reflective practice, and the use of prior knowledge. The scope of this model is to support instructors in higher education programming courses by providing organised and semantically structured knowledge, thereby facilitating the dissemination and adoption of ALMs. Accordingly, the information collected about the ALMs was structured into a conceptual model, represented as a class diagram shown in Figure 4.1.

In the model, the **Category** class represents the category of ALMs according to the method's approach. The class can be instantiated, for example, with the name Cooperative Learning,

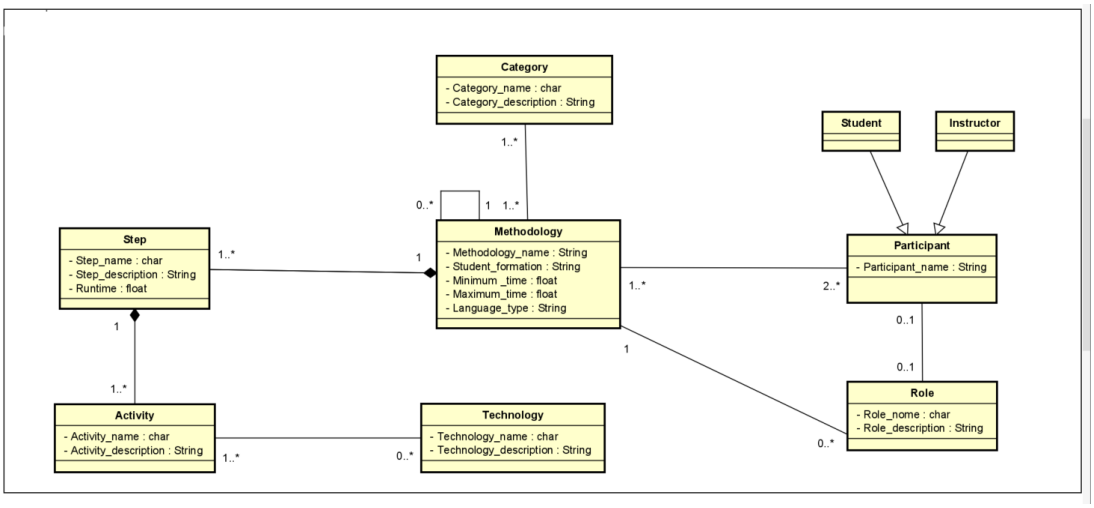


Figure 4.1: CollabProg solution model.

which includes methodologies that promote learning through cooperation among students. This category groups methodologies that focus on interaction between participants as a central element of the teaching process. When instantiated, the class is associated with methodologies aligned with this approach, such as PI and TPS, enabling the organisation and semantic representation of these relationships within the conceptual model. This class is associated with the **Methodology** class, which represents the ALMs included in CollabProg. As observed in the SMS, methodologies can be used together to improve or complement the outcomes of programming teaching. A self-relationship in the Methodology class represents this possibility. The **Step** class represents the necessary steps for adopting methodologies. The **Activity** class describes the actions to be carried out in each step of implementation in the classroom, such as planning content, presenting the methodology, and defining roles. The **Technology** class represents the possible educational technologies that can be used in each activity, including virtual environments or games. Finally, the **Participant** and **Role** classes are associated with each other and linked to the Methodology class to define the roles involved in each methodology. The details of this part of the design cycle can be seen in Appendices E and F.

4.1.2 Selection and Curation of Active Learning Methodologies

After organising the knowledge on ALM, we conducted a curation of the information related to the methodologies used in CollabProg. We examined scientific evidence and experimental studies demonstrating the application of ALMs in the classroom. The studies analysed were those selected during the SMS. Content curation is important because instructors often have difficulties identifying the origin of information, which affects the evaluation of its accuracy and authenticity (Correia, 2018). To avoid frustration among repository users, who are the instructors, we ensured that the available content is based on scientific experiments and relevant to the repository's purpose. The curation focused on studies providing analysis to assist instructors in implementing ALMs in the classroom, especially in programming education. We selected ALMs supported by scientific evidence, excluding those without experimental support or theoretical relevance. CollabProg aims to provide strategies for adopting ALMs and to ensure the quality and relevance of the knowledge shared. This allows the teaching community to

access resources to support their pedagogical practices in programming education.

We defined a set of Quality Assessment Criteria (QAC) to evaluate the quality of primary studies selected in the SMS for the development of CollabProg. The QAC assess studies on the adoption of ALMs in programming education in higher education, specifically in Computing. These criteria measure the relevance of each study for the content included in CollabProg. Table 4.1 presents the QAC and the scores each primary study can receive. We established six criteria to collect detailed information from primary studies to support instructors applying the methodology in the classroom. The criteria classify the studies as **Strong**, **Medium**, or **Weak** according to this scoring scheme: **Strong Description = 2, Medium Description = 1, Weak Description = 0**. It is important to note that not all studies meet every criterion or fall neatly into one of these categories. The classification depends on the level of detail provided in the study regarding the description of the ALM.

For example, a study may receive a **Strong Description** score if it clearly describes and specifies which metrics were used to evaluate improvement in programming teaching. Conversely, a **Weak Description** score applies when the study lacks relevant information, such as failing to mention the programming language used. An **Intermediate** or **Medium Description** is assigned if the study provides an incomplete description of the metrics used to evaluate improvement, as seen in QAC4 and QAC5. It is also worth noting that some criteria, such as QAC4 and QAC5, do not include a **Weak Description** category, reflecting the nature of the information they require. Among the defined criteria, QAC1 and QAC2 address aspects essential for the implementation and understanding of ALMs in the classroom. For this reason, studies must obtain the maximum score in both criteria to be included in CollabProg. Studies that do not meet this condition are excluded. For the other criteria, which may receive a weak rating, the absence of information in the primary study does not compromise the use of the methodology by CollabProg users. The complete protocol used to conduct the Quality Assessment (QA) of the primary studies is available online[1]. The criteria are listed below:

- **QAC1 – Description of Active Methodology**. This criterion should be strong, as studies should provide detailed information about ALMs and its benefits, allowing ColabProg users to understand better the methodology they want to adopt.
- **QAC2 – Adoption Support**. This criterion should be strong, as studies should provide clear and practical guidance on the steps necessary for ColabProg users to implement and adopt the methodology in their classrooms. This approach will instill a sense of confidence and capability in the users.
- **QAC3 – Metrics**. This criterion can be weak, so studies that present the metrics used to assess the methodology's effectiveness in improving teaching and learning are needed.
- **QAC4 – Programming Language**. This criterion can be weak, as it aims to identify the programming language used during the methodology's implementation.
- **QAC5 – Teaching Modality**. This criterion must be more robust to identify the teaching modality (face-to-face, blended learning, or distance education) in which the methodology was implemented.
- **QAC6 – Results Description**. This criterion needs to be stronger, seeking solid empirical evidence on the results of implementing the ALMs. This emphasis on empirical evidence will ensure the audience of the studies' validity and reliability.

[1] https://figshare.com/s/794c9f7e5adfdff915d1

Table 4.1: Quality Assessment Criteria X Publication Score

Criteria	Description of the criterion	Score
QAC1	**Strong Description:** If the methodology is clearly identified and described comprehensively, with information related to its concept, origin, objective, characterization, application, and the benefits of implementation in the classroom.	**2**
	Medium Description: If the methodology description is partially or incompletely described, with not all information related to its concept, origin, objective, characterization, application, and the benefits of implementation in the classroom being provided.	**1**
	Weak Description: If no descriptions related to the concept, origin, objective, characterization, application, and the benefits of implementation in the classroom are mentioned.	**0**
QAC2	**Strong Description:** If it describes in detail the steps for implementing the methodology in the classroom, clearly presents the step-by-step process to be followed for adopting the methodology, and provides relevant information about the tools and/or technologies used during the adoption of the methodology.	**2**
	Medium Description: If it describes incompletely and with few details the steps for implementing the methodology in the classroom, presents the step-by-step process for adopting the methodology incompletely and provides incomplete information about the tools and/or technologies used during the adoption of the methodology.	**1**
	Weak Description: If it does not describe the steps for implementing the methodology in the classroom, the step-by-step process for adopting the methodology, and information about the tools and/or technologies used during the adoption of the ALM.	**0**
QAC3	**Strong Description:** If it clearly describes and specifies which metrics were used to evaluate the improvement in programming teaching.	**2**
	Medium Description: If it describes incompletely which metrics were used to evaluate the improvement in programming teaching.	**1**
	Weak Description: If it does not describe or specify which metrics were used to evaluate the improvement in programming teaching.	**0**
QAC4	**Strong Description:** If it describes information that allows identifying and characterizing the type of programming language used.	**2**
	Weak Description: If no relevant information allowing the identification of the programming language used is mentioned.	**0**
QAC5	**Strong Description:** If it presents complete, clear, and relevant information about the teaching modality where the methodology was implemented.	**2**
	Weak Description: If it does not present the teaching modality where the methodology was implemented.	**0**
QAC6	**Strong Description:** If it presents a clear description of the results obtained with the adoption of the methodology during the teaching of the content, lessons learned, positive or negative points, from the instructor's perspective.	**2**
	Medium Description: If it presents incomplete information about the results and lessons learned by adopting the methodology during teaching programming.	**1**
	Weak Description: If it does not present the results achieved or the lessons learned by adopting the methodology used for teaching programming.	**0**

The process of selection and curation of ALMs was carried out by three researchers. Each step was conducted collaboratively, beginning with individual reading of the studies, followed by group discussions to validate decisions. The objective was to ensure that all analyses were aligned with predefined criteria and that any disagreements were resolved by consensus. This procedure aimed to maintain consistency in the application of the criteria and transparency in the selection of the studies included in the CollabProg repository. Each study was evaluated on a scale from 0 to 2, according to the criteria described in Tabela 4.1. Studies were classified based on their scores and excluded when necessary. Publications that received a score of 0 were removed, even if they were aligned with the research domain. The ALMs selected for inclusion in CollabProg are presented in Table 4.2.

4.1.3 CollabProg - version 1.0

After completing the selection and curation process of ALMs, we structured the collected information and created version 1.0 of CollabProg. This initial version organises the curated content into a digital repository designed to support instructors in the adoption of ALMs. Figure 4.2 presents the interface of CollabProg, using the methodology POGIL as an example. Part

Table 4.2: Selected Active Learning Methodologies for CollabProg Composition

Methodology Name	**Authors**
Blended Learning (BL)	Safana and Nat (2019)
Cooperative Learning (CL)	Pollock and Jochen (2001)
Flipped Classroom (FC)	Kumar et al. (2018)
Game-Based Learning (GBL)	Dicheva and Hodge (2018)
Gamification-Based Learning	Gonçalves et al. (2019)
Method 300 (M300)	de Castro Junior et al. (2021)
Problem-Based Learning (PBL)	dos Santos et al. (2018)
Project-Based Learning (PjBL)	Avouris et al. (2010)
Peer Review (PR)	Turner et al. (2018)
Team Based Learning (TBL)	Joshi et al. (2020)
Topdown (TopD)	Gamage (2021)
Think-Pair-Share (TPS)	Kothiyal et al. (2014)
Coding Dojo (DOJO)	Mayfield et al. (2022)

01 of Figure 4.2 provides an overview of the repository. Part 02 contains a summary of the selected ALM. Part 03 presents detailed content on the methodology, including descriptions of roles, adoption steps, and the internal structure of each step. The details of this part of the design cycle can be seen in Appendix G.

In CollabProg version 1.0, the repository is organised into three labelled menus that provide information to support users in navigating, selecting, and adopting available ALMs. Instructors can access information on ALMs, including adoption examples, tool options adopted by the community, reported experiences, and feedback from other instructors. The platform includes information on both the advantages and limitations identified in the use of different ALMs. A key feature is that no registration is required to access CollabProg; the repository is open to all users. The main interface (*Home*) provides access to the following menus:

- **About**: Provides an overview of the CollabProg repository.
- **Methodology**: Lists the ALMs mapped from the SMS results.
- **Recommendation**: Allows instructors to input characteristics about their class, the content to be taught, discipline, etc., so CollabProg can recommend the most suitable ALM for the scenario. This Recommendation provides step-by-step instructions for using the ALM, information on roles during methodology implementation, activity suggestions, and available community-adopted tool support options.
- **Register methodology**: Invites instructors to actively contribute to the CollabProg repository by sharing a new ALM or an adaptation of one already implemented or tested for teaching programming. This collaborative space is designed to foster a sense of community and shared learning among instructors and researchers.
- **Contact**: Serves as a means of communication between the researchers involved in platform development and the academic community. Users can get in touch via the authors' e-mails to report errors, problems, or suggestions for the repository.

We implemented CollabProg using three main components: back-end, front-end, and the recommendation system, with the participation of six students dedicated to the development process. The back-end manages the business logic and data, providing an API to support front-end

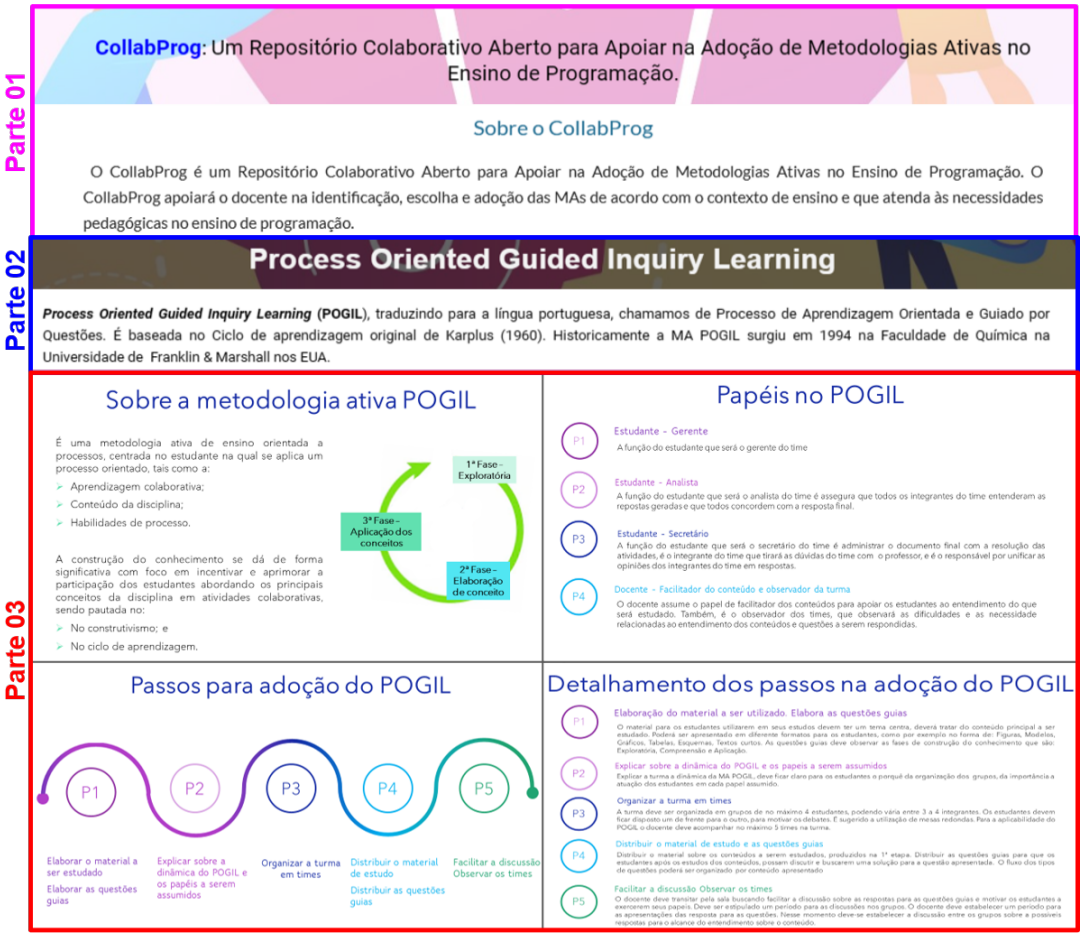


Figure 4.2: Version 1.0 of CollabProg.

interactions. It handles the registration and retrieval of information used to populate the CollabProg interface and generate methodology recommendations. It is important to note that part of the recommendation functionality remains under development and has not yet been fully implemented. The selection of technologies, tools, and programming languages prioritised options that support development, task management, and completion of project stages.

4.1.4 Evaluating CollabProg 1.0

In the Design Cycle, it is essential that stakeholders directly related to the context in which the problem is embedded evaluate the artefact (Wieringa, 2014). Accordingly, an exploratory study was conducted to assess the feasibility of use and the acceptance of CollabProg by instructors. The objective was to evaluate both aspects from the instructors' perspective. Participants were recruited through convenience sampling, involving instructors from different regions of the country. Due to geographical distance, the study artefacts were adapted for remote application.The details of this part of the design cycle can be seen in Appendix H.

The study artefacts were prepared using online tools available through Google Workspace, in-

cluding: (i) a consent form ensuring data confidentiality and participant anonymity (Ethics Committee Approval No. 4.694.031); (ii) a characterisation questionnaire to capture instructors' teaching experience and familiarity with ALMs; (iii) documents outlining the study protocol, instructions for using CollabProg, and online rooms to support the procedures; (iv) the initial version of the CollabProg web portal; (v) a lesson plan template; and (vi) a post-use questionnaire based on the TAM indicators.

Table 4.3 presents the statements answered by the instructors, structured according to the TAM dimensions: Perceived Usefulness (PU), Perceived Ease of Use (PEU), and Perceived Intention to Use (PIU). Two open-ended questions (OQ) were also included to gather more detailed insights into instructors' perceptions. The responses were analysed qualitatively using coding techniques. Full details of the planning and execution of the study evaluating CollabProg version 1.0 are available in Calderon et al. (2023b) and Calderon et al. (2024c).

Table 4.3: Instructor questions

Perceived Usefulness	
PU1	Using the CollabProg repository improved my performance in lesson planning by adopting ALMs.
PU2	Using the CollabProg repository improved my productivity in adopting ALMs.
PU3	Using the CollabProg repository allowed me to fully report the aspects of my experience in adopting active methodologies (ALMs).
PU4	I find the CollabProg repository useful for reporting my experience in adopting active methodologies (ALMs).
Perceived Ease of Use	
PEU1	The CollabProg repository was clear and easy to understand
PEU2	Using the CollabProg repository did not require much mental effort
PEU3	I think the CollabProg repository is easy to use.
PEU4	I find it easy to report my experience of adopting MAs using the CollabProg repository.
Perceived Intention to Use	
PIU1	Assuming I have access to the ColabProg repository, I intend to use it to apply AIs in programming education.
PIU2	Given that I have access to the ColabProg repository, I foresee using it to support me in adopting AIs in programming education.
PIU3	I intend to use the ColabProg repository to assess my experience with adopting an AI in the next month.
Open-Ended Questions	
OQ1	What were the main challenges/negative points perceived by you when using ColabProg?
OQ2	What were the main positive aspects you noticed when using ColabProg?

Figure 4.3 presents the overall results of the participants' perceptions of CollabProg, based on the TAM statements shown in Table 4.3, with the aim of understanding instructors' experience regarding its usefulness, ease of use, and intention to use the repository. Concerning Perceived Usefulness, all instructors fully agreed with the statements (PU1, PU2, PU3, PU4), indicating that CollabProg is useful for planning programming classes involving the adoption of ALMs. In addition, CollabProg supports or enhances instructors' productivity in their practice. It operates as a support tool that enables instructors to draw on their own experiences when selecting an ALM for use in their classes. The results indicate that instructors accept CollabProg as a tool to support the adoption of ALMs in programming education.

Concerning the Perceived Ease of Use of CollabProg, the three statements (PEU1, PEU2, and PEU3) received full agreement from most instructors. Participants reported that describing their experiences with ALM adoption using CollabProg was straightforward. They also indicated that the platform required little mental effort and was easy to understand and operate, particularly in relation to the routine demands of programming teaching. In general, instructors considered CollabProg clear, simple, and accessible. The only exception was statement PEU2, for which instructor D2 expressed partial agreement.

Finally, concerning the Perceived Intention to Use CollabProg, all instructors partially agreed

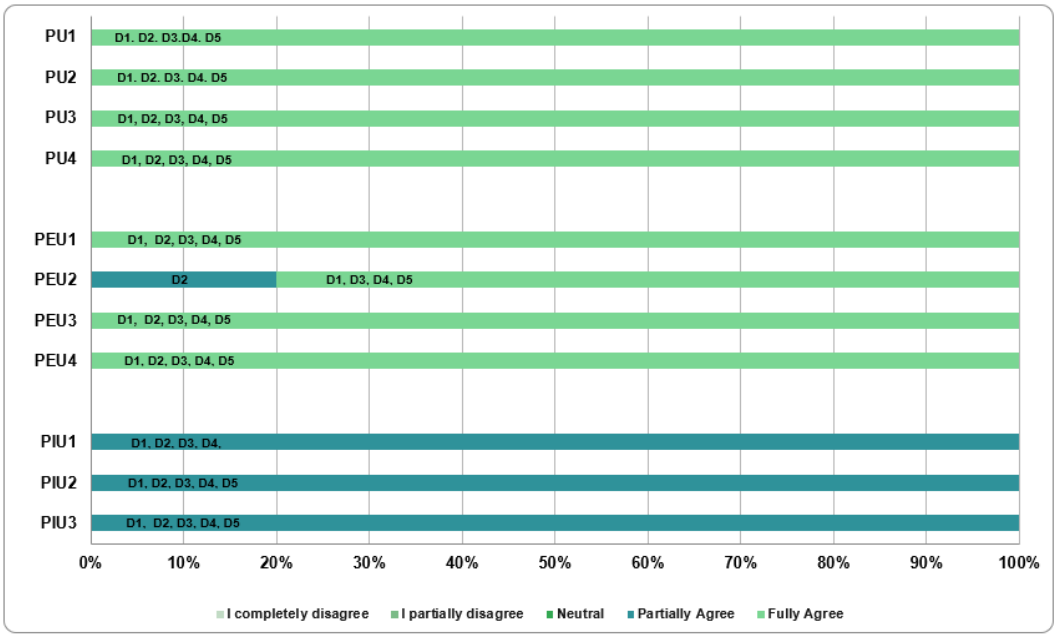


Figure 4.3: General results of perceptions about CollabProg.

with the three statements (PIU1, PIU2, and PIU3). The intention to use CollabProg is relevant to assess the community's interest and willingness to adopt the tool, as well as its acceptance as support for programming teaching. In this regard, instructors evaluated the repository positively and indicated an intention to use it.

Therefore, based on the participants' perceptions, the evaluation of the version 1.0 of CollabProg highlights its potential as a support tool for the adoption of ALMs in programming education. The results indicate acceptance in terms of usefulness, ease of use, and future use intention, although with varying degrees of agreement. The findings suggest that CollabProg can be considered a viable repository to support instructors in selecting and planning ALMs, contributing to the systematisation of pedagogical practices and the strengthening of a collaborative culture in programming education.

However, to enhance its effectiveness and better meet user needs, the next iteration of CollabProg must address certain design and behaviour criteria. Given these results, for version 2.0 it is necessary to improve the design criteria, including: a) a greater variety of ALMs, with detailed descriptions, application examples, and usage contexts; b) clear and structured guidelines for implementing each ALM, covering planning, execution, and evaluation; and c) curation of ALMs, with critical analyses and evidence-based recommendations. Regarding the behaviour criteria, improvements should ensure that the artefact: a) supports the user in understanding ALMs; b) presents clear and detailed information about the methodologies; and c) provides sufficiently detailed guidelines to facilitate implementation. These points guide the improvements proposed in the second design cycle of CollabProg.

4.2 Second Design Cycle: Improvements and Enhancements

After the initial evaluation of CollabProg by volunteer instructors from various higher education institutions, we initiated a second design cycle to refine the tool based on the feedback received. In this subsection, we present the planned improvements for version 2.0, focusing on usability, functionality, and the quality of recommendations to better support the adoption of ALMs in programming education. The evaluation results were analysed according to the design criteria established for the implementation of CollabProg, presented in subsection 3.2.2, which reflect user expectations. Details of version 2.0 are available in Appendices I and J.

We observed that requirement **DC1**, which states that the artefact should provide a variety of ALMs with detailed descriptions, application examples, and usage contexts, was not fully met during the study. This limitation was identified through instructor feedback, indicating the need for a broader selection of active methodologies within the system. To address this, we planned a more comprehensive curation that included methodologies used in combination with various approaches, according to the Quality Assessment Criteria presented in subsection 4.1.2, as well as those developed and implemented by authors of primary studies. Additionally, through the conduction of the MSL study in Calderon et al. (2023a), thirty-seven different ALMs adopted by instructors were identified. Improvements in the detailing of each ALM contributed to enhancing CollabProg in version 2.0.

In version 2.0 of CollabProg, to meet DC2, which requires the artefact to provide clear and structured guidelines for implementing each ALM, details on planning, execution, and evaluation were added to support instructors in adopting the available ALMs. CollabProg now offers specific instructions for applying the selected ALM in the classroom. Figure 4.4 presents general information, such as the time required to apply the methodology and the recommended class size, as well as specific objectives that help instructors decide based on their current context.

Regarding DC3, which states that the artefact must provide curation of ALMs, a process of carefully selecting and organising the content, including critical analyses, evidence-based recommendations, and feedback from instructors who have already applied these methodologies, version 2.0 of CollabProg maintained and enhanced the implementation of these curation criteria, as presented in Subsection 4.1.2. Additionally, in the Methodologies menu, instructors can access the View Feedback button (Figure 4.5). The instructor can see evaluations from other instructors about some specific methodology. Evaluations can be shared through star ratings and comments, providing insights into the implementation experience of the methodology in different contexts.

We believe the View Feedback feature is essential for the academic community. It promotes transparency and trust by allowing users to access evaluations and testimonials from other instructors about the implemented methodologies. This feature facilitates sharing experiences, offering valuable lessons learned and best practices that can benefit new users. By enabling more informed choices and inspiring contextual adaptations, the View Feedback feature strengthens collaboration and community engagement, creating an environment of mutual support and continuous learning.

The evaluation of CollabProg from the perspective of Behavior criteria, which refer to the artifact's contribution to programming teaching practices, revealed several areas of opportunity for improvement. Instructors suggested improving the repository regarding **BC1**, which requires that the artifact supports the user in understanding active methodologies. They highlighted the need to make explanations of the steps and concepts of the methodologies clearer and more

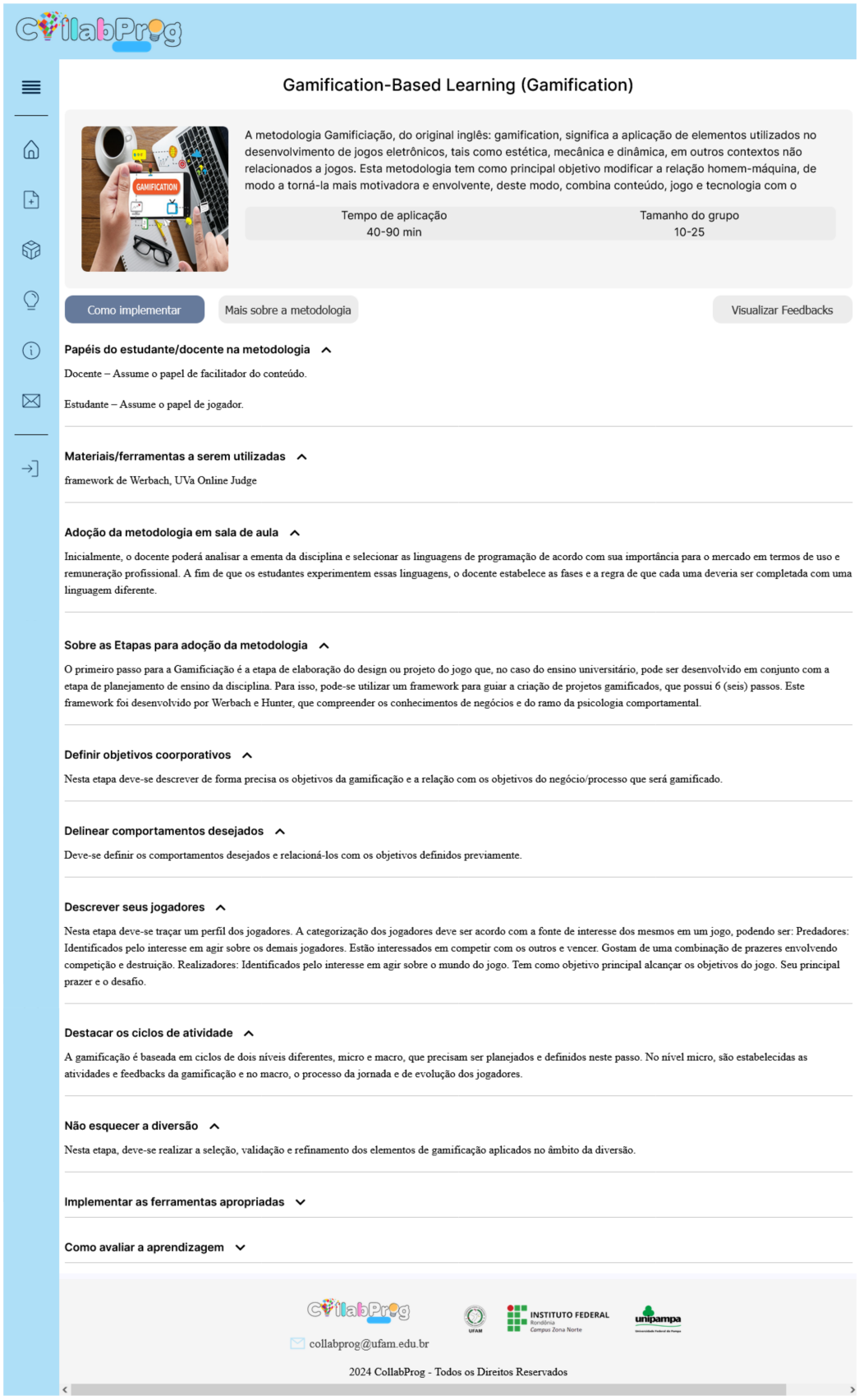


Figure 4.4: Methodology Details

Figure 4.5: View feedback button *feedback*

accessible, facilitating understanding by users. Implementing detailed tutorials and practical examples can help better meet this requirement as for **BC2**, which aims to motivate instructors to adopt Active Methodologies, CollabProg was well evaluated. Professor D3 mentioned that the intuitive presentation of the methodologies in CollabProg facilitated understanding of their operation and lesson planning compared to other sources of documentation. This positive feedback indicates that CollabProg is effectively promoting the adoption of ALMs.

Regarding **BC3**, which requires the presentation of clear and detailed information about ALMs, we realized the need for improvements in documentation and provided examples identified. Participants highlighted the importance of more detailed explanations about the assignment of roles in ALMs, aiming to avoid confusion and facilitate implementation. Improving documentation with specific cases and step-by-step descriptions can help better meet this requirement.

Finally, regarding **BC4**, which requires that guidelines be detailed to facilitate their implementation, CollabProg partially met this requirement. Assessments by instructors D1 and D3 indicated that, although CollabProg facilitated the application of methodologies and improved understanding of the available ALMs, there is still room to make guidelines more detailed and practical. Including checklists, flowcharts, and additional visual resources can make guidelines more effective.

Based on the evaluation of CollabProg and the instructors' suggestions, the repository represents a valuable tool for supporting programming teaching with ALMs. CollabProg received praise for its ability to motivate the adoption of ALMs and facilitate understanding of their operation. Opportunities for improvement were identified regarding the clarity and simplicity of explanations, detailed documentation of methodologies, and explanation of the roles assigned in each. Besides, the positive feedback on the ease of implementing AMs demonstrates the potential of CollabProg as a valuable and effective tool for instructors wishing to use active approaches in programming teaching.

Figure 4.6 shows the CollabProg homepage of version 2.0 (in Portuguese). The labeled menus and their respective icons are displayed on the left side (part 1 of Figure 4.6). This combination aims to provide a cleaner, more intuitive, and aesthetically pleasing interface, enhancing the

user experience. The "Home" menu directs to the CollabProg homepage. Part 2 of Figure 4.6 presents information about the ALMs and details about the CollabProg itself.

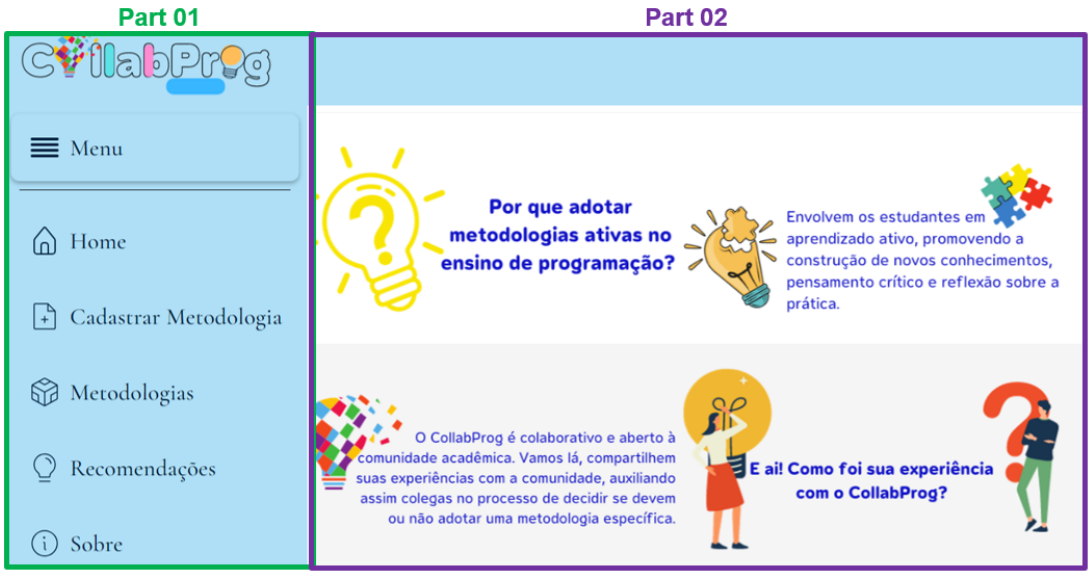


Figure 4.6: CollabProg 2.0 Homepage.

Figure 4.7 shows the **Register Methodology** menu, which directs the user to the methodology registration page. On this page, the instructor is invited to provide details of the ALM, such as the methodology description, educational objectives, implementation steps, suggested activities, and necessary resources. Each field is accompanied by a detailed explanation of how to fill it out, ensuring that the registration is done correctly and that the information provided is helpful for the community that will use the methodology.

For the registration of methodology details, the user will be guided through five pages, each with specific fields for collecting this information. Figure 4.7 presents this functionality's first and second pages. This feature enriches the tool by expanding the diversity of AMs available to other instructors. To register a methodology, the user must fill in mandatory fields, such as methodology description, taught disciplines, related content or categories, programming languages used, necessary materials, methodology principles, methodology planning, and steps for adopting the methodology. These pieces of information are essential for the community to use and follow the tried-and-tested step-by-step process.

Figure 4.8 shows the **Methodologies** menu. On this page, CollabProg provides information about each methodology, including the step-by-step implementation process, the roles of students and instructors, the necessary materials and tools, the average time for lesson planning, the steps for adopting the methodology, and how to assess learning, among other information. These guidelines help instructors understand how to implement the ALM in their classroom.

After accessing the "Methodologies" menu, the instructors are presented with a list of available methodologies for implementation. Upon selecting the one of interest, they are directed to the initial screen for the chosen methodology. On this screen, depicted in Figure 4.4, general information is displayed, such as the time required for applying the methodology and the recommended class size. Besides, we provide specific methodology objectives, which help the

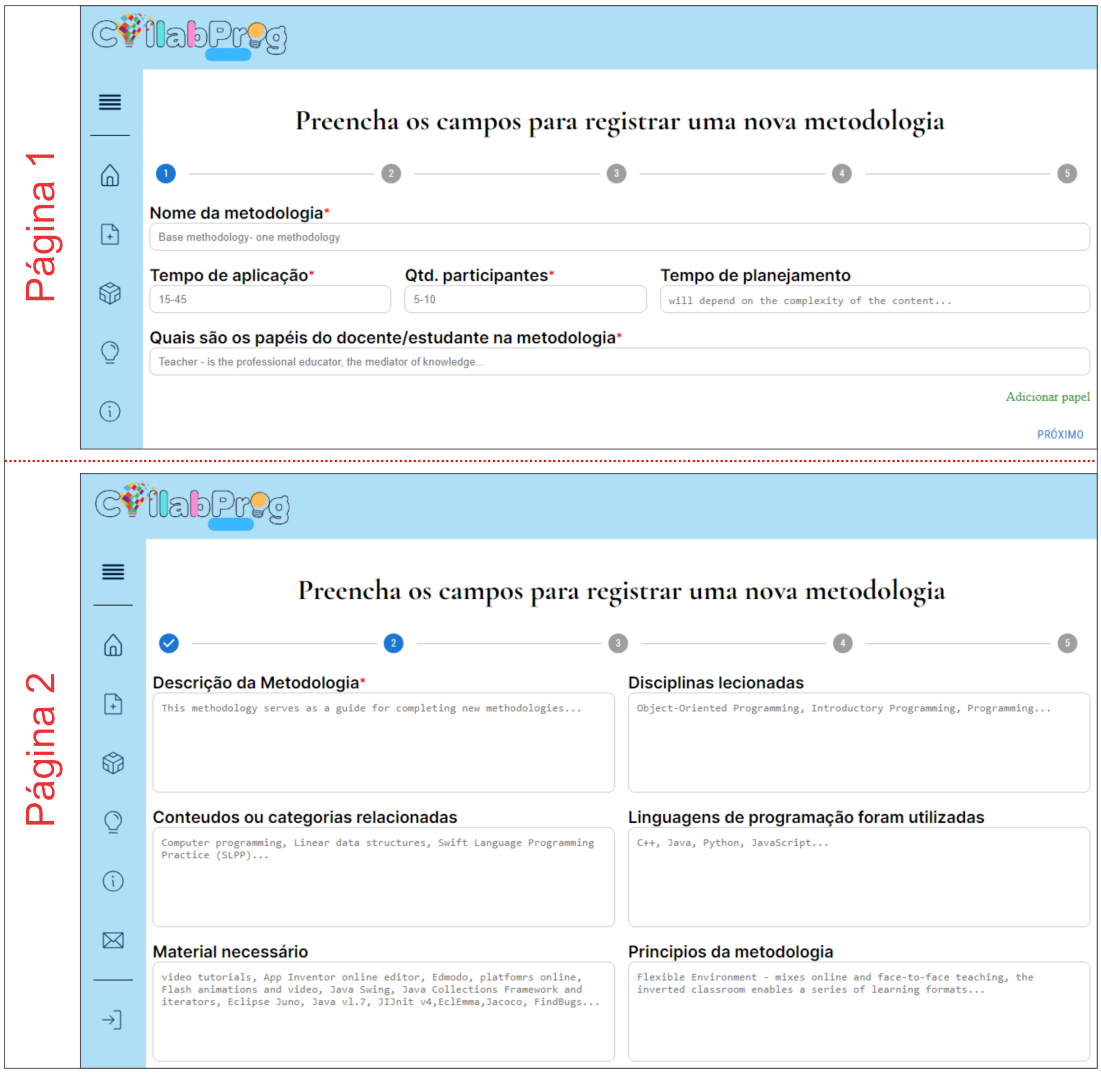


Figure 4.7: Methodology Registration Page.

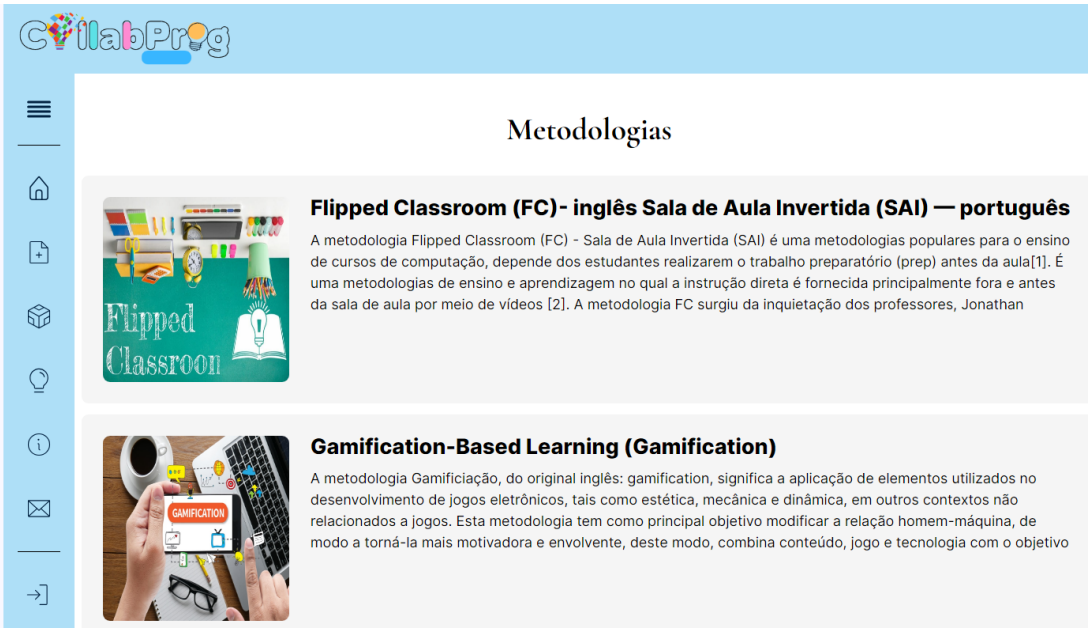


Figure 4.8: Active Learning Methodologies Page.

instructors decide, considering their current context. The current version is accessible online[2].

4.3 Chapter Conclusion

This chapter provided an overview of the conception, evaluation, and improvement of CollabProg. The primary aim was to address practical challenges faced by instructors when implementing ALMs in programming education. CollabProg was developed as a collaborative and open repository offering specific guidelines for adopting various ALMs, supporting educators in enhancing their teaching practices.

The first design cycle focused on organising and curating knowledge about ALMs, ensuring the content was scientifically grounded and relevant to programming education. This curation involved a thorough review of existing studies to include only validated methodologies. This step was essential to provide instructors with reliable resources and assist them in navigating the complexities of integrating ALMs in the classroom.

CollabProg version 1.0 was then developed, featuring a structured platform that highlighted specific methodologies, such as POGIL. The initial version was evaluated through an exploratory study, gathering instructor feedback to assess the platform's feasibility and acceptability. Results showed a positive reception, with instructors evaluating the system according to established criteria, including perceived usefulness and ease of use.

In the second design cycle, instructor feedback identified areas for improvement, notably the need for a broader range of ALMs. This prompted plans for a more comprehensive curation of methodologies. The ongoing evaluation of CollabProg ensures its evolution aligns with user

[2] `https://colabprog.ufam.edu.br/`

needs, improving its effectiveness in supporting instructors adopting ALMs in programming education.

Thus, CollabProg has demonstrated value as a tool offering instructors a structured and accessible platform to improve their teaching practices. Future iterations will continue refining the repository, incorporating additional methodologies and addressing user feedback to ensure the platform's continuous relevance and improvement in programming education.

Chapter 5

Rigor Cycle

This chapter discusses the research rigor adopted in the study and highlights its main contributions, providing the reader with an overview of the methodological robustness and the value added by the findings.

5.1 Research rigor

The rigor of this research in developing CollabProg was an essential aspect. It is associated with credibility, reliability, precision, and integrity, requiring theoretical and methodological consistency (Wieringa, 2014). This rigor was necessary in the creation, evaluation, and evolution of CollabProg. It helped avoid excessive formalism that could hinder development and assessment, while ensuring the relevance of the study. Solid theoretical foundations and existing technical knowledge supported this process.

Rigor is ensured when researchers follow a validated research method, preferably recognised and accepted by the academic community. As noted by Hevner and Chatterjee (2010) and Wieringa (2009), DSR is not limited to applying knowledge to develop an artefact; it must also contribute to the knowledge base of the domain. In this study, the development of CollabProg sought both to address practical challenges in teaching programming and to generate theoretical and methodological contributions to computing education using ALMs.

Thus, research methods were employed to document the steps taken during the DSR cycles and ensure the required rigor. Notable among these are the SMS, presented in Section 3.2, and the experimental study based on the TAM, detailed in Section 4.1.4, which is widely used in technology acceptance research due to its theoretical robustness and broad applicability.

5.2 Contributions

An important stage in the rigor cycle is reporting the main contributions of the research. To ensure the required rigor, research methods were employed to document the steps carried out during the DSR cycles. Notable among these are the SMS, survey, and experimental studies using the TAM model, which is frequently applied in research on technology acceptance and adoption due to its theoretical robustness and applicability across contexts. The main contributions of this research to date include:

- **Identification, classification, and analysis of evidence:** Catalogue and analysis of the types of ALMs applied in programming education in Computer Science.
- **Analysis of instructors' perceptions:** Investigation of instructors' perceptions and challenges in adopting ALMs in programming disciplines.
- **Application of Design Science Research:** Use of the DSR method to develop, evaluate, and evolve the open collaborative repository CollabProg, with potential to guide other researchers and instructors in the field.
- **Exploratory study:** Assessment of the feasibility and acceptance of CollabProg from instructors' perspective, offering practical insights for its use.
- **Support for instructors:** Evidence that CollabProg supported instructors from five educational institutions in adopting ALMs in programming education.
- **Evidence of ALMs' effectiveness:** Verification of ALMs' effectiveness in the literature and demonstration of their practical application in the classroom, resulting in measurable improvement in teaching programming disciplines.

These contributions provide a foundation for the continuous evolution of CollabProg and the improvement of pedagogical practices in programming education within Computer Science.

5.2.1 Publications

In this subsection, we present the publications in the proceedings of SBC events and journals that report the research results conducted for the development of CollabProg. These publications provide a solid foundation for the continuous evolution of the repository and the improvement of pedagogical practices in programming education within the field of Computer Science. The references are listed below in chronological order.

- **Paper**[1] - **RIBEIRO, Maria Ivanilse Calderon**; PASSOS, Odette Mestrinho. A Study on the active methodologies applied to teaching and learning process in the computing area. Ieee Access, v. 8, p. 219083-219097, 2020.
- **Paper**[2] - **RIBEIRO, Maria Ivanilse Calderon**; SILVA, Williamson; FEITOSA, Eduardo Luzeiro. Repositório colaborativo para apoiar a adoção de metodologias ativas no ensino de programação. In: Simpósio Brasileiro de Educação em Computação (EDUCOMP). SBC, 2021. p. 56-57.
- **Paper**[3] - **CALDERON, Ivanilse**; SILVA, Williamson; FEITOSA, Eduardo. Um Mapeamento Sistemático da Literatura sobre o uso de Metodologias Ativas durante o Ensino de Programação no Brasil. Simpósio Brasileiro de Informática na Educação (SBIE), p. 1152-1161, 2021.
- **Paper**[4] - **CALDERON, Ivanilse**; SILVA, Williamson; FEITOSA, Eduardo. CollabProg: Um Repositório Colaborativo Aberto para Apoiar na Adoção de Metodologias Ativas no Ensino de Programação. In: Simpósio Brasileiro de Educação em Computação (EDUCOMP). SBC, 2022. p. 36-39.

[1] https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=9252881
[2] https://sol.sbc.org.br/index.php/educomp_estendido/article/view/14874
[3] https://sol.sbc.org.br/index.php/sbie/article/view/18138
[4] https://sol.sbc.org.br/index.php/educomp_estendido/article/view/19411

- **Paper**[5] - **CALDERON, Ivanilse** et al. Percepção docente sobre o uso do WhatsApp como ferramenta de comunicação no ensino remoto emergencial. In: Workshop sobre Aspectos Sociais, Humanos e Econômicos de Software (WASHES). SBC, 2023. p. 31-40.
- **Paper**[6] - **CALDERON, Ivanilse**; SILVA, Williamson; FEITOSA, Eduardo. Explorando a aceitação do collabprog como um facilitador de metodologias ativas no ensino de programação. In: Simpósio Brasileiro de Informática na Educação (SBIE). SBC, 2023. p. 93-104.
- **Paper**[7] - **CALDERON, Ivanilse** et al. CollabProg: Um Repositório Colaborativo Aberto para Apoiar na Adoçao de Metodologias Ativas no Ensino de Programaçao. In: Congresso Brasileiro de Informática na Educação (CBIE). SBC, 2023. p. 189-192.
- **Paper**[8] - **CALDERON, Ivanilse**; SILVA, Williamson; FEITOSA, Eduardo. Active learning methodologies for teaching programming in undergraduate courses: A systematic mapping study. Informatics in Education, v. 23, n. 2, p. 279-322, 2024.
- **Paper**[9] - **CALDERON, Ivanilse**; SILVA, Williamson; FEITOSA, Eduardo. Uma Plataforma Web para apoiar Docentes no Ensino de Programação em Cursos de Sistemas de Informação. In: Simpósio Brasileiro de Sistemas de Informação (SBSI). SBC, 2024. p. 297-302.
- **Paper**[10] - **CALDERON, Ivanilse** et al. Um Survey sobre o Uso de Metodologias Ativas no Ensino de Programação em Universidades Brasileiras. In: Simpósio Brasileiro de Informática na Educação (SBIE). SBC, 2024. p. 2163-2177.
- **Paper**[11] - **CALDERON, Ivanilse**; SILVA, Williamson; FEITOSA, Eduardo. Building Bridges Instead of Putting up Walls: an Educational Tool to Facilitate Instructors in Adopting Active Learning Methodologies for Teaching Programming. IEEE Access, 2025.
- **Paper**[12] **CALDERON, Ivanilse**; SILVA, Williamson; FEITOSA, Eduardo. Investigating the Use of Active Learning Methodologies in Programming Education: Findings from a Brazilian National Survey. IEEE RITA, 2025 (In final adjustments following the journal's review).

Given this context, we present a series of articles that comprise our research. Article 1 covers our initial study on the investigated problem, while Article 2 addresses the preliminary concepts of the proposed artifact. Article 3 examines the adoption of methodologies within the national context. Article 4 introduces CollabProg to the scientific community as the artifact developed from these studies. Article 5 compiles and updates the findings from Articles 1 and 3. Article 6 explores the acceptance of CollabProg through an empirical study. Finally, Article 7 presents CollabProg as a web platform designed to support instructors in information systems courses.

Together, these articles provide a comprehensive view of CollabProg's development and application. They illustrate a continuous research and development process focused on enhancing teaching practices in programming education. CollabProg stands as a significant contribution

[5] https://sol.sbc.org.br/index.php/washes/article/view/24773
[6] https://sol.sbc.org.br/index.php/sbie/article/view/26653
[7] https://sol.sbc.org.br/index.php/cbie_estendido/article/view/27511
[8] https://www.ceeol.com/search/article-detail?id=1252141
[9] https://sol.sbc.org.br/index.php/sbsi_estendido/article/view/28630
[10] https://sol.sbc.org.br/index.php/sbie/article/view/31386
[11] https://ieeexplore.ieee.org/abstract/document/10938091
[12] em revisão para submissão final para IEEE RITA, 2025.

to the educational domain, offering a collaborative and open environment that supports the adoption of active learning methodologies and prepares students for the evolving demands of the computing industry.

Chapter Conclusion

This chapter presented the Rigor Cycle of the research, highlighting the theoretical and methodological foundations that supported the development, evaluation, and refinement of CollabProg. Established research methods such as a SMS, a national survey, and experimental studies based on the TAM model were employed to ensure scientific rigor, contributing to the reliability and credibility of the findings.

The chapter also outlined the main contributions of the research, including the identification and classification of evidence on ALMs in programming education, as well as a deeper understanding of the challenges faced by instructors when implementing these methodologies. The use of DSR guided the iterative development of CollabProg, resulting in theoretical advancements and practical benefits for the educational community.

In summary, the results provide a solid basis for future work, demonstrating the relevance and impact of the research. The contributions reinforce the potential of CollabProg to support instructors in adopting effective pedagogical strategies and offer insights for the continuous improvement of programming education in Computer Science.

Chapter 6

Final Considerations

This chapter presents the main conclusions of the research, outlines directions for future work, discusses the implications of the findings, and addresses potential threats to the validity of the study.

6.1 Conclusions and Future Perspectives

ALMs have gained increasing recognition in programming education as effective approaches to engage students and enhance learning outcomes. Adoption of ALMs in computer science courses is growing, yet instructors often face challenges that hinder their implementation. Inspired by DSR, this study aimed to support instructors in adopting ALMs for programming education. The application of DSR allowed for the clear definition of the research problem and guided the development, evaluation, and refinement of a supporting artifact. To better understand the landscape of ALMs in programming teaching, we initially conducted a SMS that identified 37 distinct ALMs currently employed by instructors. Moreover, the study revealed 17 publications discussing the combined use of multiple ALMs and four that proposed new methodologies, reflecting diverse strategies developed by instructors to promote ALMs in programming classes.

After completing the SMS, the curation process of the ALMs to be integrated into CollabProg commenced, focusing exclusively on content and tool support options documented in the literature for instructors' use. This approach aims to prevent user frustration by presenting only knowledge and materials backed by scientific evidence, experimentation, or demonstrated relevance. The selection process was rigorous, prioritising methodologies supported by solid evidence and excluding those lacking empirical validation or with theoretical foundations considered irrelevant to the community in this research context. Based on the SMS and suvey results and the curation of primary studies within the DSR Design Cycle, we developed, evaluated, and refined the artifact CollabProg.

CollabProg is a collaborative and open repository created to assist instructors in selecting the most suitable ALMs for their specific teaching contexts in programming education. An experimental study was conducted involving five higher education institutions in Brazil to evaluate the feasibility and acceptance of CollabProg from the perspective of instructors. This study highlights the importance of developing strategies to support instructors in programming education and to motivate students, which is a key element for effective teaching. This aspect

is particularly significant in collaborative learning environments, where social interaction plays a crucial role in the adoption of ALMs (Serrano-Cámara et al., 2014).

6.2 Research Implications

This research presents several implications with the potential to influence teaching practices, computing education research, and student development. CollabProg was developed to support instructors in programming education by providing a portal with diverse guidelines for adopting ALMs. As a collaborative repository, it offers access to tool recommendations and guidance on student assessment aligned with ALMs.

Regarding the preparation of supplementary materials, instructors can develop additional resources to complement those available in the repository, thereby providing students with clearer guidance and contextualising learning within the curriculum. Collaboration among instructors is encouraged, allowing the exchange of effective teaching practices and strategies for integrating repository resources across different programming disciplines and educational contexts. The use of CollabProg requires instructors to adapt their teaching methods and commit to ongoing professional development to maximise its benefits.

For students, the implications involve enhanced active and practical learning experiences. Supported by CollabProg, instructors can facilitate hands-on programming activities and projects that reinforce theoretical concepts and develop practical problem-solving skills. Collaborative activities such as group projects, pair programming, and class discussions foster knowledge sharing and social skill development. The repository also aids instructors in planning and implementing formative assessments, such as quizzes, code reviews, and discussions, that help identify learning difficulties and enable timely instructional adjustments.

Finally, for researchers, CollabProg highlights gaps in computing education, particularly regarding the integration of ALMs and new technologies in teaching and assessment. This opens opportunities to develop and evaluate novel methodologies promoting personalised and adaptive learning approaches. Researchers can also investigate best practices, challenges, and lessons learned in adopting ALMs, contributing to improved pedagogical frameworks and informing educational policies in computing.

6.3 Threats to the Validity of the Research

Despite precautions taken in defining the SMS protocol during the Rigor cycle, as per Kitchenham (2012), and the careful design of the survey aimed at understanding instructors' perceptions of adopting and using ALMs in programming classes, using the Opinion Survey method with an online questionnaire, targeting higher education instructors experienced with ALMs , this work presents some limitations and threats to validity. The following section discusses these threats and the measures adopted to mitigate their impact on the development of CollabProg.

a) Classification of SMS results: The classification of SMS results, which formed the knowledge base and mapped potential ALMs for CollabProg, has limitations due to the subjective nature of human classification. Although this manual process is common in the field, it introduces potential bias. To reduce this, three researchers conducted the classification, and two doctoral researchers reviewed the protocol, inclusion and exclusion criteria, and research strategy. To assess reliability, two researchers independently classified a random sample of 40 publications,

resulting in a Kappa agreement of 0.89, indicating almost perfect agreement. Based on this, the subsequent steps of selection and data extraction proceeded.

b) Scope of conducted studies: The studies carried out do not represent the entirety of research on ALM adoption by instructors in programming education. This limitation was mitigated by an iterative research strategy including pilot testing and clear participant selection criteria. Systematic methods were applied and documented to allow replication. However, the sample size limits the generalizability of the findings. Future research will involve larger samples to enhance representativeness among educators.

c) Data analysis: Data analysis performed by a single professor may introduce subjective bias in interpreting results, potentially affecting objectivity and comprehensiveness. To mitigate this, plans include involving additional researchers or professionals in data analysis to reduce personal bias and improve interpretation accuracy.

d) Regarding the survey: Limitations include the geographic concentration of participants in northern Brazil, which may affect the generalisability of the findings due to regional variations in educational practices. Furthermore, the predominance of faculty from public institutions restricts a broader perspective, as private institutions may adopt different teaching strategies. The self-reported nature of the data introduces potential bias, with educators possibly overestimating or underestimating their practices. Another important limitation is the lack of an in-depth analysis of the impact of ALMs on student learning outcomes, indicating the need for further research on the development of specific skills such as problem-solving, collaborative work, and practical application of theoretical knowledge.

6.4 Future Works

Future work will focus on formulating and evolving a model of difficulties related to adopting ALMs in programming education, aimed at evaluating and validating CollabProg. This model will be developed from the results of the experimental study and will represent the perspectives and experiences of instructors teaching programming disciplines. A survey will be conducted to evaluate the model, and its evolution will be guided by instructors' feedback. Validation will also occur through practical use of the model by instructors, ensuring that CollabProg is assessed across diverse experiences, needs, and educational contexts.

In addition to refining the systematic mapping study of ALMs based on literature, the set of methodologies available on CollabProg will be continuously updated to align the platform with pedagogical trends and the evolving requirements of instructors. This maintenance is essential to keep the repository relevant and useful for programming education.

Efforts will also focus on engaging the teaching community, particularly those who currently use traditional methods. Through tutorials and awareness resources, the aim is to demonstrate the benefits of ALMs and support their adoption by instructors. Additionally, the platform will be translated into other languages to reach a broader international audience. This internationalisation is expected to expand CollabProg's impact by fostering a global network of educators sharing best practices and contributing to the innovation of programming education worldwide.

In conclusion, CollabProg is intended to serve as technological support that consolidates, in a single online portal, strategies for adopting various ALMs in programming education. The platform will provide examples, activity suggestions, support options, tools adopted by the community, reports on experiences in different scenarios, results achieved by instructors, and

critical reflections on the advantages and limitations of the adopted methodologies.

6.5 Chapter Conclusion

This chapter presented the final considerations of the research, emphasising its contributions, implications, and limitations. By adopting the DSR methodology, the study systematically addressed the challenge of supporting instructors in adopting ALMs in programming education. From conducting a SMS to developing and evaluating the CollabProg artifact, each phase contributed to a solid understanding of the barriers faced by instructors in this context. The research identified a wide variety of ALMs applied in programming education and underscored the need to provide structured, evidence-based, and accessible support for instructors. CollabProg emerged as a collaborative and open repository with potential to transform teaching practices by offering curated methodologies, tool recommendations, assessment guidelines, and opportunities for peer collaboration. Its experimental implementation across multiple higher education institutions in Brazil demonstrated the platform's feasibility and acceptance, highlighting its relevance and potential positive impact on both instructors and students.

The implications of this work extend to educational practice, academic research, and student engagement. Instructors gain access to resources that promote more dynamic, collaborative, and student-centred approaches. Students benefit from more meaningful, practical learning experiences. Researchers acquire a foundation for exploring further questions and methods related to integrating ALMs in computing education. Despite these contributions, the research recognises its limitations, particularly regarding sample size, potential biases in classification and data analysis, and the generalisability of the findings. Measures were taken to mitigate these threats, and future work is planned to enhance the scope and reliability of the results.

Moving forward, the evolution of CollabProg will involve developing a validated model of the difficulties faced by instructors, expanding and internationalising the platform, continuously updating methodologies, and engaging a broader teaching community. Ultimately, CollabProg is expected to serve as a dynamic and evolving technological support system, enabling instructors in diverse educational contexts to adopt, adapt, and reflect on active learning strategies in programming education.

Bibliography

Ahshan, R. (2021). A framework of implementing strategies for active student engagement in remote/online teaching and learning during the covid-19 pandemic. *Education Sciences*, 11(9):483.

Arık, S. and Yılmaz, M. (2020). The effect of constructivist learning approach and active learning on environmental education: A meta-analysis study. *International Electronic Journal of Environmental Education*, 10(1):44–84.

Astrachan, O. L., Duvall, R. C., Forbes, J., and Rodger, S. H. (2002). Active learning in small to large courses. In *32nd Annual Frontiers in Education*, volume 1, pages T2A–T2A. IEEE.

Avouris, N., Kaxiras, S., Koufopavlou, O., Sgarbas, K., and Stathopoulou, P. (2010). Teaching introduction to computing through a project-based collaborative learning approach. In *2010 14th Panhellenic Conference on Informatics*, pages 237–241. IEEE.

Bacich, L. and Moran, J. (2018). *Metodologias ativas para uma educação inovadora: uma abordagem teórico-prática*. Penso Editora.

Beaubouef, T. and Mason, J. (2005). Why the high attrition rate for computer science students: some thoughts and observations. *ACM SIGCSE Bulletin*, 37(2):103–106.

Bigolin, N. M., Silveira, S. R., Bertolini, C., de Almeida, I. C., Geller, M., Parreira, F. J., da Cunha, G. B., and Macedo, R. T. (2020). Metodologias ativas de aprendizagem: um relato de experiência nas disciplinas de programação e estrutura de dados. *Research, Society and Development*, 9(1):e74911648–e74911648.

Borges, R. P., Oliveira, P. R. F., Lima, R. d. R., and De Lima, R. (2018). A systematic review of literature on methodologies, practices, and tools for programming teaching. *IEEE Latin America Transactions*, 16(5):1468–1475.

Caceffo, R., Gama, G., and Azevedo, R. (2018). Exploring active learning approaches to computer science classes. In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*, pages 922–927.

Calderon, I., Oran, A. C., Feitosa, E., and Silva, W. (2024a). Um survey sobre o uso de metodologias ativas no ensino de programação em universidades brasileiras. In *Simpósio Brasileiro de Informática na Educação (SBIE)*, pages 2163–2177. SBC.

Calderon, I., Silva, W., and Feitosa, E. (2021). Um mapeamento sistemático da literatura sobre o uso de metodologias ativas durante o ensino de programação no brasil. In *Anais do XXXII Simpósio Brasileiro de Informática na Educação*, pages 1152–1161. SBC.

Calderon, I., Silva, W., and Feitosa, E. (2022). Collabprog: Um repositório colaborativo aberto para apoiar na adoção de metodologias ativas no ensino de programação. In *Anais Estendidos do II Simpósio Brasileiro de Educação em Computação*, pages 36–39. SBC.

Calderon, I., Silva, W., and Feitosa, E. (2023a). Active learning methodologies for teaching programming in undergraduate courses: A systematic mapping study. *Informatics in Education*.

Calderon, I., Silva, W., and Feitosa, E. (2023b). Explorando a aceitação do collabprog como um facilitador de metodologias ativas no ensino de programação. In *Anais do XXXIV Simpósio Brasileiro de Informática na Educação*, pages 93–104. SBC.

Calderon, I., Silva, W., and Feitosa, E. (2024b). Active learning methodologies for teaching programming in undergraduate courses: A systematic mapping study. *Informatics in Education*, 23(2):279–322.

Calderon, I., Silva, W., and Feitosa, E. (2024c). Uma plataforma web para apoiar docentes no ensino de programação em cursos de sistemas de informação. In *Simpósio Brasileiro de Sistemas de Informação (SBSI)*, pages 297–302. SBC.

Calderon, I., Silva, W., and Feitosa, E. (2025). Building bridges instead of putting up walls: an educational tool to facilitate instructors in adopting active learning methodologies for teaching programming. *IEEE Access*.

Chang, C.-S., Chung, C.-H., and Chang, J. A. (2020). Influence of problem-based learning games on effective computer programming learning in higher education. *Educational Technology Research and Development*, 68(5):2615–2634.

Coelho, J. A., Souza, G. H., and Albuquerque, J. (2020). Desenvolvimento de questionários e aplicação na pesquisa em informática na educação. *Metodologia de Pesquisa em Informática na Educa\cão: Abordagem Quantitativa de Pesquisa. Porto Alegre: SBC. Série Metodologia de Pesquisa em Informática na Educa\cão*, 2.

Correia, A.-P. (2018). As múltiplas facetas da curadoria de conteúdos digitais. *Revista Docência e Cibercultura*, 2(3):14–32.

Corritore, C. L. and Love, B. (2020). Redesigning an introductory programming course to facilitate effective student learning: A case study. *Journal of Information Technology Education: Innovations in Practice*, 19:091–135.

da Silva, M. A. d. F. and Oliveira, M. (2019). A robótica educacional na perspectiva das metodologias ativas. In *Anais do XXV Workshop de Informática na Escola*, pages 1289–1293. SBC.

Davis, F. D., Bagozzi, R. P., and Warshaw, P. R. (1989). Technology acceptance model. *J Manag Sci*, 35(8):982–1003.

de Almeida, L., Rolim, M. P., da Silva, R., and Costa, A. (2019). E-pbl: Ferramenta de apoio ao aprendizado e uso da metodologia de aprendizado baseado em problemas. In *Anais do XXV Workshop de Informática na Escola*, pages 1399–1403. SBC.

de Castro, R. M. and Siqueira, S. (2019). Alcasystem-um portal com técnicas de aprendizagem ativa para disciplinas da área da computaçao. In *Anais dos Workshops do Congresso Brasileiro de Informática na Educação*, volume 8, page 1243.

de Castro Junior, A. A., Cheung, L. M., Batista, E. J. S., and de Lima, A. C. (2021). Uma análise preliminar da aplicação do método 300 em turmas de algoritmos e programação. In *Anais do XXIX Workshop sobre Educação em Computação*, pages 171–180. SBC.

de Farias, G. F., Brito, N., Farias, F. J. S., and DE SOUZA, M. V. (2018). Moodle como ferramenta de suporte a pbl em rede: Uma revisão sistemática. *Educação Fora da Caixa: Tendências Internacionais e Perspectivas sobre a Inovação na Educação*.

Denny, P., Luxton-Reilly, A., Tempero, E., and Hendrickx, J. (2011). Understanding the syntax barrier for novices. In *Proceedings of the 16th annual joint conference on Innovation and technology in computer science education*, pages 208–212.

Dicheva, D. and Hodge, A. (2018). Active learning through game play in a data structures course. In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*, pages 834–839.

Doolittle, P., Wojdak, K., Walters, A., et al. (2023). Defining active learning: A restricted systematic review. *Teaching and Learning Inquiry*, 11.

dos Santos, S. C., Reis, P. B., Reis, J. F., and Tavares, F. (2020). Two decades of pbl in teaching computing: a systematic mapping study. *IEEE transactions on education*, 64(3):233–244.

dos Santos, S. C., Santana, E., Santana, L., Rossi, P., Cardoso, L., Fernandes, U., Carvalho, C., and Torres, P. (2018). Applying pbl in teaching programming: an experience report. In *2018 IEEE Frontiers in Education Conference (FIE)*, pages 1–8. IEEE.

Duffany, J. L. (2017). Application of active learning techniques to the teaching of introductory programming. *IEEE Revista Iberoamericana de Tecnologias del Aprendizaje*, 12(1):62–69.

Eickholt, J. (2018). Barriers to active learning for computer science faculty. *arXiv preprint arXiv:1808.02426*.

Elliott, S. N. (1996). Educational psychology: Effective teaching, effective learning. *(No Title)*.

Eteng, I., Akpotuzor, S., Akinola, S. O., and Agbonlahor, I. (2022). A review on effective approach to teaching computer programming to undergraduates in developing countries. *Scientific African*, 16:e01240.

Feyzi Behnagh, R. and Yasrebi, S. (2020). An examination of constructivist educational technologies: Key affordances and conditions. *British Journal of Educational Technology*, 51(6):1907–1919.

Freire, L., Coutinho, J., Lima, V., and Lima, N. (2019). Uma proposta de encontros de tutoria baseada em metodologias ativas para disciplinas de programação introdutória. In *Anais dos Workshops do Congresso Brasileiro de Informática na Educação*, volume 8, page 298.

Gamage, L. N. (2021). A bottom-up approach for computer programming education. *Journal of Computing Sciences in Colleges*, 36(7):66–75.

Garcia, F. W. D. S., Carvalho, E. D. C., and Oliveira, S. R. B. (2021). Use of active methodologies for the development of a teaching plan for the algorithms subject. In *2021 IEEE Frontiers in Education Conference (FIE)*, pages 1–9. IEEE.

Gonçalves, B., Nascimento, E., Monteiro, E., Portela, C., and Oliveira, S. (2019). Elementos de gamificação aplicados no ensino-aprendizagem de programação web. In *Anais do XXVII Workshop sobre Educação em Computação*, pages 1–10. SBC.

Gonçalves, M., Souza, S. M., Barros, F., and Bittencourt, R. (2017). Percepções sobre metodologias ativas de aprendizagem de programação no ensino profissionalizante. In *Anais dos Workshops do Congresso Brasileiro de Informática na Educação*, volume 6, page 1132.

Hendrik, H. (2019). Flipping web programming class: Student's perception and performance. In *Proceedings of the 11th International Conference on Engineering Education (ICEED)*, pages 31–45.

Hevner, A. and Chatterjee, S. (2010). Design science research in information systems. *Design research in information systems: theory and practice*, pages 9–22.

Hevner, A. R. (2007). A three cycle view of design science research. *Scandinavian journal of information systems*, 19(2):4.

Imbulpitiya, A., Kodagoda, N., Gamage, A., and Suriyawansa, K. (2020). Using active learning integrated with pedagogical aspects to enhance student's learning experience in programming and related concepts. In *International Conference on Interactive Collaborative Learning*, pages 218–228. Springer.

Joshi, A., Schmidt, M., Panter, S., and Jain, A. (2020). Evaluating the benefits of team-based learning in a systems programming class. In *2020 IEEE Frontiers in Education Conference (FIE)*, pages 1–7. IEEE.

Katona, J. and Kovari, A. (2016). A brain–computer interface project applied in computer engineering. *IEEE Transactions on Education*, 59(4):319–326.

Kitchenham, B. A. (2012). Systematic review in software engineering: where we are and where we should be going. In *Proceedings of the 2nd international workshop on Evidential assessment of software technologies*, pages 1–2.

Kong, S.-C., Lai, M., and Sun, D. (2020). Teacher development in computational thinking: Design and learning outcomes of programming concepts, practices and pedagogy. *Computers & Education*, 151:103872.

Kothiyal, A., Murthy, S., and Iyer, S. (2014). Think-pair-share in a large cs1 class: does learning really happen? In *Proceedings of the 2014 conference on Innovation & technology in computer science education*, pages 51–56.

Kumar, M., Renumol, V., and Murthy, S. (2018). Flipped classroom strategy to help under-achievers in java programming. In *2018 International Conference on Learning and Teaching in Computing and Engineering (LaTICE)*, pages 44–49. IEEE.

Liao, Y.-C. and Ringler, M. (2023). Backward design: Integrating active learning into undergraduate computer science courses. *Cogent Education*, 10(1):2204055.

Lima, J., Alencar, F., and Santos, W. (2021). A preliminary guide for assertive selection of active methodologies in software engineering education. In *Brazilian Symposium on Software Engineering*, pages 170–179.

Lima, V. V. (2016). Constructivist spiral: an active learning methodology. *Interface-Comunicação, Saúde, Educação*, 21:421–434.

Luxton-Reilly, A. (2016). Learning to program is easy. In *Proceedings of the 2016 ACM Conference on Innovation and Technology in Computer Science Education*, pages 284–289.

Luxton-Reilly, A., Albluwi, I., Becker, B. A., Giannakos, M., Kumar, A. N., Ott, L., Paterson, J., Scott, M. J., Sheard, J., and Szabo, C. (2018). Introductory programming: a systematic literature review. In *Proceedings Companion of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education*, pages 55–106.

Matsushita, K. (2018). An invitation to deep active learning. *Deep active learning: Toward greater depth in university education*, pages 15–33.

Mayfield, C., Moudgalya, S. K., Yadav, A., Kussmaul, C., and Hu, H. H. (2022). Pogil in cs1: Evidence for student learning and belonging. In *Proceedings of the 53rd ACM Technical Symposium on Computer Science Education V. 1*, pages 439–445.

Moreno, B. (2019). Combinando metodologias ágeis e ativas no ensino de introdução a programação a estudantes do ensino médio. In *Anais do Workshop de Desafios da Computação Aplicada à Educação*, volume 8, pages 45–47.

Moya, E. C. (2017). Using active methodologies: The studentśview. *Procedia-Social and Behavioral Sciences*, 237:672–677.

Nguyen, K. A., Borrego, M., Finelli, C. J., DeMonbrun, M., Crockett, C., Tharayil, S., Shekhar, P., Waters, C., and Rosenberg, R. (2021). Instructor strategies to aid implementation of active learning: a systematic literature review. *International Journal of STEM Education*, 8:1–18.

Okonkwo, C. W. and Ade-Ibijola, A. (2023). Synthesis of nested loop exercises for practice in introductory programming. *Egyptian Informatics Journal*, 24(2):191–203.

O'grady, M. J. (2012). Practical problem-based learning in computing education. *ACM Transactions on Computing Education (TOCE)*, 12(3):1–16.

Paristiowati, M., Rahmawati, Y., Fitriani, E., Satrio, J. A., and Putri Hasibuan, N. A. (2022). Developing preservice chemistry teachers' engagement with sustainability education through an online project-based learning summer course program. *Sustainability*, 14(3):1783.

Parsons, P. (2011). Preparing computer science graduates for the 21st century. *Teaching Innovation Projects*, 1(1).

Penney, J., Pimentel, J. F., Steinmacher, I., and Gerosa, M. A. (2023). Anticipating user needs: Insights from design fiction on conversational agents for computational thinking. In *International Workshop on Chatbot Research and Design*, pages 204–219. Springer.

Pollock, L. and Jochen, M. (2001). Making parallel programming accessible to inexperienced programmers through cooperative learning. *ACM SIGCSE Bulletin*, 33(1):224–228.

Pundak, D. and Rozner, S. (2008). Empowering engineering college staff to adopt active learning methods. *Journal of Science Education and Technology*, 17(2):152–163.

Qian, M. and Clark, K. R. (2016). Game-based learning and 21st century skills: A review of recent research. *Computers in human behavior*, 63:50–58.

Raj, A. G. S., Patel, J., and Halverson, R. (2018). Is more active always better for teaching introductory programming? In *2018 International Conference on Learning and Teaching in Computing and Engineering (LaTICE)*, pages 103–109. IEEE.

Ribeiro, I. C., Silva, W., and Feitosa, E. L. (2021). Repositório colaborativo para apoiar a adoção de metodologias ativas no ensino de programação. In *Anais Estendidos do I Simpósio Brasileiro de Educação em Computação*, pages 56–57. SBC.

Ribeiro, M. I. C. and Passos, O. M. (2020). A study on the active methodologies applied to teaching and learning process in the computing area. *IEEE Access*, 8:219083–219097.

Safana, A. I. and Nat, M. (2019). Students' perception of a blended learning approach in an african higher institution. *J. Univers. Comput. Sci.*, 25(5):515–540.

Sasson, I., Yehuda, I., Miedijensky, S., and Malkinson, N. (2022). Designing new learning environments: An innovative pedagogical perspective. *The Curriculum Journal*, 33(1):61–81.

Selçuk, A. and Yilmaz, M. (2020). The effect of constructivist learning approach and active learning on environmental education: A meta-analysis study. *International Electronic Journal of Environmental Education*, 10(1):44–84.

Serrano-Cámara, L. M., Paredes-Velasco, M., Alcover, C.-M., and Velazquez-Iturbide, J. Á. (2014). An evaluation of students' motivation in computer-supported collaborative learning of programming concepts. *Computers in human behavior*, 31:499–508.

Sharma, V., Bhagat, K. K., Huang, H.-H., and Chen, N.-S. (2022). The design and evaluation of an ar-based serious game to teach programming. *Computers & Graphics*, 103:1–18.

Silva, W., Gadelha, B., Steinmacher, I., and Conte, T. (2020a). Towards an open repository for teaching software modeling applying active learning strategies. In *2020 IEEE/ACM 42nd International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET)*, pages 162–172. IEEE.

Silva, W., Steinmacher, I., and Conte, T. (2019). Students' and instructors' perceptions of five different active learning strategies used to teach software modeling. *IEEE Access*, 7:184063–184077.

Silva, W. A. F. et al. (2020b). Opensmals: um repositório aberto para auxiliar no ensino de modelagem de software empregando estratégias de aprendizagem ativa.

Sobral, S. R. (2021a). Project based learning with peer assessment in an introductory programming course.

Sobral, S. R. (2021b). Strategies on teaching introducing to programming in higher education. In *World Conference on Information Systems and Technologies*, pages 133–150. Springer.

Sobrinho, H., Castro, L., Nogueira, A., Harada, E., and Gadelha, B. (2016). Organizando o conhecimento sobre técnicas de aprendizagem colaborativas. *Nuevas Ideas em Informatica Educativa*, 12:152–156.

Tharayil, S., Borrego, M., Prince, M., Nguyen, K. A., Shekhar, P., Finelli, C. J., and Waters, C. (2018). Strategies to mitigate student resistance to active learning. *International Journal of STEM Education*, 5(1):1–16.

Travers, J. F., Elliott, S. N., and Kratochwill, T. R. (1993). *Educational psychology: Effective teaching, effective learning.* Brown & Benchmark/Wm. C. Brown Publ.

Turner, S. A., Pérez-Quiñones, M. A., and Edwards, S. H. (2018). Peer review in cs2: Conceptual learning and high-level thinking. *ACM Transactions on Computing Education (TOCE)*, 18(3):1–37.

Tutal, Ö. and Yazar, T. (2022). Active learning promotes more positive attitudes towards the course: A meta-analysis. *Review of Education*, 10(1):e3346.

Venter, M. (2020). Gamification in stem programming courses: State of the art. In *2020 ieee global engineering education conference (educon)*, pages 859–866. IEEE.

Wieringa, R. (2009). Design science as nested problem solving. In *Proceedings of the 4th international conference on design science research in information systems and technology*, pages 1–12.

Wieringa, R. J. (2014). Design science methodology for information systems and software engineering.

Witt, D. T., Kemczinski, A., and dos Santos, L. M. (2018). Resolução de problemas: Abordagens aplicadas no ensino de computação. *Anais do Computer on the Beach*, pages 731–740.

Yannier, N., Hudson, S. E., Koedinger, K. R., Hirsh-Pasek, K., Golinkoff, R. M., Munakata, Y., Doebel, S., Schwartz, D. L., Deslauriers, L., McCarty, L., et al. (2021). Active learning:"hands-on" meets "minds-on". *Science*, 374(6563):26–30.

Zhang-Kennedy, L. and Chiasson, S. (2021). A systematic review of multimedia tools for cybersecurity awareness and education. *ACM Computing Surveys (CSUR)*, 54(1):1–39.

Appendix A

IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received September 13, 2020, accepted October 12, 2020, date of publication November 9, 2020,
date of current version December 16, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3036976

A Study on the Active Methodologies Applied to Teaching and Learning Process in the Computing Area

MARIA IVANILSE CALDERON RIBEIRO[1] AND ODETTE MESTRINHO PASSOS[2]
[1]Institute of Computing, Federal University of Amazonas, Manaus 69067-005, Brazil
[2]Institute of Exact Sciences and Technology, Federal University of Amazonas, Itacoatiara 69103-128, Brazil

Corresponding author: Maria Ivanilse Calderon Ribeiro (ivanilse.calderon@icomp.ufam.edu.br)

This work was supported in part by the Institute of Computing at the Federal University of Amazonas (UFAM), Manaus, AM - Brazil, in part by the Foundation for Research Support of the State of Amazonas (FAPEAM) - POSGRAD 2017 (Resolution 002/2016), in part by the Coordination for the Improvement of Higher Education Personnel - Brazil (CAPES) - Finance Code 001, and in part by the support of Federal Institute of Education, Science and Technology of Rondônia (IFRO)/Campus Porto Velho North Zone.

ABSTRACT Active Methodologies allow an active process in teaching and learning contents, promote responsible student involvement and bring satisfaction and enrichment to educational practices and active learning. Generally, students have learning difficulties in Computer Science courses, as they need to develop computational skills and thinking. The goals of this article is to characterize and analyze the types of Active Methodologies that are being applied in teaching and learning activities in Computer Science. Thus, this investigation was carried out through a Systematic Mapping Study, focusing on the use of the types of methodologies in view of the results achieved. It presents students' perceptions, benefits, and difficulties in adopting these methodologies in the classroom. The results show 6 types of different Active Methodologies used in 35 publications selected, different types of techniques or studies that were used, the publications trend per year, the courses that were worked in analyzed publications, and some benefits and difficulty related to the adoption of Active Methodologies. Regarding to students' perception, we identified different type feelings. Thus, the contributions of this study consist in a research focused on the use of Active Methodologies in a very broad sense, including the perceptions of teachers and students regarding the use of different teaching and learning methodologies. In addition, it shows the specific benefits and possible difficulties experienced in the use of Active Methodologies as teaching strategies. Consequently, some findings from this study may have the potential to support or direct choices of these methodologies in different Computer Science courses.

INDEX TERMS Active methodologies, teaching in computing, learning, students' perception.

I. INTRODUCTION

The old method of teaching in which students were getting used to stay seated on their places, writing down and listening to a lecture of a teacher, have changed [1]. The nature of computer use has changed remarkably in the past fifty years. However, most Computer Science (CS) courses are still often teaching through that old paradigm that is not adequated to deal with modern concerns. Even in the face of the current generation of students and the nature of computing, most computer courses are still teaching in traditional ways [2]. That scenario needs a new conception that can brought a profound pedagogical renewal that requires knowledge and domain of new methodologies [3].

Active methodologies (AM) can support the development of self-competencies and skills. Because, on an increasingly complex society, mere transmission of information no longer characterizes an efficient teaching and learning process [4]. Therefore, AM are teaching strategies centered on the effective participation of students in the construction of the learning process, in a flexible, interconnected and hybrid way [5]. Also, on the new way, the act of learning needs to become a reconstructive process that allows the students to establish different relationships between facts and objects, producing resignifications and reconstructions and contributing to their application in different contexts [6], brings satisfaction and enrichment for both teacher and students [7].

The associate editor coordinating the review of this manuscript and approving it for publication was Chia-Wen Tsai.

Although there are several papers investigating about AM in teaching and learning, there is still a shortage of papers that shows how these AM influence and also can be exploited for support knowledge of teaching and learning in CS courses. Therefore, we have identified the need for a comprehensive research bout influence and support AM in teaching and learning in CS courses.

Systematic Mapping Studies (SMS) provides an overview of a research area, identifying the quantity, the types of research carried out, the results available, in addition to the frequency of publications over time to identify trends [8]. Our aims in this study, is characterizes and analyzes types of AM most applied to teaching and learning activities in CS courses. It also presents the students' perceptions about the AM applied in teaching, some benefits, and difficulties in adopting the different types of AM inside of a classroom.

From an initial selection of 753 publications, we have identified 35 different publications that have used AM to support in teaching and learning in CS courses. From the selected publications, the following AM were the most cited: Gamification (GM); Problem-Based Learning (PBL); Project-Based Learning (ProjBL); Peer instruction (PI); Flipped classroom (FC); and Team-Based Learning (TBL). Also, we have identified the research goal per AM applied, the publications trend per year, the main proposals of the studies mapped, technical or studies used per methodology, the computing area and active methodology applied, the courses that were worked in analyzed publications, and some benefits and difficulty related to the adoption of AM. Regarding to students' perception, we identified the feelings of Satisfaction in learning the content, Motivation to learn content and Feeling of mastery of content.

Besides this introductory section, this paper is structured as follows: Section 2 present related works; Section 3 describes the research method to define search strategy and research questions; Section 4 discusses the results; finally, Section 5 describes limitations and validity of this article, and Section 6 shows some conclusions.

II. RELATED WORKS

In recent years, several studies have reported evaluations and comparisons regarding of use of AM as teaching strategies in computer science e.g., [9]–[11]. The goal of those studies was to demonstrate the use of the AM with a focus on its use and apply them as teaching methods. Those works point out to AM as possible support to improve teaching in the computing and, consequently, to minimize the avoidance of courses in the area.

Raes *et al.* [12] present the results of an experiment on that subject comparing the students' learning experiences in a lecture face-to-face versus virtual students. The results show that although the hybrid virtual classroom is promising flexibility in education as it gives to students the choice for when and where attend the course, it is also the most challenging one to teach and to learn as a remote participant.

Farias & Nunes [13] argue that many studies in computing and education science present tools or environments related to teach in programming that does not reach the expected effect on the students' learning experience. That research also shows some relevant studies that, involving active programming learning for high school and university students, contribute to the construction of an innovative educational scenarios involving active programming learning.

Silva & Oliveira [14] have published an experience report of how robotics fits as an effective instrument from the perspective of AM for education. Those authors point out that robotics in education presents a possible path to be taken and used as a strategy applied to AM for the development and assimilation of knowledge, capable of promoting increasing participation and interest on the part of students, inside and outside school.

Moreno [15] proposes the discussion of evaluating how the principles, values, techniques and agile development processes can be relate to the AM to optimize teaching and learning process in introductory programming subjects with focus, specifically, on students of the high school. Thus, that author seeks to understand how teaching and learning, in introduction to programming, can be optimized by converging the active techniques widely discussed in the literature with those considered agile in software engineering.

The paper of Silva *et al.* [16], by his turn, seems to aim to present an systematic literature review on the use of digital games on teaching of programming for beginners in computing at university level in the last decade in Brazil. The authors argue that teaching programming is part of the basic academic training in CS and related areas.

We have observed that the use of AM in the context of learning computing still faces some challenges, mainly in relation to the attitude of the teacher and the students in relation to the use and effective applicability of the methodologies. In addition, the students' posture in relation to his autonomy for studies or in relation to an active posture for his studies inside or outside classroom, negatively reflect on skills related to computational thinking and learning computing [17], [18].

Resuming, all those previous works show that, besides a positive differential for teaching computing, today the adoption of AM are still a challenge because some teachers have resistance to adopt new teaching techniques. Furthermore, the training of new professionals in the area of technology, as well as in other areas of knowledge in computer programming, is a challenging task for hundreds of scientific researches, proposing several strategies that address from robotics, programming language, and educational games and to pedagogical approaches [19]. Also, that review has concluded that students have their own problems related to autonomy for study.

III. METHODS

For this paper, we have conducted SMS to collect all evidences that fit eligible criteria pre-specified following the

recommendations presented by Kitchenham & Charters [20]. Those kinds of studies also help to identify gaps in current research in order to suggest areas for further investigation [21] and by gathering, synthesizing and reviewing study findings [22]. So, this map was undertaken in three phases: planning, conducting, and reporting based on a systematic review protocol. Details of those stages are described in the following sub-sections.

To examine the current use of AM applied in CS courses, our research questions were:

RQ1. Which types of AM are being applied in teaching and learning activities in computing courses?

RQ2. Which are the courses' perceptions about the AM applied in teaching?

RQ3. Which are the benefits and difficulties in relation to the adoption of AM?

A. DATA SOURCE

The main digital libraries that were used to search for primary studies was Scopus, because: (i) It provides Index for publications of most events in the computer and education area, according to Dybá *et al.* [23]; (ii) It is an important repositories and are widely used for research in the scientific community; (iii) Its databases provide the best results, have strengths in different areas and return papers from more traditionally indexed [24]. In addition, we also manually searched the symposium and conference proceedings and journals in which relevant studies to the computer and education area domain had previously been published: (i) Brazilian Symposium on Informatics in Education (SBIE); (ii) Brazilian Symposium on Games and Digital Entertainment (SBGames); (iii) Computer Workshop at School (WIE); (iv) Computer Education Workshop (WEI); (v) New Technologies in Education Journal (RENOTE); (vi) Journal of Informatics in Education (RBIE); and (vii) International Congress of Educational Informatics (TISE). The period of analysis of the proceedings of the symposium above was from 2010 to 2019. Also, it is important to mention that in the Brazilian Symposium on Games and Digital Entertainment the search was made in all tracks; however, the proceedings of this symposium of the year 2014 was not used, as it is not available at the time of the search.

B. SEARCH STRATEGY

For the construction and refinement of the search string, we have flowed the recommendations of Petersen *et al.* [8]. We have performed the three advise steps, which are: (i) Consultation to the experts for the construction of the mapping protocol; (ii) The refinement and (iii) Test of the search string and selection of the keywords for manually searched.

Furthermore, to facilitate the identification of search string terms, the terms were defined from the Population, Intervention, Comparison, Output (PICO) parameters, by Kitchenham & Charters [20] and the terms related to each parameter, when applicable (see Table 1).

TABLE 1. Terms used to instantiate parameters PICO.

Parameter	Used terms of search
(P) Population: Works published in conferences or journals presenting models and artifacts involving active methodologies	"active methodologies" OR "methodologies" OR "active learning" OR "teaching computer" OR "educational experiences" OR "computer programming skills"
(I) Intervention: Teaching-learning activities of computer science course	activities OR teaching OR learning OR course OR computing
(C) Comparison: It does not apply, because it is a characterization review	not applicable, is a mapping for characterization
(O) Output: Concepts, definitions, methods, models, applications, discussion of problems	concepts OR approach OR method OR model

(("active methodologies" OR "methodologies" OR "active learning" OR "teaching computer" OR "educational experiences" OR "computer programming skills")
AND
(activities OR teaching OR learning OR course OR computing)
AND
(concepts OR approach OR method OR model))

FIGURE 1. Search string used in the SMS.

Moreover, we used the search string in which Boolean OR has been applied to join alternate terms and synonyms in each main part; and Boolean AND has been used to join the three main parts. Figure 1 shows the search string of this work.

C. STUDY SELECTION

During the SMS, only relevant publications to the research question were selected for further analysis. Kitchenham & Charters [20] had suggested the definition of inclusion and exclusion criteria for papers that are returned by the search string. Any paper that did not meet all the inclusion criteria must be deleted. Therefore, we have used the five inclusion criteria to select articles (see TABLE 2).

D. SEARCH RETURNS AND DATA EXTRACTION

The literature search identified 753 publications. After the removal of 105 duplicate papers, applying the 1st filter (selection based on title, keywords and abstract) and 2nd filter

TABLE 2. Inclusion set of criteria.

id	Inclusion criteria
Inc 1	Publication must present topics related to the use of AM in teaching activities in the context of computer and education area.
Inc 2	Publication must present details for applying of AM in teaching activities of course computer science.
Inc 3	Publication must present concepts, approach, method or model in the context of computer and education area.
Inc 4	Articles were written in English and Brazilian Portuguese.
Inc 5	Articles were published between 2010 to 2019.

(complete analysis of the study) to removal out-of-scope and duplicate papers (see Fig. 2), the final number of studies reviewed was reduced to 35 relevant papers that are listed in the (Supplementary data Appendix 1).

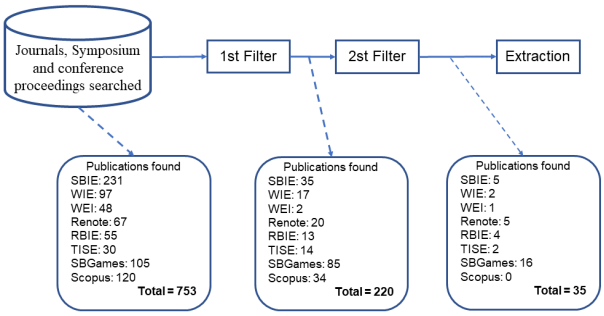


FIGURE 2. Shows the results obtained that answers the research.

In this scenario, we can observe that most of the researches selected in this mapping were published in SBGames, i.e., in this symposium the community can find most of publications related to AM teaching strategies in computer science. Also, we can observe SBIE, another symposium also used by the computer community in education. Furthermore, the journals RENOTE and RBIE also have been used.

IV. RESULTS AND ANALYSIS

In this section, we present our results according to the research questions.

A. ACTIVE METHODOLOGIES IN TEACHING OF COMPUTING

RQ1. Which types of AM are being applied in teaching and learning activities in computing courses?

Fig. 3 shows the results of 6 different types of AM most applied in the teaching and learning activities of different CS courses: GM; PBL; ProjBL; PI; FC and TBL.

Between publications analyzed, were applied the TBL and PI, but those types were used not alone in the work. For example, on the publications [10], [25]–[27] the researchers have used more than one AM as teaching strategies in their research. TBL methodology goes beyond covering the content, as it allows the use of course concepts to solve problems. Therefore, on TBL, learning is favored through group interaction; after the questions raised, they are discussed within the groups, the answers are presented to the class, thus revising the main points of the subject [18], [28]. The professional school educators have found TBL particularly attractive because it offers powerful solutions to several major problems they face in teaching [29]. Thus, it is an active learning method developed to help students achieve goals of the course while learning how to function in teams.

FC is a methodology that means that events that have traditionally taken place inside the classroom now take place outside the classroom and vice versa. The use of learning technologies, particularly multimedia, provides new

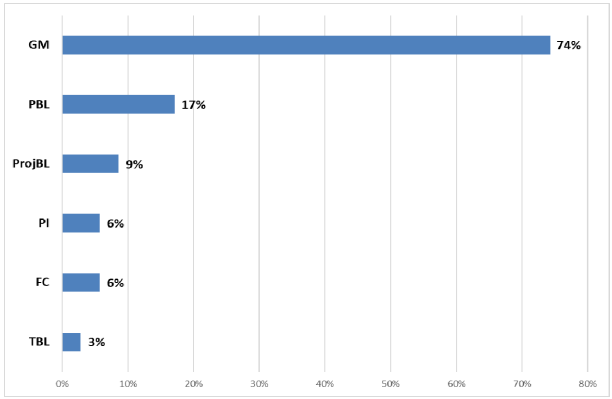


FIGURE 3. Types of AM are applied. In increasing order: TBL; FC; PI; ProjBL; PBL and GM.

opportunities for students to learn, opportunities that are not possible with other media [30]. Basically, the concept of a flipped class is this: what is traditionally done in class is now done at home, and what is traditionally done as homework is now completed in class. In the flipped model, the time is completely restructured [31].

Furthermore, it is important to mention that the AM applied in conjunction with TBL and PI were the PBL, ProjBL, FC and GM together, i.e., around 14% (5 publications) of set analyzed. PI is collaborative methodology, developed by teacher Eric Mazur of Harvard University. It aims to involve all students during class, promoting activities in which they are encouraged to apply the concepts discussed at that time, while explaining them to their colleagues [18]. Therefore, approach with PI will help them learn, mainly because the students have to play a central role in their own learning with the instructor as their coach [32] [33].

PBL is a methodology emerged in the 70s, through the doctor Howard Barrows, being applied in medical classes and has gained acceptance and is becoming increasingly effective across a variety of course in higher education and an educational method for the teaching of computing is being used in computer science [9]. It is considered to be an educational strategy centered on the students, which helps him in the development of reasoning and communication, essential skills for success in his professional life [34].

Fig. 3 shows the mapping results, PBL, ProjBL, PI and FC types of AM that were used in a few publications. Those methodologies were used just in 17% (7 publications), 9% (3 publications) and 6% (2 publications) respectively of the total set analyzed. Therefore, researches that use the methodology PBL, in general, seems to challenge the students to perform high-level mental tasks, such as analysis, synthesis and evaluation. That method was used totally focused on the health area, but nowadays it has been accepted in the teaching of several areas of knowledge, mainly in Computing, both on elementary and on high school [9].

ProjBL is a methodology that organizes learning around projects, according to definitions found in Project-based learning handbooks for teachers. For this methodology,

projects are complex tasks, based on challenging questions or problems, that involve students in design, problem-solving, decision making, or investigative activities; give students the opportunity to work relatively autonomously over extended periods of time; and culminate in realistic products or presentations [35] [36].

Ultimately, around 71% of the studies (25 publications) of set analyzed presented works used GM as methodology to teaching courses or measure the learning. Thus, it is the use of game elements and design for purposes unrelated to games to get people motivated to achieve specific goals [37] [38]. Therefore, gamification can provide an edge in learning delivery when it is designed, developed, and deployed properly. Thus, the effort should not focus solely on points, badges, and leader boards. Results of the change have bilateral nature– they can affect students' results and help them to understand the educational content and create conditions for an effective learning process [39]. We have observed that, on large-scale, concepts of gamification are applied into other areas almost every day [40]. And, on educational context, it is no different, because those researches have linked positive impacts on game experiences in different cases, particularly on player's experiences and interactions during learning courses on computing.

That scenario address that there is a greater production of procedure or technique for teaching in computing applying AM. Thus, we can notice a great concern of the educational community that demands of teachers to prepare dynamic classes and master how to use the different AM to motivate students in addition to producing different materials. It is important, because an increasing number of strategies are gaining prominence in terms of getting students' attention, changing the traditional way of teaching and learning [1], and support the search for learning objects that might help teachers on that task.

We also have analyzed the main objective of the researchers in relation to the use of AM; those research interests present different challenges encountered in the studies for the application of AM on teaching of computation (see Table 3). Those investigated studies, in general, present their objective as a search for different strategies for teaching and learning applying different AM, which results in different solutions for teaching.

This overview of research goal shows the importance of knowing about that topic and the need for developing more work to understand and increase useful tools to improve different means about teaching and learning. It is so because educational field is facing an impasse due to numerous changes in society: it is necessary to evolve and to make everyone learn in a competent and constructive way [41]. Also, the current social demands require much more from teachers in the classroom, not only a new attitude, but new ways of transmitting their knowledge [42].

Table 3, in general, shows concerns related to teaching and learning of content by students, research to awaken motivation and interest in the studied subject, research for developing quality professionals applying methodologies that leave behind the traditional method of decorating content and its mechanical reproduction. Thus, those researches have used different types of AM, as case studies involving the active learning of programming for students of different levels of education, since basic education up to postgraduate. Therefore, we can conclude that more research is required to improve innovative educational scenarios involving active learning in computer science.

Notably, the results shows that around 63% of the students (22 publications) analyzed applied AM as teaching strategies for teaching University. While around 20% of the studies (7 publications) were focused on Technical High School, just around 12% of the studies (4 publications) are related with others education levels, e.g., Postgraduate studies, High school, and Basic education. That scenario shows that AM are being adopted at all levels of education, i.e., it reveals that education professionals, especially in computing, are looking for innovate in relation to their teaching methodologies adopted in the classroom (see Fig. 4).

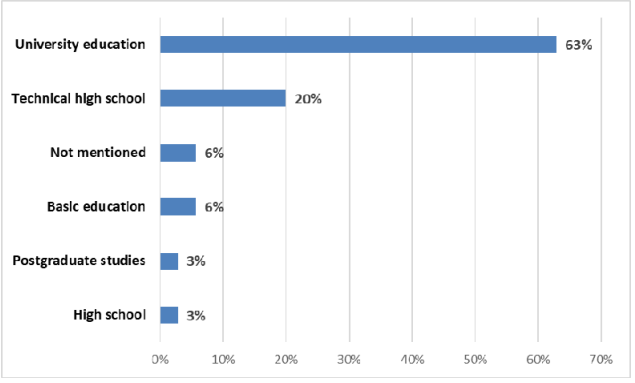


FIGURE 4. Education level system: High school, Postgraduate studies, Basic education, Not mentioned, Technical high school, University education.

Some results also revealed that the effects of the transition from a teaching-centered model of education to a learning-centered one involves a great cultural change for the University as an educational institution [43]. However, the social changes have leading to a change of perception in the teaching-learning process that promotes the emergence of the so-called active learning methodologies [18]. It is more perspective on higher education, so, we have analyzed the education level system in publications, because in higher education, there are growing trends toward flexible online course delivery [44].

Besides that, it is important to recognize that during the past few years, primary studies have been increased published regarding use AM as teaching strategies in the context of education [45]–[47] and have helped to refocus academic researches on method to teaching and learning, but it is still insufficient. Table 4 shows AM apply by education level system.

TABLE 3. Research goal per AM applied.

AM	Research Goal	Reference
ProjBL, PBL, TBL and PI	Increase students' motivation, belief in self-efficacy and autonomy, enhancing the learning of programming languages.	1
	Report the results of experience as the case study carried out for the first two periods of curriculum reformulation.	1
ProjBL and PBL	To present an analysis of AM aimed to learning undergraduate and graduate students.	1
GM, FC, and PI	Report experience of the use of gamification elements in conjunction with the hybrid teaching methodology in the inverted classroom modality applied to programming course.	1
FC	Resolution of practical problems, involving automation of solutions for monitoring network equipment and user actions, log analysis, data synchronization and automatic emission of alert messages, implemented in the Python programming language.	1
PBL	Shows how the use of PBL can contribute both to learning software engineering and to stimulating professional skills. Train programmers based on methodologies that focus on students learning and that favors the development of programming skills.	3
	Tests AM in undergraduate teaching and checks if students feel more engaged and motivated to carry out activities and develop their knowledge.	2
GM	Investigates the influence of gamification on students' engagement and performance in programming learning.	11
	Motivates students by serving as a tool to aid the teaching and learning process, making the practice of logic skills, and casting a light on creating algorithms, making fun and relaxed activity to be applied both inside and outside the classroom.	9
	Helps students to practice some introductory programming content in a fun way.	5

TABLE 4. AM applied by education level system.

AM	Education level system	Ref.
ProjBL, PBL, TBL and PI	Technical High School	1
ProjBL and PBL	University Education	1
	Postgraduate studies	1
GM, FC, and PI	University Education	1
FC	University Education	1
PBL	University Education	4
GM	Basic Education	1
	High School	1
	Technical High School	7
	University Education	17

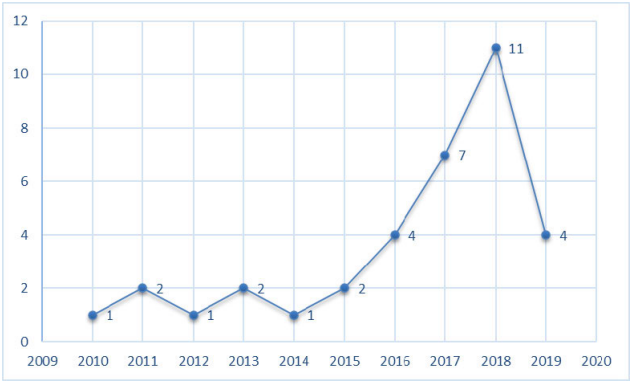


FIGURE 5. Publications per year. The figure includes line graphs showing number of publications returned by search and publications that met the inclusion criteria (2010-2019 period).

We have observed that GM is being applied more among the methodologies mapped in this study, that its use ranges from basic education to graduation and it is present in 74% of the analyzed publications that use GM as an AM in teaching. Following, PBL methodology appears in 4 surveys, i.e., about 11%. Finally, the TBL, PI, FC and ProjBL methodologies are mentioned each in only 1 survey, i.e., about 2.80% of the analyzed publications.

The results presented above show us that, among the most significant and applied AM as teaching strategies in the computation, we can find GM and PBL. However, a few applications in the classroom were also using FC, PI, TBL and ProjBL. This scenario demonstrates that the use of those teaching methodologies, as teaching strategies in computing, can be seen as effective teaching methods, as they support the stimulation of the students' initiative to create opportunities for learning content inside and outside the classroom, because AM can promote proactivity, commitment to the educational process and linking learning to significant aspects of reality [48].

In additional, we also have analyzed the distribution of publications per year (Fig. 5). The interest of investigations on topic of AM as teaching strategies in computing has began around 2010 with 1 publication. In the following years, from 2010 to 2015, publications remained stable, 1 or 2 paper per year. But, in 2016 and 2017 the number of publications has increased up to 4 and 7, respectively. 2018 confirms that evolution becoming the year in which most researches were publishing on that subject, 11 in total. But in 2019, that number fell again, just 4 publications.

However, we have considered an incipient number of publications for whom that applied of AM, supported by the principle of autonomy into the classroom, becomes of vital importance due to impact they can have on students' learning [49]. Also, AM have numerous forms of use for teaching and learning the computing courses. Furthermore, the applying of AM can favor the autonomy of the students both in face-to-face and distance education, favoring curiosity, stimulating individual and collective decision making, arising from activities of social practice and of students' contexts [50]. Therefore, AM emerges as a proposal to focus on the process of teaching and learning in the search for the

TABLE 5. Students' perceptions about AM.

AM	Students' Perception	Reference
ProjBL, PBL, TBL and PI	Developed confidence and interest in learning the contents.	1
ProjBL and PBL	Developed skills and experience in carrying out projects with execution efficiently.	1
GM, FC, and PI	Improve academic performance and, consequently, increasing programming skills.	1
FC	Represented a significant improvement in the group's performance.	1
PBL	Most students managed to learn the new concepts.	3
	Best comprehension and facilitated learning even for students inexperienced in programming.	2
	Focused on learning, being motivated to study, and learn the content provided. Students seem to have more desire to learn.	2
	Realized the importance of their own commitment, started to value the engagement of the team and collective practices.	1
GM	Learned basic programming concepts and developed logical-mathematical reasoning. There was a significant gain in learning the contents.	4
	Aroused great interest in the students remained engaged in their goals until the end of the activity.	2
	Stimulated by the new methodology, many students continued to attend the course.	3
	They had a pleasant experience and they managed to learn the proposed content.	2
	Students cited the game's ability to teach programming in a fun way.	2
	The students started to perform activities in the middle of the weekend, so they surprised, and this showed the degree of involvement provided by the game.	3
	Students assessed the gamified course more motivating than a non-gamified course and considered the practice interesting for learning.	4
	They wanted to learn more about the topic and noted that there was no difficulty in understanding. They were able to associate it with content already learned and that the level of complexity was adequate. The majority agreed that they learned surprising or unexpected concepts.	2
	Participating in the proposed activities, students experience and carry out exercises that allowed them to verify their speed and practicality in performing them.	2
	Provided a positive experience for players regarding social interaction and fun. In addition, students pointed out that the experience with the game will contribute to their performance in professional life.	
	Understanding about collaborative activity carried out by teams of students, considered the practice interesting for learning. It influenced them to study, so that the goals and activities could be fulfilled, which allowed them to advance in the game by their own effort.	3
	Learned and remembered the concepts of project management, because allowed students to fix the subjects better understanding the way of creating the missions, testing, and consequently improving the applicability of techniques.	2

active participation of all involved, centered on the reality in which they are inserted [51].

Thereby, that scenario shows combination of learning by challenges, the use of real problems and the games with the flipped classroom allow students to learn by doing, together and at their own place [41]. At the same time, teacher acts as an advisor, supervisor, and facilitator of the learning process, not only as the sole source of information and knowledge [45]. Thus, AM are teaching strategies centered on the effective participation of students in the construction of the learning process, in a flexible, interconnected and hybrid way [17].

B. STUDENTS' PERCEPTION ABOUT ACTIVE METHODOLOGIES IN TEACHING COMPUTING

RQ2. Which are the students' perceptions about the AM applied in teaching?

Table 5 shows AM applied and description in relation to students' perception; there were feeling of Satisfaction in learning the content, Motivation to learn content, and Feeling of mastery of content. Those results have indicated, in general, that the students' perceptions were positives regarding AM applied as strategy in teaching, i.e. AM enable greater interaction between teachers and students, benefiting both. Students acquire greater protagonism and independence in teaching and learning process. Teachers have a great opportunity to innovate, to propose new ways to teach and to stimulate their students [42], but it is necessary their full commitment with all process. It is required from them to be aware of their role in order to achieve successful results, also need to actively practice the directions proposed by that type of methodology. So it can be seen as liberating for both in the sense of leaving the traditional forms of teaching and learning in favor of a new way to enhance the productive involvement of the teacher and the consequent active and innovative learning of the students.

The informations obtained from this mapping have made possible to identify the students' perceptions aroused with the use of different types of AM, enabling teachers to identify pedagogical practices that most attract student's participation. Because, reflecting on education in the contemporary context, includes, among other important aspects, the discussion on AM produced in a collaborative way and its implications for the experiences of students in the classroom [17].

We have considered that, from those three dimensions, the students' perspectives presented in some publications, in general, can support the construction of knowledge in order to monitor and to measure the data related to the acquisition and development of knowledge, skills and attitudes achieved by students, according to the research goals in question.

TABLE 6. Dimension of students' perception.

Dimensions in relation to students' perception	AM	Reference
Satisfaction in learning the content	GM, ProjBL, PBL, FC, PI and TBL	23
Motivation to learn content	GM, ProjBL, PBL, FC, PI and TBL	22
Feeling of mastery of content	GM, ProjBL, PBL, PI and TBL	11

Also, many teachers imagine that all learning, including expository class, is inherently active. They consider that, while the students participates watching an expository class, they are actively involved, but cognitive science researches has indicated that students must do more than simply listen for an effective learning [18].

Therefore, from the adoption of active teaching methodological practices combined with Digital Information and Communication Technologies (TDICs), inherent to the CS, today's teacher and future teachers can make teaching and learning process more attractive to the eyes of students [52] and, for that, it is possible to use the AM as teaching strategies in computing new didactic methods seeking to combine the use of technologies, pedagogical knowledge and the different types of AM that currently exist for teaching.

Table 6 has mapping results organized in dimension identified to reply research question RQ 2. Not all articles explicitly have presented the students' perceptions regarding the applying of AM. Thus, some perceptions presented are based on the researchers' observations made during their field research. It can be observed when in some publications survey were used to learn about the acceptance and the applying of AM to learn the content. It was possible to identify three dimensions in relation to students' perception presented in those publications: a) Satisfaction in learning the proposed content: concerns the perception of the students about his learning of the concepts treated in the teaching of the subject's contents appears in 23 publication (65%); b) Motivation to learn the content: it is related to the students' perception of will and searches to learn even more about the concepts treated in the teaching of the subject's contents and it was observed in 22 publication (62%); and c) Feeling of mastery in relation to the proposed content: this perception is related to the ability to practice activities outside the classroom, to teach or share the content learned with colleagues and was notice in 11 papers (31%).

The new generation of students from the end of the 20th century seems not to be interested in attending a class in the same way every day [52]. Originally, the active method of learning works with the child's experience, so the teacher could support his students to reflect and encourage him to make decisions [53]. To understand the students' perceptions about the AM applied in teaching, we have applied different technical or studies to obtain the results presented.

TABLE 7. Technical or studies used per methodology.

Technical or Studies Used	AM	Reference
Online tool and survey	GM and PBL	6
Survey	GM	4
Survey and Game	GM	4
Computer tool	PBL and GM	2
Case study	PBL and GM	2
Case study and survey	PBL	2
Survey and framework	GM	1
Survey, online tool, and game	GM	1
Survey, online tool, and WhatsApp	FC	1
Survey and computer tool	GM	1
Case study and tool	PBL	1
Case study, survey, and game	GM	1
Framework	GM	1
Survey, framework, and online tool	GM	1
Game and interview	GM	1
Online tool and manual game	GM	1
Survey and manual game	GM	1
Survey and online course	ProjBL	1
Online tool	GM	1
Survey and game	GM	1
Survey, workshop, and focus group	GM	1

We can observe in mapping results that some researches have used more than one different technical or study to AM apply. Table 7 presents that scenario mapped.

It shows us that most of the researches have used the following technical or studies to get and analyze the perceptions of students: Survey, Online Tool, Games, Case Study, Framework, Manual Game, Computer Tool, Online Course, Focus Group, Interview, Workshop and Software. We have observed that most researches have used survey technical to gathering information about students' perspectives. In general, the goals was to explore the perception of students in the CS in order to analyze improvements, or the lack of it, in learning, according to different active teaching methodologies that have been used and in order to stimulate the knowledge skills of the studied contents.

Thus, we have noticed that survey, online tools, games, and case study were technical or studies most used by researchers. Also, those technical or studies were used together with others on most researchers analyzed, i.e. researchers as Oliveira & Barros [54] that applied the case study, game, and survey to get data about students' perceptions. In this sense, interview, focus group, workshop and online tools, online course, manual game, framework software and use of WhatsApp were combined with those technical or studies to understand the students' perceptions about the AM applied in teaching, presented in Table 7.

Thereby, the use of those techniques reflects the interest of the author to know how the students' perceptions about the AM applied in teaching of computing courses

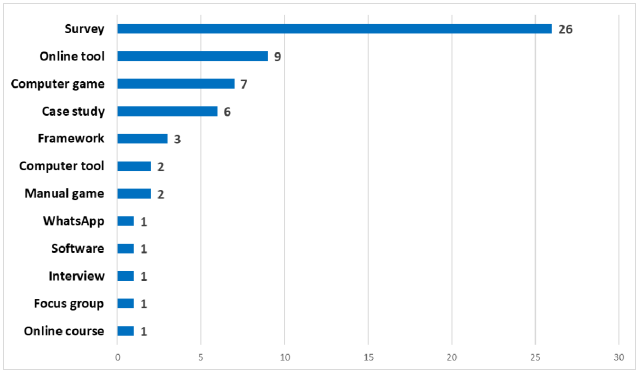


FIGURE 6. Type of technical or studies used in research. Bars graph showing the number of technical that used to apply AM. In increasing order: Online course, Focus group, Interview, Software, WhatsApp, Manual game, Computer tool, Framework, Case study, Computer game, Online tool, and Survey.

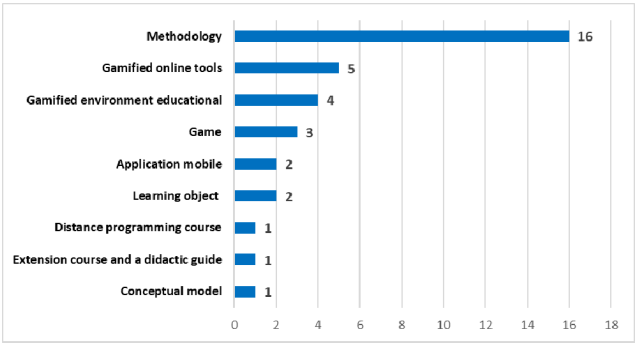


FIGURE 7. Proposals de studies mapping. Bars graph showing the type and number of proposals of studies that used to apply AM. In increasing order: Conceptual model, Extension course and a didactic guide, Distance programming course, Learning object, Application mobile, Game, Gamified environment educational, Gamified online tools, and Methodology.

could better support the learning process and could motivate researches to produce knowledge about process of teaching. Also, it becomes an important way to provide different and new ways of getting experiences in computing. So, in this context, it can be seen that most of the researchers have applied AM in teaching to instigate the students to participate in the class, through group works or discussion of problems. Those type of methodologies, as Lovato *et al.* [18] points, are those that place students as protagonists, while the teachers are mediators or facilitators of the process.

Fig. 6 shows some scenario about number of publications by type of technique or study used on the researchers analyzed. In relation of survey, this technical appears in 26 of publications. In sequence, the most used were online tools in 9 publications, the computer games in 7, and case study in 6 researches. The techniques or studies less used were the framework in 3, manual game and computer tool appears in 2 publications, while the others were, in general, used just one per each publication.

We observed that those techniques or studies have been used in education as a form to know different types of knowledge and to promote a critical, creative, and reflective attitude concerning to adoption of AM strategy in computer science. Because, the technological evolution that we are witnessing today brings challenges on the reality of education and so it points to need for changes that allows a redirection capable of answering those challenges and improve the skills of the new generations of students, since their educational beginning, allowing them to develop knowledge and skills for a future in which technology undergoes continuous developments [14].

Fig. 7 presents the distribution of publications per proposal presented by the research. We have noticed that between those proposals analyzed are commonly presented in works that seek to analyze students' school performance and as well as improve teaching and learning process. Also, those proposals seek to provide experience in the use of gamified on-line tools, to support teaching and learning in the most diverse course of computing area. They also seek to start the stimulation of computational thinking through a gamified application and to motivate students to practice some introductory content in the area of computing in a fun way.

Table 8 shows the mains courses worked on those research analyzed. It may reflect, in general, two propositions regarding traditional methodologies in computing: (i) those courses are more difficult to learn; and (ii) they are more difficult to teach using traditional methodologies. In any case, we can infer that those researchers were looking for solutions to problems that are evident, and we understand that this problem is not an exception. Therefore, the courses presented above can reflect the research' interest and necessity in knowing how AM can be used like teaching strategies centered on effective participation both the teacher and the students. Thus, on non-collaborative processes, teachers have more roles to perform, as work is controlled and organized by them, while in the collaborative process it is more opened and students become more active [55].

Thereby, it might help students to acquire autonomy over their interests and skills, motivating them to search and to research on the topics brought by contents. In short, they might realize that is worth learning [56], because with the appearance of the so-called AM of learning, the students becomes the protagonist and the use of this type of methodology allows the development of new skills, such as initiative, creativity, criticality reflective, capacity for self-assessment and cooperation to work as a team; and the teacher acts as an advisor, supervisor, and facilitator of the process [18], being necessary for this sense observing the different learning processes.

In this scenario, observing and evaluating the students' perception of his or her learning while AM are adopted as teaching strategies is not a trivial task. However, the literature shows that several studies are being carried out in order to analyze the potential of AM for learning [49], [52] or to facilitate retention of knowledge by students [18], [57]. Nevertheless, so far, there are few studies that aim to evaluate students' experiences in relation to active teaching methodologies [58]. Also, we have noticed that it is necessary, due to the similarities between some active methodologies, to give

TABLE 8. Courses worked in analyzed publications.

AM	Course	Reference
ProjBL, PBL, TBL and PI	Computer programming	1
ProjBL and PBL	Software testing	1
	Computer programming	1
GM, FC, and PI	Project management	1
	Computer programming	1
FC	Computer network	1
PBL	Programming logic	1
	Software engineering	1
	Troubleshooting I	1
	Computer programming	2
GM	Algorithms	2
	Data base	1
	Data structure	1
	Project management	1
	Introduction of informatics	1
	Introduction of programming	2
	Programming logic	4
	Scientific methodology	1
	Computer programming	9
	Free and proprietary software	1
	Theory of graphs	1
	Software testing	1

support for teachers on some type of learning. It is so because, in general, various AM were applied on those articles analyzed, however, in the course of this research, we have faced a lack of classification that could clearly present the use of AM in relation to the categorization of types of learning.

Thus, we have reached a conclusion that the interest in knowing the students' perception in this mapping of the literature can be very useful in different aspects regarding teaching and learning process. For example, it may be possible to know if some students need additional help to understand the information explained in the classroom or even to promote the intuition of means to help those students in their presented difficulty. Each study here analyzed has presented students' perceptions in different ways, i.e. they did not clearly show types of perception of the students regarding percentage of acceptance in relation to the applying of AM in the classroom. So, we think that is important present one classification that could help following researches and teachers that are interested on applying such methodology.

Therefore, considering the course of the computer science organized in three different fields, following the Reference Curriculum for Undergraduate Courses in Computing and Informatics by Brazilian Computer Society (SBC): I) Computability Fundaments (CF), which comprises the core of subjects that involve scientific parts and the fundamental techniques for the solid formation of the various computer program; II) Computing Technology (CT), which comprises the core of the course and represents a set of aggregated and consolidated knowledge that enable students to develop solving problems in the various application domains; and III) Information Systems (IS), which comprises the core of the course that enables students to use the resources of Information Technology in solving problems of productive sectors of society.

Table 9 presents the AM used per field of computing area in relation to the course worked on the researches analyzed in this paper. Therefore, it reflects, generally, that the GM and PBL were types of AM more used in three different area of computing. Thus, those types of AM can provide positives students' perception in relation of learning of course, because learning is most effective when it is active, experiential, situated, problem-based and provides immediate feedback [59]. Also, this research demonstrates that students engagement in course remains mixed, because, the GM and PBL were used in tree field of computing area and they are often put forth as a possible answer to change the teaching and learning landscape and to make it more attractive and interactive for students. Those types of AM, generally, can support students in relation of learning, because our results show that students' motivation and engagement is positively related to use of different AM in course.

TABLE 9. Computing area and active methodology applied.

Computing area	AM
Computability Fundaments	GM, PBL, ProjBL, TBL and PI
Computing Technology	GM, PBL and FC
Information Systems	GM, PI, ProjBL and PBL

ProjBL and PI are the types of AM most used to teach in the fields of IS and CF. It might means that teacher has to adapt his/her programming languages or programming technique, because the quality of the teaching is partly dependent on the teacher's competence in using the technology [12], and this type of course requires some changes in the teachers' teaching methods and different approaches in order to motivate and to engage students in program. ProjBL and PI are most used both in IS and CF. FC and TBL are most used only in CF. It might mean that teacher needs to actively learn how to work with different AM and has to get opportunities to try things out and to evaluates outcomes taking students' perceptions as basis, because courses of CF are core of CS learning.

So, to deal with the needs for more significant learning in CS, those types of AM can be developed and provide a richer engaging learning experience for teachers and students. Those methodologies can motivate processes of teaching and learning in different courses, because those methodologies might stimulate and engage all people involved in this process of teaching and learning. Also, it agrees with education literatures when they address AM like a set of processes, procedures, techniques, and tools that involve the students actively in the teaching and learning process.

TABLE 10. Benefits in relation to the adoption of AM.

AM	Description	Reference
ProjBL, PBL, TBL and PI	Improved students' confidence, encouraging them to seek answers to their questions on their own.	1
ProjBL and PBL	Integrated practice of different course. Development of high level social and cognitive skills.	1
GM, FC, and PI	Reconcile theory and practice and turn activities into meaningful learning.	1
	Significant increase in students' averages. The failure rate dropped. The approval rating has increased.	1
	Significant improvement in group performance, significant increase in the average.	1
PBL	Recognition of the importance of collective practices in their formation. Stimulated self-assessment in relation to their own involvement in teamwork and the involvement of other colleagues. Greater approximation between theory and practice.	2
	It stimulated programming skills such as comprehension, analysis, observation of details, sequencing, and abstraction.	1
	The increased sense of responsibility of the students, who obtained engagement and course to study the contents related to the context of the problem, better understanding the situation presented; encouraging reading, the use of logical reasoning and discussions; encouraging more detailed and in-depth investigation of the problem presented.	1
	It allowed students having a vision of how the practical experience in a real project helps the team to have a better understanding of the theories, practices, methods, processes, and tools that involve Software Engineering. Make classes dynamic and attractive; and created an environment of cooperation between students. Encouragement of proactivity in the search for knowledge, confidence in your skills and that of your peers, commitment to solving tasks, teamwork, strategic thinking, and resilience in the face of difficult tasks.	1
GM	Make classes dynamic and attractive; and created an environment of cooperation between students, serving as a stimulus for their integral development. Encouragement of proactivity in the search for knowledge, confidence in your skills and that of your peers, commitment to solving tasks, teamwork, strategic thinking, and resilience in the face of difficult tasks.	4
	Caused a decrease in any interaction resistance. It provoked greater interest in the students, when it was evidenced that they felt stimulated to obtain a good performance. Position the students (player) in the role of a researcher involved in a rich narrative, thus simulating the activities that he should perform in his studies.	2
	Aroused creativity and the ability to think and act in a favorable condition in the conduct of computer studies, making the learning process more attractive and interactive, understood as an important tool to support teachers in school education.	2
	Considerable drop in the number of failing students. Improvement in students' performance in the course. Minimized the students' difficulties in relation to the difficulty in learning computer programming. It motivates students' interaction, and engagement to carry out the activities.	3
	Great impact on students' participation and development levels. Increase in attendance to classes. Improved students learning outcomes. Significant impact on classroom participation, and evidence that attention to support material and learning outcomes may benefit.	2
	It brought together several elements and references, such as board, cards, characters, and obstacles, in a simple system that adheres to the reality of the classrooms. The students developed five PC-related skills: data analysis skills, data collection skills, decomposition, abstraction, and data representation.	6
	Motivated the study of computer programming at the levels: basic knowledge of programming and basic concepts and assessment of the ability to read codes. Positive influenced on using gamification, engagement techniques to achieve the educational goals required to learn programming.	2
	Competitiveness influences students' motivation to learn and overcome challenges. It provides the students with learning in a playful and enjoyable way; makes it possible to relate scientific and abstract concepts with familiar terms, with the students' reality.	2
	Reduction in the number of failures in the courses.	3

C. BENEFITS AND DIFFICULTIES IN RELATION TO ADOPTION ACTIVE METHODOLOGIES IN TEACHING OF COMPUTING

RQ3: Which are the benefits and difficulties in relation to the adoption of AM?

We have observed the most studies, in general, obtained benefits when they adopted GM as an AM for teaching, see Table 10. However, they also reflects difficulties (see Table 11), caused by the lack of commitment of students outside the classroom. Thereby, we have chosen to present those reports in relation to the benefits and difficulties presented in the texts of the publications. In general, it can support the use or adequacy of new teaching techniques applying AM, in addition to sharing the related benefits and difficulties in order to direct new practices.

Attracting students' attention and keeping them involved are essential points for the process of learning and developing critical thinking. Teacher plays a role as an activator of learning [49]. Therefore, the adoption of this type of methodology for teaching in computing will certainly bring benefits for both students and teachers.

In the scenario presented by Table 10, we have understood that it is possible to infer that there is a positive influence on teaching and learning process in relation to the use of AM in CS courses. For example, GM, according to the analyzed works, is an engagement technique for achieve the educational goals required to learn programming. However, we also have observed that even with the positive influence of the use of AM as teaching strategies, there were yet difficulties (see Table 11), mainly in relation to the students' awareness

TABLE 11. Difficulty in relation to the adoption of AM.

AM	Description	Reference
ProjBL, PBL, TBL and PI	Lack of motivation and difficulties in understanding the content.	1
	Difficulty in motivating and mobilizing the attention of students. They can't obtain a broader view of the acquired concepts and establish a connection between theory and practice with greater effectiveness.	1
ProjBL and PBL	Not all students were able to acquire the skills in a timely manner to apply these concepts in solving the problem. Some students, when finishing a problem without completing all the requirements, had difficulties in continuing the same project in the subsequent problem. Finally, this change brought some difficulties in learning, the accumulation of deficiencies in projects of longer duration and the cognitive overload in dealing with several concepts simultaneously.	1
GM, FC, and PI	Lack of motivation of students to carry out practical activities. Lack of commitment on the part of some students, who insist on not studying the content before the class (face-to-face meeting), thus promoting a lesser use in the course.	1
FC	Leave the comfort zone, access the platform, and frequently participate in the flexible study environment available. Make students aware of how to access and participate frequently in the available materials, as students do not have the habit of accessing the study station or actively participating in face-to-face meetings.	1
	The course duration was considered insufficient for all the suggested activities to be completed comfortably.	2
	Difficulties of students with solving problems involving programming. Difficulties were also evidenced with basic questions about computer network theories. In autonomous studies, we observed difficulties for students to study alone, as it was a new practice for most.	1
PBL	Many academics have difficulties in positioning themselves in front of colleagues and tutors to expose their work and answer questions. Also, some aspects related to the programming and operation of the computer and difficulties in relation to technical issues.	2
	Identify which students are really engaged in the tasks and which are collaborating with the team, since simply sharing tasks does not represent group work.	1
GM	Low understanding of the basics in programing; a weak mathematical base; the lack of understanding of the statements; exercises dissociated from "real" problems and little extra-class availability for studies.	6
	Difficulties in coding, due to the lack of programming logic base, presenting a great deficit of basic technical knowledge necessary for the good progress of the course. Lack of basic prior knowledge to study data structure. Little time available for studies.	4
	The lack of motivation of students and their difficulty in maintaining a continuous pace of study.	5
	Large classes, which make it impossible to carry out individualized monitoring and heterogeneous, which present disparity of knowledge and learning pace.	1
	Diversity of students learning rhythms, combined with large classes and lack of motivation.	1
	The workload required to keep the rankings and the site up to date, as two different platforms are being used to present and correct challenges.	1
	The need for faculty involvement and collaboration for the success of gamification and the right time for its application. In addition, achieving an appropriate balance between education and entertainment.	1
	Ensure the motivational factor of students. Mentality based on the repetition and memorization of concepts and content, without the concern of abstracting the knowledge acquired in solving problems.	3
	Lack of infrastructure and, in some cases, difficulties in applying non-traditional teaching methodologies due to students' cultural issues, awakening in teachers the need to use new technologies.	2

about importance of their self-studies organization and their autonomy, especially when adopted a technique in which teacher acts just presenting a problem and students should acting looking for solutions.

Table 11 shows difficulties presented by teachers. In general, those difficulties were related to use new teaching techniques. It is important to mention that those difficulties are not linked only to students' awareness of the importance of their studies, i. e., the difficulties involved, range from the structures available for teaching to the behavior of the teaching staff facing the *new* need to adopt different methodologies for more effective teaching. Thereby, it is necessary to know the method that can fulfill the needs of educational institutions in which the teacher operates, whether in basic education, high school, undergraduate and among others [60]. Because, within education, it is necessary to debate whether or not learning improves when students are allowed to explore the educational content by themselves or if students must be strictly guided in the topics to be learned. On the other hand, in classroom environments, teachers present students choices because they believe it increases effort and learning [61].

Those result have demonstrated that most researchers have found different benefits in relation to the AM. It is important to mention that not all publications clearly show the items benefits or difficulties. However, they present the results achieved and the difficulties faced in conducting their researches. Thereby, we can infer that the use of different AM positively influences the teaching because, in general way, it addresses different techniques for teaching,, and it addresses new activities and new engagement techniques required to achieve educational goals in learning different CS courses.

Therefore, AM have produced positive results for students in relation to learning and, for teachers, in relation to teaching practices, because, as Michaelsen have pointed [62], it forces students to break with their passivity. Considering the benefits presented in the analyzed publications, some advantages could be: (i) developing students' skills at a high level; (ii) promoting the development of personal and team skills; and (iii) also bringing advantages to the teacher, such as enthusiasm them in the classroom and career continuity as pointed by Guimarães *et al.* [52]. Those results are very

important, because in the 20th century, education is a consequence of a process that involves several thinkers, who discuss teaching models and highlight the need for students' autonomy [18].

Finally, notably, all those researches also confirm a need for reflection about our traditional teaching practices and the adoption of AM applied in teaching and learning course on computing area. Because, according to the words of Vanbecelaere *et al.* [59] ''learning is most effective when it is active, experiential, situated, problem-based and provides immediate feedback''. So, without any doubts, technology has found its way into many classrooms around the world to help educational process to be more effective and enjoyable. Despite the great potential may technology have for facilitating and promoting students learning, teachers are also challenged to not only familiarize themselves with those technologies but to put it in practice as well [63].

V. THREATS TO VALIDITY

We believe that some limitations of this study can be related to publication, selection bias, inaccuracy in data extraction and erroneous classification. For Kitchenham & Charters [20], limitations related to publication bias refers to the problem that positive results are more likely to be published than negatives, because negative results take longer to be published or are mentioned in other publications to a lesser extent. In order to reduce that obstacle, as much as possible, we have done search on symposium, conference proceedings and journals in which the most relevant studies to the Computer and Education area are published frequently: Brazilian Symposium on Informatics in Education, Brazilian Symposium on Games and Digital, Entertainment, Computer Workshop at School, Computer Education Workshop, New Technologies in Education Magazine, Journal of Informatics in Education, International Congress of Educational Informatics and Scopus Library. This last platform is a digital library that is frequently used for researchers for access to relevant journals related to the same subject that we have been studying here. However, we did not consider other sources such as searching on conference bases or workshops that may affect, somehow, the validity of our results.

The bias that refers to the selection of publications is related to distortions that could happened in a statistical analysis due to the criteria used to select publications. In order to mitigate that threat, we have used the inclusion criteria to gather the largest possible number of publications that fit the AM as being applied in the teaching and learning activities of course of computing area, as well as the criteria for the exclusion of articles that did not present the necessary information related to this study. So, we have elaborated a SMS and validated it with other professionals of the areas with recognized experience in the conduction of that kind of work. We also have detected other limitations that could bring some problems to the results of this work and that are related to the systematic of procedure in carrying out this study. Thus, analyzing our main goals in relation to the accomplishment of this SMS, we have decided to categorize the selected publications and to identify representative studies instead of carrying out tests of validations on the results achieved by those authors. Besides that, we also have included on our mapping others specific questions, such as, regarding methodologies, technical or studies used, and this may have affected our results.

VI. CONCLUSION

This paper have presented a SMS that summarizes existing information regarding types of AM being applied in teaching and learning activities in CS courses. From an initial number of 753 papers, a total of 35 were selected for carrying out the mapping study. And the results obtained have allowed us to extract some conclusions regarding the state-of-the-art in the area, to identify several research gaps, and to extract some guidelines for innovative directions in computing education. Moreover, the application of a well-defined review protocol will also allow us to efficiently update and extend the SMS in future years.

As a result, our analyses have revealed or presented the following significant findings: a) an overview of the active methodologies applied to teach in computation that shows a variety of 6 types of AM; b) the AM used per field of computing area in relation to students' perception; c) different types of technical or studies used in research; d) and some benefits and difficulties in relation to the adoption of AM to teaching. According to those findings, we suggest that this review can greatly help and inform about the use of AM in the teaching and learning computing. Our review has showed a variety of AM that have been used and have identified the most common ones. It also has provided an overview of the methods used when validating the corresponding active methodology applied. In recent years, a great number of AM has been used as techniques for teaching in computing. However, in the current existing mapping studies, the perceptions of students have been forgotten. So, we have joined our mapping of both teachers' and students' points of view.

Finally, we expect that AM used in teaching computing in this review could also be effective in solving problems in other areas that share similar characteristics about learning difficulty, practice, and content abstraction students. Thus, it might allow teachers to experience the consequences of different methodologies choices as powerful approach to promote engagement, motivation, empathy, awareness, and constructive behavior for students. Also, we have highlighted the importance of continuing this research, such as those analyzed about mapping, in order to increase and to offer more consistent bibliography, both qualitative and quantitative data. That way, our research could sensibly allows the measurement and validation of the applicability of the AM as a strategic tool for teaching different contents in different course on computing or other areas.

REFERENCES

[1] R. M. Castro and E. S. Siqueira, ''Alternative teaching techniques (active learning) for computer disciplines: A systematic mapping in the Brazil context,'' in *Proc. School Comput. Sci. Workshop*, 2019, vol. 25, no. 1, p. 1409.

[2] P. Parsons, "Preparing computer science graduates for the 21st century," *Teach. Innov. Proj.*, vol. 1, no. 1, pp. 1–90, 2011.
[3] M. Leon, "Quality teaching innovation and improvement of university teaching. Study and analysis project: Teaching innovation," Ministry Educ., Madrid, Spain, Tech. Rep., 2010, pp. 1–23.
[4] C. P. dos Santos and E. S. R. Soares, "Learning and teacher-student relationship at the university: Two sides of the same coin," *Study Aval. Educ.*, vol. 22, no. 49, pp. 353–369, 2011.
[5] J. Moran, "Active methodologies and hybrid models in education," in *New Digital Technologies: Reflections on Mediation, Learning and Development*, S. Yaegashi *et al.*, Eds. Curitiba, Brazil: CRV, 2017, pp. 23–35. Accessed: Jun. 27, 2020. [Online]. Available: http://www2.eca.usp.br/moran/wpcontent/uploads/2018/03/Metodológicas_Ativas.pdf
[6] P. Demo, *Teacher of the Future and Knowledge Reconstruction*. London, U.K.: Voices, 2004.
[7] M. A. de Armanda, R. L. Rodrigues, and E. V. C. Garcia, "A systematic mapping for problem based learning applied to computer science," in *Proc. School Comput. Sci. Workshop*, 2012, vol. 1, no. 1.
[8] A. K. Hartwig, M. Silveira, L. Fronza, M. Mattos, and E. L. P. A. de Kohler, "Active methodologies for the teaching of computing: A systematic review and a practical study," in *Proc. Workshop Inform. School*, Nov. 2019, vol. 25, no. 1, p. 1, doi: 10.5753/cbie.wie.2019.1139.
[9] J. V. Lima, M. de Melo Alves Júnior, A. Moya, R. Almeida, P. Anjos, M. Lencastre, R. A. de Araújo Fagundes Fagundes, and F. lencar, "Active methodologies and software engineering teaching: A systematic literature review," in *Proc. Workshop Inform. School*, Nov. 2019, vol. 25, no. 1, p. 1, doi: 10.5753/cbie.wie.2019.1014.
[10] A. Raes, P. Vanneste, M. Pieters, I. Windey, W. Van Den Noortgate, and F. Depaepe, "Learning and instruction in the hybrid virtual classroom: An investigation of students' engagement and the effect of quizzes," *Comput. Edu.*, vol. 143, Jan. 2020, Art. no. 103682.
[11] F. L. de Oliveira and F. E. I. Nunes, "Active learning in programming teaching: A systematic literature review," in *Proc. Workshops Brazilian Congr. Inform. Educ.*, 2019, vol. 8, no 1, p. 377.
[12] M. A. de F. da Silva and E. M. Oliveira, "Educational robotics in the perspective of active methodologies," in *Proc. School Comput. Sci. Workshop*, 2019, vol. 25, no. 1, p. 1289.
[13] B. Moreno, "Combining agile and active methodologies in teaching introduction to programming to high school students," in *Proc. Workshop Challenges Comput. Appl. Educ.*, 2019, vol. 8, no 1, p. 45.
[14] R. R. Silva, J. Fernandes, and E. R. Santos, "Overview of the use of digital games in teaching programming at the higher education level in the last decade: A systematic literature review," in *Proc. Brazilian Symp. Comput. Educ.*, 2018, vol. 29, no 1, p. 535.
[15] L. Bacich and E. J. Moran, *Active Methodologies for Innovative Education: A Theoretical-Practical Approach*. Coventry, U.K.: Penso Editor, 2018.
[16] F. L. Lovato, A. Michelotti, and E. E. L. da Silva Loreto, "Active learning methodologies: A brief review," *Acta Sci.*, vol. 20, no. 2, pp. 154–171, 2018.
[17] T. R. da Silva, T. Medeiros, H. Medeiros, R. Lopes, and E. Aranha, "Programming teaching-learning: A systematic literature review," *Brazilian J. Inform. School*, vol. 23, no. 1, p. 182, 2015.
[18] B. Kitchenham and E. S. Charters, "Guidelines for performing systematic literature reviews in software engineering, version 2.3," Dept. EBSE, Keele Univ., Univ. Durham, Durham, U.K., Tech. Rep., 2007.
[19] A. Fernandez, E. Insfran, and S. Abrahão, "Usability evaluation methods for the Web: A systematic mapping study," *Inf. Softw. Technol.*, vol. 53, no. 8, pp. 789–817, Aug. 2011.
[20] J. Jesson, L. Matheson, and E F. M. Lacey, *Doing Your Literature Review: Traditional and systematic techniques*. Newbury Park, CA, USA: Sage, 2011.
[21] T. Dyba, T. Dingsoyr, and E. G. K. Hanssen, "Applying systematic reviews to diverse study types: An experience report," in *Proc. 1st Int. Symp. Empirical Softw. Eng. Meas.*, Sep. 2007, pp. 225–234.
[22] S. R. Lambert, "Do MOOCs contribute to student equity and social inclusion? A systematic review 2014–18," *Comput. Edu.*, vol. 145, Feb. 2020, Art. no. 103693.
[23] K. Petersen, S. Vakkalanka, and L. Kuzniarz, "Guidelines for conducting systematic mapping studies in software engineering: An update," *Inf. Softw. Technol.*, vol. 64, pp. 1–18, Aug. 2015.
[24] E. C. Moya, "Using active methodologies: The student-view," *Procedia-Social Behav. Sci.*, vol. 237, pp. 672–677, 2017, doi: 10.1016/j.sbspro.2017.02.040.
[25] F. L. Noguero, *Participatory Methodology in Education Universitaria*, vol. 9. Madrid, Spain: Narcea Ediciones, 2005.
[26] S. V. Silva and E. A. R. Gonçalves, "The simulated practice in teaching project management through the 'integrated management tool," Projects Manage., New Ideas Educ. Inform. TISE, Tech. Rep., 2015, p. 6.
[27] R. A. Bittencourt, C. A. Rodrigues, and E. D. S. S. Cruz, "An integrated object-oriented programming experience, data structures and systems design with PBL," in *Proc. 21st Workshop Comput. Educ.-33rd Congr. Brazilian Computer Soc.* Maceió, Brazil: SBC, 2013, p. 10.
[28] L. F. D. Pereira, F. Fábio Lapolli, F. F. Sampaio, C. L. R. Motta, and C. E. T. Oliveira, "Learning objects workshop: An approach to the teaching of computing in technical courses," *Brazilian J. Inform. School*, vol. 18, no. 3, pp. 4–18, Sep. 2010, doi: 10.5753/RBIE.2010.18.03.04.
[29] D. Ravindranath, T. L. Gay, and M. B. Riba, "Trainees as teachers in team-based learning," *Academic Psychiatry*, vol. 34, no. 4, pp. 294–297, Jul. 2010.
[30] R. H. Steadman, W. C. Coates, Y. M. Huang, R. Matevosian, B. R. Larmon, and L. McCullough, "Simulation-based training is superior to problem-based learning for the acquisition of critical assessment and management skills," *Crit. Care Med.*, vol. 34, no. 1, pp. 151–157, 2006.
[31] M. J. Lage, G. J. Platt, and M. Treglia, "Inverting the classroom: A gateway to creating an inclusive learning environment," *J. Econ. Edu.*, vol. 31, no. 1, pp. 30–43, Jan. 2000.
[32] J. Bergmann and E A. Sams, *Flip Your Classroom: Reach Every Student in Every Class Every Day*. Washington, DC, USA: International Society for Technology in Education, USA, 2012, pp. 15–17.
[33] R. Sayer, E. Marshman, and C. Singh, "Case study evaluating just-in-time teaching and peer instruction using clickers in a quantum mechanics course," *Phys. Rev. Phys. Edu. Res.*, vol. 12, no. 2, Oct. 2016.
[34] E. Mazur, *Peer Instruction: The Active Learning Revolution*. Coventry, U.K.: Penso Editor, 2015.
[35] A. G. P. de Andrade, F. A. C. dos Santos, Jr., J. M. Pimentel, J. C. N. Bittencourt, and E. T. B. de Santana, "Application of the PBL method in software engineering teaching: Student's view," in *Proc. ERBASE, SBC*, Brazil, 2010, pp. 1–10.
[36] J. W. Thomas, "A review of research on project-based learning," Autodesk Found., San Rafael, CA, USA, Tech. Rep., 2000, pp. 230–243.
[37] J. R. Mergendoller and E. J. W. Thomas. (2001). *Managing Project-Based Learning: Principles From the Field*. Buck Institute for Education. [Online]. Available: httpwwwBieOrg
[38] S. Deterding, R. Khaled, L. E. Nacke, and E. D. Dixon, "Gamification: Toward a definition," in *Proc. CHI Gamification Workshop*, Vancouver BC, Canada, vol. 12, 2011.
[39] L. A. Ribeiro, T. L. da Silva, and E. A. Q. Mussi, "Gamification: A methodology to motivate engagement and participation in a higher education environment," *Int. J. Educ. Res.*, vol. 6, no. 4, pp. 249–264, 2018.
[40] G. Kiryakova, N. Angelova, and E. L. Yordanova, "Gamification in education," in *Proc. 9th Int. Balkan Educ. Sci. Conf.*, 2014, pp. 1–5.
[41] A. C. T. Klock, I. Gasparini, and E. M. S. Pimenta, "Designing, developing and evaluating gamification: An overview and conceptual approach," in *Data Analytics Approaches in Educational Games and Gamification Systems*, A. Tlili and E M. Chang, Singapore: Springer, 2019, pp. 227–246.
[42] J. Morán, "Changing education with active methodologies," *Coleç. Mídias Contemp. Media Convergences Educ. Cid. Youth Approaches*, vol. 2, no. 1, pp. 15–33, 2015.
[43] A. Diesel, A. L. S. Baldez, and E. S. N. Martins, "The principles of active teaching methodologies: A theoretical approach," *Rev. Thema*, vol. 14, no. 1, pp. 268–288, 2017.
[44] R. L. Moffitt, C. Padgett, and R. Grieve, "Accessibility and emotionality of online assessment feedback: Using emoticons to enhance student perceptions of marker competence and warmth," *Comput. Edu.*, vol. 143, Jan. 2020, Art. no. 103654.
[45] E. F. Barbosa and E. D. G. de Moura, "Active learning methodologies in Professional and Technological Education," *Bol. Tec. Senac*, vol. 39, no. 2, p. 2, 2013.
[46] R. Pereira, "Active method: Questioning techniques of reality applied to basic education and higher education," in *Proc. 6th Int. Colloq. 'Educ. Contemp.*, São Cristovão, Brazil, 2012, pp. 1–15.
[47] D. Woods, *Problem Based Learning: How to Get the Most From PBL*. Hamilton, ON, Canada: McMaster Univ., 1994.
[48] V. V. Lima, "Constructivist spiral: An active teaching-learning methodology," *Interface-Comun. Health Educ.*, vol. 21, pp. 421–434, Oct. 2016.
[49] A. Corrêa, D. Zuasnabar, M. Santibanez, S. Silva, L. Prestes, and E. P. F. da Silva, "The use of mobile learning and active methodologies in the educational context," in *Proc. Workshops Brazilian Congr. Inform. Educ.*, 2019, vol. 8, no. 1, p. 730.
[50] A. B. de Machado, "Active methodologies, integral knowledge," in *Innovative Practices in Active Methodologies*, S. R. Dias and A. N. Volpato, Eds. Florianópolis, Brazil: Context Digital, 2017.

[51] D. B. Felippe, A. N. Volpato, I. Araldi, and E. S. R. Dias, *Innovative Practices in Active Methodologies*, S. R. Dias and A. N. Volpato, Eds. Florianópolis, Brazil: Context Digital, 2017, p. 179.
[52] F. Guimarães, M. Leite, F. Reinaldo, and E. G. Ito, "Active teaching methods combined with technology for teaching practice: An experience report," in *Proc. Workshop Inform. School*, 2018, vol. 24, no. 1, p. 1, doi: 10.5753/cbie.wie.2018.333.
[53] A. Prado, *Understanding the 21st Century Student and How to Teach This New Generation*. São Paulo, Brazil: Geekie, 2015.
[54] E. Oliveira and R. Barros, "Gaia ABstração game: Proposal of a game to mediate the teaching-learning process of the object-oriented paradigm," in *Proc. 17th Symp. Bras. Games E Entertainment Digit.*, 2013, pp. 533–540.
[55] P. L. Torres and E. A. F. Irala, "Collaborative learning," in *Some to Entertain Thinking and Act*, P. L. Torres, Ed. Curitiba, Brazil: SENAR, 2007, pp. 65–98.
[56] J. H. Helm e L. G. Katz, *Young Investigators: The Project Approach in the Early Years*. New York, NY, USA: Teachers College Press, 2016.
[57] L. de Almeida, M. P. Rolim, R. da Silva, and E. A. Costa, "E-PBL?: Support tool for learning and using the problem based learning methodology," in *Proc. Workshop Inform. School*, Nov. 2019, vol. 25, no. 1, p. 1, doi: 10.5753/cbie.wie.2019.1399.
[58] J. B. S. Júnior, A. Kronbauer, and E. J. Campos, "Assessing users' experience with active methodologies in Human-Computer Interaction classes," in *Proc. Brazilian Symp. Inform. School*, Nov. 2019, vol. 30, no. 1, p. 1, doi: 10.5753/cbie.sbie.2019.1201.
[59] S. Vanbecelaere, K. Van den Berghe, F. Cornillie, D. Sasanguie, B. Reynvoet, and F. Depaepe, "The effects of two digital educational games on cognitive and non-cognitive math and reading outcomes," *Comput. Edu.*, vol. 143, Jan. 2020, Art. no. 103680.
[60] T. S. Borges and G. Alencar, "Active methodologies in the promotion of critical student education: The use of active methodologies as a didactic resource in the critical education of higher education students," *Cairu Rev.*, vol. 3, no. 4, pp. 119–143, 2014.
[61] C. Ferguson, E. L. van den Broek, and H. van Oostendorp, "On the role of interaction mode and story structure in virtual reality serious games," *Comput. Edu.*, vol. 143, Jan. 2020, Art. no. 103671.
[62] L. K. Michaelsen, "Getting started with team learning," in *Team Learn. Transform. Use Small Groups Westport*. Westport, CT, USA: Greenwood, 2002.
[63] R. Scherer, F. Siddiq, and J. Tondeur, "All the same or different? Revisiting measures of teachers' technology acceptance," *Comput. Edu.*, vol. 143, Jan. 2020, Art. no. 103656.
[64] T. S. C. da Silva, J. C. B. de Melo, and E. P. C. A. R. de Tedesco, "A model to promote student engagement in programming learning using gamification," *Brazilian J. Inform. School*, vol. 26, no. 3, p. 3, 2018.
[65] E. Pantaleão, L. R. Amaral, and E. G. B. E. Silva, "An approach based on the Robocode environment for teaching high school programming," *Brazilian J. Inform. School*, vol. 25, no. 3, p. 3, 2017, doi: 10.5753/rbie.2017.25.03.95.
[66] M. C. Cera, M. H. D. Forno, and E. V. G. Vieira, "A Proposal for Teaching Software Engineering from Problem Solving," *Brazilian J. Inform. School*, vol. 20, no. 3, p. 3, Dec. 2012, doi: 10.5753/rbie.2012.20.03.116.
[67] S. V. Silva and A. R. Gonçalves, "The simulated practice in teaching project management through the integrated management tool," Project Manage., New Ideas Educ. Inform. TISE, Tech. Rep., 2015, vol. 6.
[68] A. F. F. Costa, A. F. M. F. de Melo, G. G. Moreira, M. A. de Carvalho, and E. M. V. A. de Lima, "Application of inverted room and gamification elements to improve teaching-learning in object oriented programming," Project Manage., New Ideas Educ. Inform. TISE, Tech. Rep., 2017, vol. 13, pp. 223–232.
[69] W. Nagai, C. Izeki, and E. R. Dias, *Experience in the Use of Gamified Online Tools in the Introduction to Computer Programming*, vol. 22, no. 1, 2016, p. 301.
[70] M. G. de Oliveira, A. Neves, M. F. S. Lopes, H. F. Medeiros, M. B. Andrade, and E. L. L. Reblin, "A distance programming course with active methodologies and learning analysis by software metrics," *Rev. New Technol. Educ.*, vol. 15, no. 1, pp. 1–10, 2017.
[71] R. I. Casarotto, G. Bernardi, A. Z. Cordenonsi, and E. R. D. Medina, "Logirunner: A board game as a tool to aid the teaching and learning of algorithms and programming logic," *RENOTE-Rev. News Tecnol. Educ.*, vol. 16, no. 1, pp. 1–10, 2018.
[72] R. I. Casarotto, G. Bernardi, A. Z. Cordenonsi, and R. D. Medina, "The use of unplugged computing in a gamification context for teaching data structures," *Renote-Rev. News Tecnol. Educ.*, vol. 16, no. 2, pp. 546–555, 2018.
[73] J. A. Moreira and W. M. Monteiro, "Inverted classroom—An experience in teaching-learning programming for computer network administration," *Rev. New Technol. Educ.*, vol. 16, no. 1, pp. 1–11, 2018.
[74] M. E. C. Natal, B. A. Barbosa, J. C. Hernandes, B. de Sousa Much, M. Bigolin, S. J. R. da Silva, C. B. Silva, and L. F. B. de Carvalho, "Trilogic: A gamified environment as an aid tool for teaching programming logic learning," *Rev. New Technol. Educ.*, vol. 16, no. 2, pp. 41–50, 2018.
[75] D. S. de Souza Rabelo *et al.*, "Development of computer systems using problem-based learning," in *Proc. Brazilian Symp. Comput. Educ.*, 2018, vol. 29, no. 1, pp. 188–197, doi: 10.5753/cbie.sbie.2018.188.
[76] F. I. R. Pessoa, A. L. S. O. Araujo, W. Andrade, and E. D. Guerrero, "T-mind: A gamified application to encourage the development of computational thinking skills," in *Proc. Brazilian Symp. Comput. Educ.*, 2017, vol. 28, no. 1, p. 645.
[77] M. A. de Azevêdo Silva and A. Dantas, *KLouro: An Educational Game to Motivate Students Who Are New to Programming*, vol. 25, no. 1, 2014, p. 702.
[78] E. H. S. Raposo and V. Dantas, "The snake challenge-using gamification to motivate students in an introductory programming discipline," in *Proc. Brazilian Symp. Comput. Educ.*, 2016, vol. 27, no. 1, p. 577.
[79] M. C. Meireles and B. Bonifácio, "Use of agile methods and problem-based learning in software engineering teaching: An experience report," in *Proc. Brazilian Symp. Comput. Educ.*, 2015, vol. 26, no. 1, p. 180.
[80] R. F. de Azevedo and B. C. de Paula, "Proposed methodology for learning computer programming through the recontextualization of serious games in the style game & watch," in *Proc. SBC SBGames*, 2011, pp. 1–9.
[81] R. T. Figueiredo and C. Figueiredo, "Wargrafos- game to aid in learning the discipline of graph theory," in *Proc. 12th Symp. Bras. Games Entertainment Digit. SBGames*, 2011, pp. 1–9.
[82] E. D. de Oliveira and R. M. de Barros, "Gaia ABstração game: Proposal for a game to mediate the teaching-learning process of the object-oriented paradigm," in *Proc. 17th Brazilian Symp. Games Digit. Entertainment*, São Paulo, Brazil, 2013, pp. 533–540.
[83] S. A. Melo, "Game of code: Applying gamification in programming disciplines," in *Proc. SBGames XV SBGames*, São Paulo, Brazil, Sep. 2016, pp. 1241–1244.
[84] J. Mattar *et al.*, "Gamification and games for scientific methodology: Proposed board game and game," in *Proc. SBGames XVI SBGames*, Curitiba, Brazil, Nov. 2017, pp. 757–763.
[85] S. Melo and C. d S. S. Neto, "Game of code: Development and evaluation of a gamified activity for programming disciplines," presented at the em 16th Symp. Brazilian Games Entertainment Digit. (SBGames), 2017.
[86] E. D. de Oliveira, L. de Souza Mendes, M. C. Camargo, M. H. S. Senegalha, and R. M. de Barros, "Gaia abstraction game BD: A game that aims to help the teaching-learning process of entify-relationship concepts and relational model," in *Proc. SBGames, XVI SBGames*, Curitiba, Brazil, Nov. 2017, pp. 27–33.
[87] J. Mombach *et al.*, "POOkemon: A game about object-oriented programming," in *Proc. SBGames*, 2018, p. 5.

MARIA IVANILSE CALDERON RIBEIRO is currently pursuing the Ph.D. degree in informatics with the Federal University of Amazonas (UFAM). Her research interests include software engineering education, active learning strategies, and related topics. She is currently a Professor of basic, technical and technological education with the Federal Institute of Education, Science and Technology of Rondônia (IFRO)/Campus Porto Velho North Zone.

ODETTE MESTRINHO PASSOS received the Licentiate and bachelor's degrees in mathematics, the master's degree in informatics, and the Ph.D. degree in informatics from the Federal University of Amazonas, in 1995, 2003, and 2014, respectively. She is currently an adjunct Professor with the Federal University of Amazonas / Institute of Exact Sciences and Technology. She has experience in computer science, working mainly in informatics in education, software process improvement, and experimental software engineering.

Appendix B

Um Mapeamento Sistemático da Literatura sobre o uso de Metodologias Ativas durante o Ensino de Programação no Brasil

Ivanilse Calderon[1,3], Williamson Silva[2], Eduardo Feitosa[1]

[1]Instituto de Computação (IComp) – Universidade Federal do Amazonas (UFAM)
Manaus, AM – Brasil

[2]Laboratory of Empirical Studies in Software Engineering (LESSE) - Departamento de Engenharia de Software - Universidade Federal do Pampa (UNIPAMPA)
- Alegrete, RS - Brasil

[3]Instituto Federal de Educação, Ciência e Tecnologia de Rondônia (IFRO)
Campus Porto Velho Zona Norte - Porto Velho, RO - Brasil

{[1,3]ivanilse.calderon,[1]efeitosa}@icomp.ufam.edu.br

[2]williamsonsilva@unipampa.edu.br

Abstract. *Teaching programming is a challenge for instructors of Computer Science courses. Instructors have been adopting Active Learning Methodologies (ALMs) into teaching practices of computer programming to minimize the challenges faced in the classroom. In this sense, we performed a systematic literature mapping to summarize the main ALMs adopted by instructors during teaching programming in undergraduate courses in the Brazilian context. We identified main ALMs, and Educational Games and Gamification are the most adopted by Brazilian instructors. We also identified students' perceptions about the use of these ALMs in the classroom.*

Resumo. *O ensino de programação é um desafio para os docentes dos cursos de Computação. Como forma de tentar minimizar os desafios enfrentados em sala de aula, os docentes vem adotando as Metodologias Ativas (MAs) em suas práticas de ensino e aprendizagem de programação de computadores. Nesse sentido, conduziu-se um Mapeamento Sistemático da Literatura para sumarizar as principais MAs adotadas pelos docentes durante o ensino de programação nos cursos de graduação no cenário brasileiro. A partir dos resultados, foram identificadas dez tipos de MAs, sendo Jogos Educacionais e Gamificação as mais adotadas pelos docentes brasileiros. Foram identificadas também as percepções dos estudantes sobre o uso destas MAs em sala de aula.*

1. Introdução

O processo de aprendizagem em disciplinas de programação de computadores, em cursos da área de Computação (conhecidas na literatura como CS1 e CS2), é uma atividade complexa e difícil [Luxton-Reilly et al. 2018]. Estas disciplinas requerem que os estudantes desenvolvam, ao longo da aprendizagem, diferentes habilidades como alta capacidade cognitiva de abstração dos problemas, resolução de problemas, raciocínio e pensamento

DOI: 10.5753/sbie.2021.217564

lógico [Raj et al. 2018]. Para que os estudantes desenvolvam tais habilidades, os docentes precisam adotar novas estratégias de ensino e obter uma postura mais transformadora em sala de aula, proporcionando um ambiente de aprendizagem engajador para os estudantes [Acharya and Gayana 2021, Silva et al. 2019].

Diante deste cenário, as Metodologias Ativas (MAs) vêm ganhando destaque entre os docentes [Ribeiro et al. 2021]. Segundo Diesel et al. (2017), as MAs possibilitam uma mudança no paradigma de aprendizagem, onde o estudante sai do papel de agente passivo (apenas escuta e recebe o conteúdo que é transmitido pelo docente) e passa para o papel de agente ativo da aprendizagem, tornando-se o principal responsável por sua aprendizagem. As MAs são estratégias de ensino centradas na participação efetiva dos estudantes e, por conta disso, auxiliam na construção do processo de aprendizagem de uma maneira flexível, interligada e híbrida [Bacich and Moran 2018]. Sob esta perspectiva, é importante refletir sobre à adoção de MAs durante o ensino de programação frente às estratégias de ensino tradicionais que continuam centradas no docente.

Nesse sentido, este artigo tem por objetivo sumarizar as principais MAs adotadas pelos docentes durante o ensino de programação de computadores em cursos de graduação no Brasil. Para isso, foi conduzido um Mapeamento Sistemático da Literatura (MSL) nos principais eventos e periódicos relacionados à Informática na Educação e Educação em Computação no Brasil, entre os anos de 2010 e 2021. Os resultados alcançados mostram um panorama sobre a aplicação das MAs no ensino de programação e as percepções dos estudantes, com relação às MAs, enquanto aprendem conteúdos relacionados à programação de computadores.

2. Método de Pesquisa

Um MSL [Kitchenham and Charters 2007] foi realizado para identificar o panorama sobre as MAs adotadas para o ensino de programação de computadores na educação superior no Brasil. Os procedimentos utilizados serão detalhados nas subseções a seguir.

2.1. Questões de Pesquisa

Para guiar este trabalho definiu-se as seguintes Questões de Pesquisa (QP):

- **QP1**: Quais as metodologias ativas que são comumente adotadas por docentes durante o ensino e aprendizagem de programação de computadores?
- **QP2**: Como os estudantes percebem as metodologias ativas durante a aprendizagem de programação?

2.2. Estratégia de Busca

Este MSL se propõe a investigar as MAs adotadas nos cursos de graduação do Brasil. Esta pesquisa foi realizada manualmente nos seguintes eventos e revistas científicas de Educação em Computação e Informática na Educação no Brasil: Simpósio Brasileiro de Informática na Educação (SBIE), Workshop de Informática na Escola (WIE), Workshop de Educação em Computação (WEI), Simpósio Brasileiro de Jogos e Entretenimento Digital (SBGames), Congresso Internacional de Informática Educativa (TISE), Revista Novas Tecnologias na Educação (RENOTE) e Revista Brasileira de Informática na Educação (RBIE). Considerando que a Sociedade Brasileira de Computação vem trabalhando, desde 2010, na reformulação do Currículo de Referência dos cursos de

Computação [Castro and Siqueira 2019], neste trabalho considerou-se o período de 2010 até 2021, para conhecer o cenário nacional em relação à utilização das MAs na última década. Este MSL foi conduzido até maio de 2021.

2.3. Seleção das publicações

Para a seleção das publicações identificadas, os seguintes Critérios de Inclusão (CI) foram definidos: **CI1**, devem ser selecionadas publicações que apresentam MAs e/ou as percepções dos docentes/estudantes sobre as MAs adotadas durante o ensino de programação; e **CI2**, devem ser selecionadas publicações que apresentam estudos experimentais sobre o uso das MAs durante o ensino de programação. Também foram definidos os Critérios de Exclusão (CE): **CE1**, publicações não disponíveis para a leitura e coleta dos dados (publicações pagas, por exemplo); **CE2**, publicações que não atendam os critérios de inclusão; **CE3**, publicações que não estejam nos idiomas Português ou Inglês; **CE4**, publicações duplicadas.

Na primeira etapa do processo de seleção (**1º Filtro**), realizou-se a leitura do título, palavras-chave e resumo de cada publicação. Neste momento, foram selecionadas as publicações que atendessem pelo menos um dos critérios de inclusão. Em caso de dúvida, a publicação era incluída para uma análise posterior. Na segunda etapa (**2º Filtro**), realizou-se a leitura completa das publicações selecionadas no 1º Filtro utilizando os critérios de inclusão e exclusão para decidir se publicação seria selecionada ou não.

2.4. Extração dos dados

Nesta fase, foram extraídos os dados categorizados da seguinte forma: informações gerais (título, autores, ano, tipo e o local de publicação), metodologia (tipos de MAs identificadas nas publicações), trechos das publicações que apresentavam às percepções dos estudantes sobre as MAs, nome das disciplinas, linguagens de programação mencionadas e informações sobre as ferramentas utilizadas em conjunto com as MAs.

3. Resultados Obtidos

A Tabela 1 apresenta a quantidade de publicações retornadas por evento e revista (segunda coluna), bem como a quantidade de artigos selecionados no 1º e no 2º Filtro. Um total de 21 publicações foram selecionadas com base nos critérios descritos na Subseção 2.3.

Tabela 1. Quantitativo das publicações retornadas.

Bibliotecas Digitais	Publicações	1º Filtro	2º Filtro
SBGames	315	16	7
SBIE	305	14	4
RENOTE	265	5	2
RBIE	336	14	1
TISE	21	2	1
WEI	88	19	4
WIE	177	12	2
Total	**753**	**220**	**21**

Os resultados mostram que houve um aumento na quantidade de publicações em 2018 e 2019 (Figura 1). No período de 2011 a 2017 observa-se uma oscilação no quantitativo de publicações (entre uma e três publicações por ano). Nos anos de 2012 e 2015 não foram identificadas publicações no contexto deste MSL. Percebe-se que há

uma quantidade significativa de publicações sobre o uso de MAs para a aprendizagem de programação. Logo, acredita-se que a comunidade está constantemente pesquisando e publicando sobre a adoção de MAs para apoiar a prática docente.

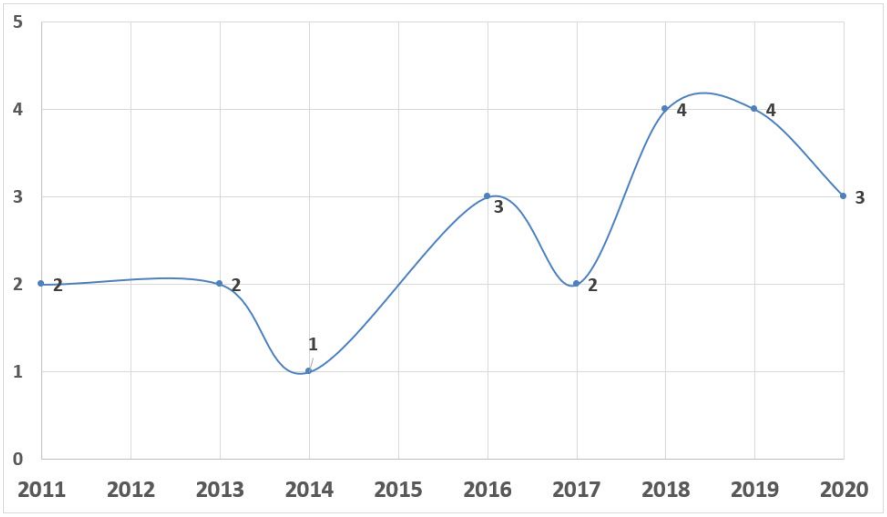


Figura 1. Tendência de publicação por ano.

3.1. QP1 - Quais as MAs que são comumente adotadas pelos docentes durante o ensino e aprendizagem de programação de computadores?

Em relação a QP1, a Figura 2 apresenta uma visão geral das MAs identificadas no âmbito deste MSL. Mais detalhes sobre os resultados das MAs podem ser encontradas no relatório técnico (https://figshare.com/s/21338d322bcfed888be5).

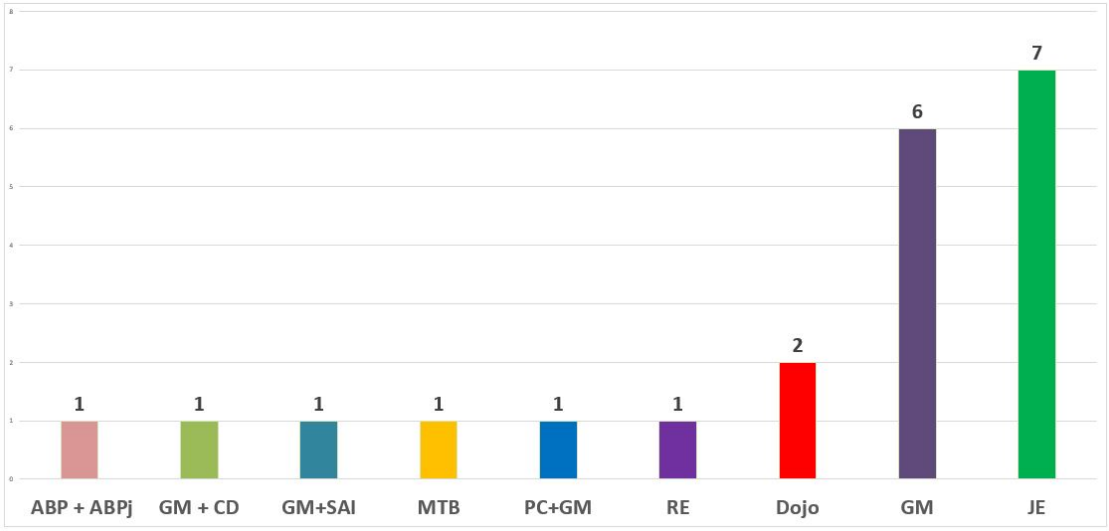


Figura 2. Tipos de Metodologias ativas identificadas nas publicações.

Ao todo, foram identificadas dez (10) tipos de MAs adotadas pelos docentes, são elas: Aprendizagem Baseada em Problemas (ABP); Aprendizagem Baseada em Projetos (ABPj); Coding Dojo (Dojo); Computação Desplugada (CD); Sala de Aula Inve-

tida (SAI); Gamificação (GM); Jogos Educacionais (JE), Método Baseado em Tutoriais (MBT), Programação Competitiva (PC) e Robótica Educacional (RE).

Na Figura 2 percebe-se que Jogos Educacionais é a metodologia com maior quantidade de publicações disponíveis (7) e, consequentemente, a mais adotada pelos docentes. Destaca-se que, no contexto dos JEs, existe uma associação automática dessa MA com jogos digitais. Stephan et al. (2020) apresentam o GameProgJP, uma abordagem que visa apoiar o ensino de programação empregando como recurso pedagógico o desenvolvimento de jogos. O jogo é divido em quatro partes, sendo que cada parte abordava determinado conteúdo da disciplina: 1ª parte, declaração de variáveis e uso de funções; 2ª parte, estruturas condicionais; 3ª parte, estruturas de repetição, vetores e strings; 4ª parte, matrizes e estruturas de dados heterogêneas. Ao final são geradas quatro versões do jogo. As mecânicas do jogo são implementadas de acordo com o conteúdo ensinado ao longo da disciplina [Stephan et al. 2020]. Como resultados, os autores notaram que a abordagem motivou e aumentou o interesse dos estudantes por programação.

No que diz respeito à Gamificação (GM), essa MA vem sendo utilizada pelos docentes pois torna a prática das disciplinas de programação uma atividade leve, divertida e descontraída para os estudantes [Casarotto et al. 2018]. Para auxiliar neste processo, os docentes têm adotado ferramentas e atividades gamificadas com foco na resolução de problemas, como o Kahoot e Socrative [Nagai et al. 2016], o Portal URI Online Judge [Brito et al. 2019] e o UVa Online Judge [Melo et al. 2016]. Outro ponto a ser destacado é que a GM normalmente vem sendo adotada em conjunto com outras MAs. Costa et al. (2017) relatam uma experiência sobre uso de elementos de gamificação combinados com as metodologia SAI para ensinar os conteúdos de Programação Orientada à Objetos. Como resultados, os autores comentam que à adoção combinada destas MAs permitiu a melhoria do rendimento acadêmico e do engajamento dos estudantes nas aulas.

Foram identificadas também as disciplinas de programação em que as MAs foram adotadas, são elas: Algoritmos (AL), Algoritmos II (ALII), Estruturas de Dados (ED), Introdução à Programação (IP), Laboratório de Programação (LProg), Linguagem de Programação (LP), Lógica de Programação (LogProg), Programação Orientada à Objetos (POO), Programação (Prog), Programação Web (ProgWeb) e Teoria dos Gráfos (TG). Observou-se que a linguagem de programação C foi a mais utilizada pelos docentes para ensinar programação, seguida pela linguagem Java e Python. A Tabela 2 apresenta um resumo das MAs encontradas, as disciplinas ao qual foram adotadas e, quando houver, as ferramentas utilizadas pelos docentes para apoiar na adoção das MAs. [1]

Tabela 2. MAs utilizadas por disciplina e ferramentas utilizadas.

MAs	Nome das Disciplinas e Ferramentas Adotadas
JE	LogProg e AL (Jogo Logirunner), IP (Jogo Klouro), POO (Jogo Gaia ABstração Game, POOkemon), Prog (Jogos Bullfrogs, Metrocity, Carcassonne e o jogo Um império em oito minutos) e LProg (GameProgJF)
GM	POO (cod[edu]), Prog (Kahoot, code.Org, Socrative), IP (Canvas e Huxley), ALII (on-line UVa Online Judge e o Jogo Game of Code (Goc))
Dojo	AL, IP, LP (IDE DevC++)
GM + SAI	POO (Moodle)
MBT	Prog (FlashPunk)
PC + GM	AL, ED (URI online Judge)
RE	Prog (Code Blocks e Arduíno)

[1]Para mais detalhes, ver o relatório técnico deste MSL (https://figshare.com/s/21338d322bcfed888be5).

Em se tratando de ensino de programação, o uso de ferramentas para a execução de tarefas é indispensável. Diante disso, é fato que utilizar as MAs com o suporte ferramental de ferramentas (UVa Online Judge, Kahoot, code.Org e Socrative, por exemplo) contribui para atrair a atenção do estudante em diversas ocasiões nas aulas. Além disso, apoia no engajamento e motivação dos estudantes, em especial nos momentos iniciais do ensino, visto que é imprescindível para o processo de ensino e aprendizagem. Desta forma, nota-se que a utilização de ferramentas computacionais, aliadas as MAs, contribui para que a construção e compartilhamento do conhecimento dos estudantes. O uso de ferramentas poderá ser uma contribuição pedagógica quando na transposição didática no ensino frente ao perfil dos estudantes na contemporaneidade. Porém, ressalta-se que nem sempre o fato de utilizar alguma ferramenta garante que os objetivos de aprendizagem sejam alcançados. Isso dependerá, muitas vezes, do perfil do estudante, do contexto, da disciplina e do conteúdo em que a ferramenta foi aplicada [Blatt et al. 2017, Silva et al. 2019].

3.2. QP2 - Como os estudantes percebem as MAs durante a aprendizagem de programação?

Para responder a QP2, realizou-se uma análise qualitativa das percepções dos estudantes coletadas a partir de cada publicação aceita, em relação as MAs adotadas. O objetivo principal desta análise foi identificar os pontos positivos e negativos relatados na literatura, sob a ótica dos estudantes, em relação às MAs. Para realizar a análise qualitativa, foi criada uma lista com todas as percepções identificadas. Cada uma das percepções foi analisada e, a partir disso, criaram-se códigos. Em seguida, estes códigos foram analisados e agrupados de acordo com as suas características, formando conceitos relevantes e que são representados neste trabalho por meio de Categorias e Subcategorias. Ressalta-se que um pesquisador-autor realizou a análise. Em seguida, ela foi revisada e discutida com outro pesquisador-autor, que possui mais de seis anos de experiência em análise qualitativa. A Tabela 3 mostra as percepções dos estudantes agrupadas de acordo com as Categorias identificadas: **Engajamento**, **Desempenho**, **Interação** e **Colaboração**, e **Motivação**.

Tabela 3. Percepções dos Estudantes em relação às MAs.

Cat.	Subcategorias	ABP	ABPj	CD	Dojo	PC	GM	SAI	JE	MBT	RE	#artigos
Engajamento	(+) Engajados para aprender os conteúdos ensinados	X		X			X		X			06
	(+) Engajados devido ao trabalho em equipe	X			X	X						01
	(+) Engajados devido aos momentos de discussão e de tirar dúvidas durante a aula						X		X			02
Desempenho	(+) Melhoria no desempenho na disciplina			X		X	X	X				03
	(+) Melhoria do desenvolvimento de competências profissionais	X	X		X	X						02
	(+) Melhoria nas habilidades de programação					X	X	X				01
	(+) Contribuiu para o entendimento dos conceitos ensinados				X				X			04
Interação e colaboração	(+) Melhoria da participação em sala de aula	X			X	X	X					03
	(+) Incentivou a colaboração e interação entre estudantes e estudantes e docentes para resolução dos exercícios	X			X	X	X					02
Motivação	(+) Motivou os estudantes aprenderam de forma divertida a disciplina				X	X	X		X	X	X	04
	(+) Motivou os estudantes a continuarem frequentando a disciplina						X			X		01
	(+) Motivou os estudantes a se sentiram envolvidos no aprendizado				X	X	X		X		X	04
	(+) Possibilitou os estudantes buscarem diferentes formas de resolver os problemas				X		X		X			03

Para a categoria "**Engajamento**" foram associadas três subcategorias que retratam os motivos pelos quais os estudantes se sentiram mais engajados para aprender programação com aquelas MAs. A percepção em relação ao engajamento foi identificada quando os docentes empregaram as seguintes MAs: ABP, CD, Dojo, GM e JE. Observa-se que estas MAs contribuem para o despertar dos estudantes para uma postura ativa, criativa e colaborativa, visto que se mostram engajados no trabalho em equipe e para discutir as questões durante a aula, buscando tirar dúvidas.

Na categoria "**Desempenho**", as seguintes MAs se destacaram: ABP, ABPj, CD, Dojo, PC, GM, SAI e JE. Nota-se que todas elas estão relacionadas à melhoria do desempenho dos estudantes e algumas mais especificamente ao desenvolvimento de competências profissionais, como comunicação mais ampla, trabalho em equipe e autodidatismo. Além disso, nota-se uma discussão sobre a melhoria das habilidades de programação, como a capacidade para resolução de problemas, entendimento do funcionamento básico da linguagem de programação e a capacidade de leitura de código [Nagai et al. 2016], além de contribuir para o entendimento dos conceitos ensinados. Deste modo, os estudantes conseguem desenvolver habilidades relacionadas aos conhecimentos e práticas da programação, construindo competências profissionais.

Partindo do pressuposto que a construção do conhecimento perpassa pela troca de experiências e compartilhamento do conhecimento construído, observou-se que as MAs ABP, Dojo, PC e GM foram as que mais contribuíram para o despertar da "**Interação e Colaboração**" entre os estudantes e entre os estudantes e docentes. Isto ocorre, pois estas MAs permitem a melhoria da participação em sala de aula e troca de conhecimentos por meio da interação nas discussões. Por fim, observa-se que a categoria "**Motivação**" esteve associada a forma divertida adotada para ensinar o conteúdo (MAs CD, Dojo, PC, GM, JE, MBT e RE). A motivação dos estudantes refletiu numa melhoria na frequência com que os estudantes participavam das aulas, uma vez que eram desafiados a buscarem formas inovadoras de resolver os problemas, dentro e fora da sala de aula.

Em relação as percepções negativas das MAS, tais percepções estavam relacionadas a baixa compreensão dos estudantes sobre os conteúdos das disciplinas e não sobre às MAs em si, tais como [Silva et al. 2018, Moreira and Monteiro 2018]: dificuldade de compreender o funcionamento das estruturas de controle, dificuldade de criar algoritmos que resolvam problemas concretos; dificuldade em aprender a pensar algoritmicamente, dificuldades de se posicionar em frente aos colegas e docentes para expor o seu trabalho e responder a questionamentos. Também houveram aspectos negativos relacionados a programação e ao funcionamento do computador, e dificuldades sobre questões técnicas, por exemplo, a manipulação de arquivos em Java [Cera et al. 2012], falta de habilidades necessárias, como a resolução de problemas [Nagai et al. 2016] ou mesmo a falta de motivação dos estudantes e sua dificuldade para manter um ritmo de estudos contínuo [Raposo and Dantas 2016]. Acredita-se que as percepções identificadas e mapeadas podem subsidiar a construção de conhecimentos para a escolha das MAs a serem utilizadas, frente ao objetivo de aprendizagem a ser alcançado. Além disso, o interesse em conhecer a percepção dos estudantes é importante, pois possibilita, por exemplo, saber se alguns estudantes precisam de ajuda adicional para compreender os conteúdos debatidos em sala de aula ou mesmo para promover (novos) meios para ajudar nas dificuldades apresentadas.

4. Discussão dos Resultados

Os resultados apresentados neste trabalho mostram que há uma preocupação relacionada à adoção / aplicação de diferentes MAs durante o ensino de programação, haja vista que vem sendo exigido cada vez mais do corpo docente a preparação de aulas dinâmicas e o domínio de como utilizar as diferentes MAs para motivar e engajar os estudantes. Percebe-se também um número crescente de MAs que vêm ganhando destaque, no sentido de chamar a atenção dos estudantes e mudando a forma tradicional de aprendizagem de programação [Castro and Siqueira 2019]. Frente a isso, as pesquisas identificadas neste MSL mostram a importância de se explorar esse tema e a necessidade de desenvolver mais estudos para um melhor entendimento de como e quando utilizar MAs no ensino da programação nos cursos da área da Computação e frente as diferentes necessidades dos envolvidos no processo de ensino e aprendizagem no cenário educacional.

Além disso, esse panorama demonstra que foram encontrados mais benefícios do que dificuldades, em especial na perspectiva do estudante, em relação a adoção das MAs nas aulas de programação. Portanto, conclui-se que a adoção de MAs, de maneira geral, vem produzido resultados de aprendizagem positivos para os estudantes, uma vez que de certa forma as MAs fazem com que os estudantes rompam com sua passividade em sala de aula [Mulcahy 2002]. Desta forma, é necessário apoiar e capacitar o corpo docente para que possam atender às necessidades do ensino [Borges and Alencar 2014].

Ao observar a literatura, identificou-se os trabalhos de Borges et al. (2018) e Medeiros et al. (2018) . Contudo, estes trabalhos olham para a literatura sob a ótica de estratégias de ensino (tradicionais ou não) que podem ser adotadas durante o ensino de programação, diferentemente deste MSL, que foca apenas nas MAs. O trabalho de Berssanette e Francisco (2021) apresenta os resultados de MSL em que foram identificas estratégias pedagógicas adotadas no ensino de programação, similarmente ao conduzido neste MSL. Constatou-se que apesar do estudo ter sido executado apenas no contexto brasileiro e com foco no ensino superior, os resultados encontrados neste MSL estão alinhados com os resultados de Berssanette e Francisco (2021) .

5. Considerações Finais

O foco desta pesquisa é sumarizar as principais MAs utilizadas no ensino de programação em cursos superiores no Brasil, além de relatar as percepções dos estudantes sobre o processo de aprendizagem com estas MAs. Para isso, foi conduzido um MSL nos principais eventos e revistas científicos do Brasil. A partir dos resultados alcançados, foram identificadas que dez metodologias ativas são comumente adotadas pelos docentes.

As MAs que mais se destacaram foram JE e GM, estas MAs podem fornecer um feedback rápido, já que muitas vezes são apoiadas por ferramentas tecnológicas e que visam facilitar o processo de aprendizagem de conceitos associadas à prática das teorias aprendidas e a colaboração entre os estudantes. Também, tais ferramentas fornecem meios para que as práticas para a aprendizagem possam ser realizadas em equipe, o que desperta a curiosidade, motiva e engaja os estudantes a aprender, a compartilhar e buscar ativamente a construções de seus novos conhecimentos, praticando os conceitos com a utilização e o suporte de tecnologias físicas ou digitais.

Como contribuições, este trabalho apresenta diferentes MAs para suporte ao ensino e a aprendizagem ativa da programação. Isto é necessário pois, muitas vezes, os

docentes não sabem qual MA adotar em sala de aula frente aos diferentes assuntos e disciplinas que precisa ministrar ao longo do semestre. Como trabalhos futuros pretende-se desenvolver um repositório colaborativo aberto em que os docentes possam identificar, selecionar, adotar, discutir, comentar, avaliar e possivelmente colaborar com (novas ou não) MAs utilizadas durante o ensino de programação.

Referências

Acharya, S. and Gayana, M. (2021). Enhanced learning and improved productivity of students' using project based learning approaches for programming courses. *Journal of Engineering Education Transformations*, 34:524–530.

Bacich, L. and Moran, J. (2018). *Metodologias ativas para uma educação inovadora: uma abordagem teórico-prática*. Penso Editora.

Berssanette, J. and de Francisco, A. (2021). Active learning in the context of the teaching/learning of computer programming: A systematic review. *Journal of Information Technology Education: Research*, 20(1):201–220.

Blatt, L., Becker, V., and Ferreira, A. (2017). Mapeamento sistemático sobre metodologias e ferramentas de apoio para o ensino de programação. In *Anais do Workshop de Informática na Escola*, volume 23, page 815.

Borges, R., Oliveira, P. R. F., Lima, R. d. R., and De Lima, R. (2018). A systematic review of literature on methodologies, practices, and tools for programming teaching. *IEEE Latin America Transactions*, 16(5):1468–1475.

Borges, T. S. and Alencar, G. (2014). Metodologias ativas na promoção da formação crítica do estudante: o uso das metodologias ativas como recurso didático na formação crítica do estudante do ensino superior. *Cairu em revista*, 3(4):119–143.

Brito, P., Fortes, R., Faria, F., Lopes, R. A., Santos, V., and Magalhães, F. (2019). Programação competitiva como ferramenta de apoio ao ensino de algoritmos e estrutura de dados para alunos de ciência da computação. In *Simpósio Brasileiro de Informática na Educação (SBIE)*, volume 30, page 359.

Casarotto, R. I., Bernardi, G., Cordenonsi, A. Z., and Medina, R. D. (2018). Logirunner: um jogo de tabuleiro como ferramenta para o auxílio do ensino e aprendizagem de algoritmos e lógica de programação. *RENOTE*, 16(1).

Castro, R. M. and Siqueira, S. (2019). Técnicas alternativas de ensino (aprendizagem ativa) para disciplinas da computação: Um mapeamento sistemático no contexto brasil. In *Anais do Workshop de Informática na Escola*, volume 25, pages 1409–1413.

Cera, M. C., Dal Forno, M. H., and Vieira, V. G. (2012). Uma proposta para o ensino de engenharia de software a partir da resolução de problemas. *Revista Brasileira de Informática na Educação*, 20(03):116.

Costa, A. F. F., de MELO, A. F. M. F., MOREIRA, G. G., CARVALHO, M. d. A., and LIMA, M. V. d. A. (2017). Aplicação de sala invertida e elementos de gamificação para melhoria do ensino-aprendizagem em programação orientada a objetos.

Diesel, A., Baldez, A. L. S., and Martins, S. N. (2017). Os princípios das metodologias ativas de ensino: uma abordagem teórica. *Revista Thema*, 14(1):268–288.

Kitchenham, B. and Charters, S. (2007). Guidelines for performing systematic literature reviews in software engineering. Technical report, Keele University and Durham University Joint Report.

Luxton-Reilly, A., Albluwi, I., Becker, B. A., Giannakos, M., Kumar, A. N., Ott, L., Paterson, J., Scott, M. J., Sheard, J., and Szabo, C. (2018). Introductory programming: a systematic literature review. In *Proceedings of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education*, pages 55–106.

Medeiros, R. P., Ramalho, G. L., and Falcão, T. P. (2018). A systematic literature review on teaching and learning introductory programming in higher education. *IEEE Transactions on Education*, 62(2):77–90.

Melo, S., Oliveira, R., and Soares Neto, C. d. S. (2016). Game of code: aplicando gamificação em disciplinas de programação. In *XV Simpósio Brasileiro de Jogos e Entretenimento Digital (SBgames 2016)*.

Moreira, J. A. and Monteiro, W. M. (2018). O uso da computação desplugada em um contexto de gamificação para o ensino de estrutura de dados. *RENOTE*, 16(2):546–555.

Mulcahy, D. (2002). *Team-based learning: A transformative use of small groups*. Greenwood publishing group.

Nagai, W., Izeki, C., and Dias, R. (2016). Experiência no uso de ferramentas online gamificadas na introdução à programação de computadores. In *Anais do Workshop de Informática na Escola*, volume 22, page 301.

Raj, A. G. S., Patel, J., and Halverson, R. (2018). Is more active always better for teaching introductory programming? In *2018 International Conference on Learning and Teaching in Computing and Engineering (LaTICE)*, pages 103–109. IEEE.

Raposo, E. H. S. and Dantas, V. (2016). O desafio da serpente-usando gamification para motivar alunos em uma disciplina introdutória de programação. In *Simpósio Brasileiro de Informática na Educação (SBIE)*, volume 27, page 577.

Ribeiro, I. C., Silva, W., and Feitosa, E. L. (2021). Repositório colaborativo para apoiar a adoção de metodologias ativas no ensino de programação. In *Anais Estendidos do Simpósio Brasileiro de Educação em Computação*, pages 56–57. SBC.

Silva, T. S. C., de Melo, J. C. B., and Tedesco, P. C. d. A. R. (2018). Um modelo para promover o engajamento estudantil no aprendizado de programação utilizando gamification. *Revista Brasileira de Informática na Educação*, 26(03):120.

Silva, W., Steinmacher, I., and Conte, T. (2019). Students' and instructors' perceptions of five different active learning strategies used to teach software modeling. *IEEE Access*, 7:184063–184077.

Stephan, J., Oliveira, A., and Renhe, M. C. (2020). O uso de jogos para apoiar o ensino e aprendizagem de programação. In *Anais do XXXI Simpósio Brasileiro de Informática na Educação*, pages 381–390. SBC.

Diante deste cenario, as Metodologias Ativas (MAs) v´ em ganhando destaque entˆ re os docentes [Ribeiro et al. 2021]. Segundo Diesel et al. (2017), as MAs possibilitam uma mudanc¸a no paradigma de aprendizagem, onde o estudante sai do papel de agente passivo (apenas escuta e recebe o conteudo que ´ e transmitido pelo docente) e passa para ´ o papel de agente ativo da aprendizagem, tornando-se o principal responsavel por sua ´ aprendizagem.

Appendix C

Informatics in Education, 2024, Vol. 23, No. 2, 279–322

DOI: 10.15388/infedu.2024.11

Active Learning Methodologies for Teaching Programming in Undergraduate Courses: A Systematic Mapping Study

Ivanilse CALDERON[1,3], Williamson SILVA[2], Eduardo FEITOSA[3]
[1]*Federal Institute of Rondônia - IFRO, Brazil*
[2]*Federal University of Pampa - UNIPAMPA, Brazil*
[3]*Federal University of Amazonas - UFAM, Brazil*
e-mail: ivanilse.calderon@ifro.edu.br, williamson.silva@gmail.com, efeitosa@icomp.ufam.edu.br

Received: April 2023

Abstract. Teaching programming is a complex process requiring learning to develop different skills. To minimize the challenges faced in the classroom, instructors have been adopting active methodologies in teaching computer programming. This article presents a Systematic Mapping Study (SMS) to identify and categorize the types of methodologies that instructors have adopted for teaching programming. We evaluated 3,850 papers published from 2000 to 2022. The results provide an overview and comprehensive view of active learning methodologies employed in teaching programming, technologies, programming languages, and the metrics used to observe student learning in this context. In the results, we identified thirty-seven different ALMs adopted by instructors. We realized that seventeen publications describe teaching approaches that combine more than one ALM, and the most reported methodologies in the studies are Flipped Classroom and Gamification-Based Learning. In addition, we are proposing an educational and collaborative tool called CollabProg, which summarizes the primary active learning methodologies identified in this SMS. CollabProg will assist instructors in selecting appropriate ALMs that align with their pedagogical requirements and teaching programming context.

Keywords: teaching programming, active learning methodologies, computer programming.

1. Introduction

Teaching and learning computing is not trivial due to the fundamental subjects in the area, especially those related to programming (Luxton-Reilly *et al.*, 2018), since they are considered complex and require the complete understanding of abstract concepts (Raj *et al.*, 2018; Turpen *et al.*, 2016). Learning programming requires students to plan solutions to problems, transform the plans into syntactically correct instructions for execution, and assess the consequential results of executing those instructions (Chao, 2016).

Analyzing the Computer Science (CS) curriculum, we perceive that the introductory CS courses (CS0, CS1, and/or CS2) provide the understanding of fundamental programming topics for the students (Lang *et al.*, 2006). Typically, they are curricular units that promote the initial contact of Science, Technology, Engineering, and Mathematics (STEM) undergraduate students with computational thinking and programming languages. However, why do introductory programming courses have high failure and dropout rates?

We highlight two reasons. We identify two reasons. First, higher education institutions are often associated with traditional teaching methods and resistance to change (West *et al.*, 2007). Additionally, most instructors adopt traditional teaching methodologies, causing students to lose interest in learning. Second, according to Sobral (2021b), teaching and learning how to program are challenging tasks. Teaching programming is more than coding and translating an algorithm into a language that a computer can understand. It is to think and solve the problem of creating an algorithm (Sobral, 2021c). For computer science students, acquiring the necessary skills for software development is one of the main challenges faced. These problems make students unable to develop specific skills (e.g., abstraction) and often abandon classes and sometimes even the course (Sobral, 2021b). To combat these problems, instructors and researchers must constantly update and/or modify teaching methodologies (Garcia *et al.*, 2021).

Over the past few decades, there has been a significant evolution in technological resources that can support the teaching and learning process. As a positive contribution to the teaching process, active learning methodologies have been widely adopted in developing strategies to overcome learning difficulties, lack of motivation or engagement on the part of students, or even dropping out of the course (Sobral, 2021a).

Active Learning Methodologies (ALMs) combine active student participation, experimental learning, and action learning. These methodologies make students more responsible for learning, increasing their motivation and satisfaction (Imbulpitiya *et al.*, 2020). It is essential to highlight that ALMs induce aspects of active learning, including other concepts, such as collaborative and cooperative learning. In active learning, students learn through instructor-defined activities, which are responsible for supervising and proposing discussions and challenges, and performed through collaborative or cooperative learning, which involves two or more participants (de Andrade *et al.*, 2021). According to Chandrasekaran *et al.* (2016), the ALMs are considered necessary in the learning process since they involve students actively constructing knowledge and change the role of the instructor, who was previously a transmitter of content and information for a learning facilitator. Think-pair-share, Group Writing assignments, Peer Instruction, and Problem-Based Learning are examples of ALMs employed to teach and learn programming.

In the educational context of teaching programming, it is crucial to recognize that programming is a practical skill that demands hands-on experience for mastery. ALMs, such as hands-on projects, labs, and interactive exercises, allow students to engage with and apply programming concepts directly. This iterative process contributes to developing their problem-solving and programming skills over time. ALMs embody teaching

methodologies prioritizing the student's central role in learning, fostering engagement, active participation, and the construction of knowledge. They prove highly effective due to the inherently practical and problem-solving nature of programming itself, facilitating practical learning, honing problem-solving abilities, fostering collaboration, and promoting student teamwork (Eickholt, 2018).

However, which ALM should instructors adopt for teaching programming in computing? To answer this question, we must first consider several related questions: In which course or course will the instructor use the ALM? Will the instructor incorporate ALMs throughout the entire course, or will they use them in specific contexts? Does the instructor know ALM? Does he have time to learn how to use it? Although secondary studies have been conducted to examine publications analyzing the adoption of ALMs (de Andrade *et al.* (2021), Garcia *et al.* (2022), Suarez-Escalona *et al.* (2022), Ahshan (2021)), they have not centered explicitly on identifying suitable methodologies to aid educators in teaching programming at the higher education level, nor have they proposed a collaborative and open repository to support programming instructors. Through an SMS, we can compile the factors that may bolster programming teaching and ascertain which ALMs have been embraced, enabling educators to implement these methodologies in their classrooms.

This research aims to summarize and characterize, through a Systematic Mapping Study (SMS), the ALMs employed in teaching computer programming in undergraduate computing courses. Thus, this SMS provides an overview of the current scenario and characterizes the research that adopts different ALMs when teaching computer programming. It also identifies the contents/classes, tools, and programming languages and the metrics presented in the publications. We hope that Computer Science Education communities and researchers will use this research to improve academic education and industry training.

The remainder of this paper is organized as follows. Section 2 describes the background. The protocol of the systematic mapping is presented in Section 3. In Section 4, we present the results of selected studies. Section 5 contains a discussion of the results. Section 6 shows the effects of this SMS results in the proposal for the new educational technology called CollabProg. Section 7 addresses threats to validity. Finally, conclusions and further work are presented in Section 7.

2. Background

This section presents the theoretical concepts of teaching computer programming and active learning methodologies.

2.1. *Teaching Computer Programming*

Programming is recognized as an essential competency for addressing real-world problems using computational tools in the 21st Century (Chao, 2016), and consequently,

the promotion of skills related to computer programming has been encouraged. Learning computer programming is a crucial step towards developing these skills.

Programming courses should stimulate and develop students' skills and competencies necessary for them to be able to solve complex real-world problems. In other words, skills may encompass coding (the ability to write computer code using specific programming languages to create programs and solutions), problem-solving, logical thinking, debugging, and abstraction. The ACM and IEEE curriculums state that students are expected to learn the knowledge, skills, and attitudes presented at the undergraduate level (ACM and IEEE, 2013). For Petri and von Wangenheim (2017), computer science graduates should be able to design and implement systems involving software and hardware.

However, when it comes to teaching and learning programming, the literature over the years has shown that, when teaching programming to students, instructors could be more successful and need to be (Berssanette and de Francisco, 2021). When instructing programming, it's crucial to recognize that competencies extend beyond mere technical skills; they encompass the ability to apply these skills across diverse contexts and effectively combine them to attain larger objectives. These competencies include problem-solving, collaboration, self-learning, analysis, adaptation, and technical communication. Consequently, programming is one of the most prevalent means of nurturing computational thinking, as it requires the application of computer science concepts such as abstraction, debugging, remixing, and iteration to address problem-solving (Yang *et al.*, 2023).

In light of this, innovative pedagogical approaches to teaching programming have become an ongoing topic of discussion in universities and colleges worldwide. The teaching of programming is centered on the three aspects of programming: design, development, and testing (Kong *et al.*, 2020). The inadequate balance in applying these concepts results in a disproportionate amount of time that the student spends to abstract the problem from the real world and create a solution, then develop this solution and test it. This leads to frustration and demotivation and is a severe problem of these core disciplines for computer science (Rajaravivarma, 2005). Lister *et al.* (2004) and Tenenberg and Fincher (2005) highlight significant deficiencies in the learning outcomes of students who studied programming in different higher education courses. These scenarios originate from mistakes at the beginning of studies and poor understanding of basic concepts, procedures, and processes (Kinnunen and Malmi, 2006). Moreover, some deficiencies are identified in the teaching of programming, particularly concerning the students' lack of skills for programming (McCracken *et al.*, 2001).

According to Barnes *et al.* (2008) and Parsons (2011), the nature of computing and this generation of students has changed remarkably in recent years. However, most higher education computing courses are still taught in traditional ways and may not be adequate to keep pace with modern concerns and may not support the necessary learning. According to (Petri and von Wangenheim, 2017), student-centered instructional strategies are needed to achieve more effective learning at higher levels, thus allowing them to learn by doing.

2.2. *Active Learning Methodologies*

Active learning (AL) or Active Learning Methodologies (ALM), a term popular in US education circles in the 1980s, encourages learners to take responsibility for their learning, requiring their experience in education to inform their process of learning (Zayapragassarazan and Kumar, 2012). The premise is to engage more actively the students through various methodologies, strategies, approaches, and student-centered pedagogical techniques so that they become involved in the teaching and learning process. The idea is that they apply their knowledge meaningfully, employing higher-order thinking skills and reflecting on their learning to build new knowledge (Berssanette and de Francisco, 2021).

Although understanding the concept behind ALM is simple, it does not have a specific or strict definition. ALMs have no specific definition and can have different interpretations depending on the subject or group of learners involved (Hativa, 2001; Kane, 2007). On the other hand, it is easy to observe that ALMs can draw from various learning theories emphasizing active student participation, knowledge construction, and the development of practical skills, especially Constructivist Theory (Ben-Ari, 2001; Jonassen *et al.*, 1995) where the knowledge is not simply absorbed from textbooks and lectures but actively constructed by the student (Ben-Ari, 2001).

It is a fact that ALMs help instructors develop and improve general principles about teaching and learning. Using ALMs, instructors are responsible for organizing appropriate learning activities that allow learners to explore and develop their knowledge and thinking. They must use practical teaching methods by providing numerous examples of activities and pedagogical techniques that students can enjoy in various learning situations. Various teaching methods have been created to achieve this goal (Hativa, 2001; Kane, 2007). In practice, the possibilities for adopting ALMs vary widely in intensity and implementation and include diverse approaches such as group problem solving, use of tools, and the realization of projects in classes or workshops (Freeman *et al.*, 2014). So, the typical question made by instructors is: Which ALM should I adopt in my classroom?

There is much evidence in the literature about the advantages of using ALMs in teaching, especially in computing. Several researchers have highlighted the positive impacts on student learning, attitudes, critical thinking, and reducing students' failures in subjects for teaching programming (Park and Choi, 2014). The use of ALMs allows the instructors to create learning situations for students to build knowledge about the contents learned to develop critical thinking and reflections on the exercises they carry out, as well as exploring attitudes, personal values, and learning through doing (Parsons, 2011).

However, adopting ALMs for teaching programming has practical implications for instructors who wish to implement active learning. There are many ALMs to be adopted. The possibilities vary widely in intensity and implementation and include diverse approaches such as group problem solving, use of tools, and the realization of projects in classes or workshops (Freeman *et al.*, 2014). But which choice? Do the instructors know the various successful or unsuccessful ways of using and implementing ALMs? Do they have some knowledge and planning to be considered to use an ALM?

To address these questions, this research investigates how instructors have used active learning methodologies while teaching programming in undergraduate courses. In addition, we were also interested in which subjects they were applied to, which programming languages were used, and if they were realized experimental studies.

3. Research Methodology

We conducted a Systematic Mapping Study (SMS) to identify the scenario in which instructors used the ALMs while teaching programming. The SMS follows the procedures described in Kitchenham (2012), i.e., planning, conducting, and analyzing the results. The planning activities and their steps are described in the following subsections, and Sections 4 and 5 show the results.

3.1. *Research Questions*

We defined the following Research Question (RQ) to guide our work:

- **RQ1**: How have instructors used active methodologies during the teaching of programming in undergraduate courses?

To answer the research question, we sought to identify three aspects in the selected publications: **(i)** Which ALMs have been adopted for teaching programming? **(ii)** What is the programming teaching context?, and **(iii)** What kinds of experiments have been performed by the researchers? Based on the three aspects, research sub-questions (SQs) were defined for each element to answer specific questions (see Table 1).

3.2. *Search Strategy*

This SMS proposes investigating the ALMs instructors adopt while teaching programming in undergraduate courses. For this, we used the search mechanism available in most digital libraries based on textual research expressions and a manual search of events in

Table 1
Sub-questions. Source: The authors.

Aspect	Sub-questions
Methodology	**SQ1.** Which ALMs were addressed in the publications?
Teaching	**SQ2.** Which subjects were mentioned in the publications?
	SQ3. Which programming languages were reported in the publications?
Experiments	**SQ4.** What type of experimental study was carried out?
	SQ5. What evaluation metrics were reported in the publications?
	SQ6. Which technologies were adopted during the teaching of programming?

computing. According to Steinmacher *et al.* (2015), the definition of the search string is an essential phase for the effectiveness of the search stage of an SMS. The search string was defined based on two essential terms of our research questions: (1) active methodologies and (2) teaching of programming. Besides this, to help us, the studies by Kelleher and Pausch (2005), Raj *et al.* (2018), Tharayil *et al.* (2018), and Aksit *et al.* (2016) were used as control articles to support the selection of keywords and synonyms related to the research questions.

Therefore, the query was iteratively evolved several times to ensure that a comprehensive set of synonyms was used to allow high coverage. A search string refinement process was performed to include new terms from previously selected publications and verify whether the control articles provided hits via the test search strings. The search string used in this study is presented below.

("active learning" OR "active methodology")
AND
("introductory programming" OR "introduction to programming" OR "novice programming" OR "novice programmers" OR "CS1" OR "CS 1" OR "programming course" OR "learn programming" OR "learning to program" OR "teach programming" OR "training programming" OR "instruction programming" OR "coaching programming")

After defining the search string, we selected the following libraries: **(i)** IEEE Xplore Digital Library (IEEE)[1], **(ii)** ACM Digital Library (ACM)[2], and **(iii)** Scopus Library[3]. These libraries were selected for the following reasons: (i) They possess robust search engines with effective operations and broad search scope; (ii) Scopus serves as a meta-library, indexing publications from several renowned publishers, including Springer, Elsevier, and Taylor & Francis; (iii) ACM and IEEE rank as the top two digital libraries in Computer Science. Our choice of these databases is informed by recommendations from prior systematic literature reviews, affirming their suitability andrelevance as sources (Nakamura *et al.*, 2022).

Additionally, a manual search was carried out in the following events and scientific journals on education in computing and informatics in education in Brazil: **(i)** Brazilian Symposium on Informatics in Education (SBIE), **(ii)** Workshop on Computing at School (WIE), **(iii)** Computer Education Workshop (WEI), **(iv)** Brazilian Symposium on Games and Digital Entertainment (SBGames), **(v)** International Congress of Educational Informatics (TISE), **(vi)** New Journal Technologies in Education (RENOTE) and **(vii)** Brazilian Journal of Informatics in Education (RBIE). The choice to perform searches in Brazilian sources, including journals and specialized events in the field of computing

[1] https://www.ieee.org/

[2] http://dl.acm.org/

[3] http://www.scopus.com/

and informatics education in Brazil, was motivated by several vital reasons that align with the scope of this research. First and foremost, it is crucial to emphasize that Brazil's educational and technological landscape possesses distinct characteristics that can significantly influence the emergence of pedagogical approaches and practices that are both unique and highly relevant to the national context. As suggested by Mendes *et al.* (2020), it is advisable to follow the references cited in each selected paper to discover additional pertinent sources. Consequently, exploring Brazilian sources has provided access to studies, research findings, and local experiences frequently unavailable internationally. This enrichment contributes significantly to the discourse and comprehension of the challenges and progress in computing education within a distinct contextual framework.

Our aim in incorporating Brazilian sources was to encourage cultural and linguistic diversity in academic discourse, enabling researchers and educators from diverse backgrounds to share their knowledge and promoting a more inclusive and worldwide outlook in studies related to educational informatics. Thus, it was a strategic decision to incorporate Brazilian sources into the research to enhance and provide context to the results despite potential limitations in linguistic accessibility for confident readers of the international journal.

3.3. *Publication Selection Criteria*

Following the procedures described by Kuhrmann *et al.* (2017), inclusion criteria (**IC**) and exclusion criteria (**EC**) were defined for the publications returned by the search string. These criteria are needed to select only relevant publications for the search and filter publications that require further analysis. The criteria are presented in Table 2.

Table 2
Criteria for inclusion or exclusion of publications. Source: The authors.

Criteria	ID	Description
Inclusion of publication (IC)	**IC1**	Publications that discuss the perceptions of instructors and/or students regarding the ALMs used during the teaching and learning of programming classes should be selected.
	IC2	Publications that present experimental studies on the use of ALMs during the teaching of programming should be selected.
	IC3	Publications that present learning assessment metrics about the use of the ALM(s) adopted should be selected.
Exclusion of publication (EC)	**EC1**	Publication is not available for reading and data collection (paid publications or those not made available by the search engine).
	EC2	Publications that do not meet the inclusion criteria.
	EC3	Publications not written in English or Portuguese.
	EC4	The following types of publication: books, doctoral theses, master's dissertations, patents, tutorials, workshop proposals, or posters.
	EC5	Duplicate publications (for example, a paper with a study published in different places or on different dates). In this case, we considered only the most complete and latest version.

3.4. *Processes for the Selection of Publications*

We applied two selection filters (inclusion and exclusion criteria) in the returned publications. We adopted the Start tool[4] to help us filter the papers. If the search returned duplicate papers, the tool would indicate this, and only one article remained for analysis.

In the **1st Filter**, we analyzed the titles and abstracts of the returned publications, and only the publications that adopted ALMs for teaching programming were selected. Via this filter, we excluded only papers that were clearly out of scope. In case of doubts regarding the publication's relevance, the articles were kept for further analysis.

In the **2nd Filter**, we read the publications selected by the first filter to conduct a more detailed analysis and identify and extract the data according to the inclusion and exclusion criteria.

3.5. *Data Extraction*

From the publications selected, we extracted relevant information using a form summarized in Table 3.

3.6. *Execution of Systematic Mapping*

The systematic mapping involved three researchers to reduce the interpretation bias of a single researcher. Two Ph.D. researchers reviewed the inclusion and exclusion criteria protocol and analyzed the search strategy.

To assess the reliability of the publication selection process, two researchers independently ranked a sample of 40 publications randomly selected from the set of publications returned to measure the level of agreement among them.

In this classification, the title and abstract of each publication were evaluated and classified based on the selection criteria. Cohen's Kappa coefficient was applied after this step, and the statistical test was used, which is a measure of intra-and inter-observer agreement and the degree of understanding beyond what would be expected by chance alone (Cohen, 1960). The evaluation result showed a consensus between researchers of 0.89 (Kappa concordance), representing an almost perfect concordance. Based on this result, the steps of selecting and extracting data from publications were continued.

3.7. *Identified Publications*

Initially, 3,850 publications were found in the digital libraries and annals: 954 in the Scopus library, 2,190 in the manual search, 373 in the IEEE library, and 333 in the

[4] `http://lapes.dc.ufscar.br/tools/start_tool`

Table 3
Data to be extracted from publications. Source: The authors

Aspect	Extraction items	Data to be extracted
General information	Title	The title of the publication.
	Author(s)	The name of the author(s) of publication.
	Type of publication	The type of publication (e.g., paper in a journal, conference paper, etc.).
	Publication Year	The publication year of the paper.
	Venue of the paper	The name of the venue where the paper was published.
Methodologies	Identified ALM	Name of the ALM addressed in the publication.
Teaching	Subject	The name of the subject taught.
	Course	The name of the course reported.
	Language	Name of the programming language that was used.
Experiments	Experimental study	Does the publication present an experimental study?
	Type of experimental study	Does the publication describe the type of study? If yes, which one (Unterkalmsteiner *et al.*, 2011; Creswell *et al.*, 2006): (i) case study, if an empirical inquiry investigates a contemporary phenomenon within its real-life context, especially when the boundaries between phenomenon and context are not evident; (ii) experience report, if the focus of the study is directed towards reporting educational experiences without stating research questions or a theoretical concept, which is then evaluated empirically; (iii) controlled experiment, if the study performs an empirical investigation that manipulates one or more variables or factors in the studied context, verifying the effects of this manipulation; (iv) action research, if the study states this research method explicitly; (v) survey, if the study collects quantitative and/or qualitative data using a questionnaire or interviews; (vi) mixed methods if involves collecting, analyzing, and mixing qualitative and quantitative approaches in a single study or a series of studies.
	Technologies	Does the publication present the technologies, tools, and applications used in teaching programming? If yes, list them.
	Metrics	Does the publication describe the metrics used to evaluate the improvement in teaching programming? If there are metrics, specify the metric used in the publication.

ACM library. After removing duplicate publications, the total number of publications selected for analysis using the first filter was 3,709. Of these 3,709 publications, 2,979 were excluded after using the first filter since they did not meet the inclusion criteria.

According to the established inclusion and exclusion criteria, the remaining 730 publications were read and analyzed using the second filter. At the end of the evaluation process, 80 publications were accepted and had their data extracted. Fig. 1 summarizes the complete data selection and extraction process. The publications selected in this SMS are presented in Table 13, organized by their relevance as obtained from digital libraries.

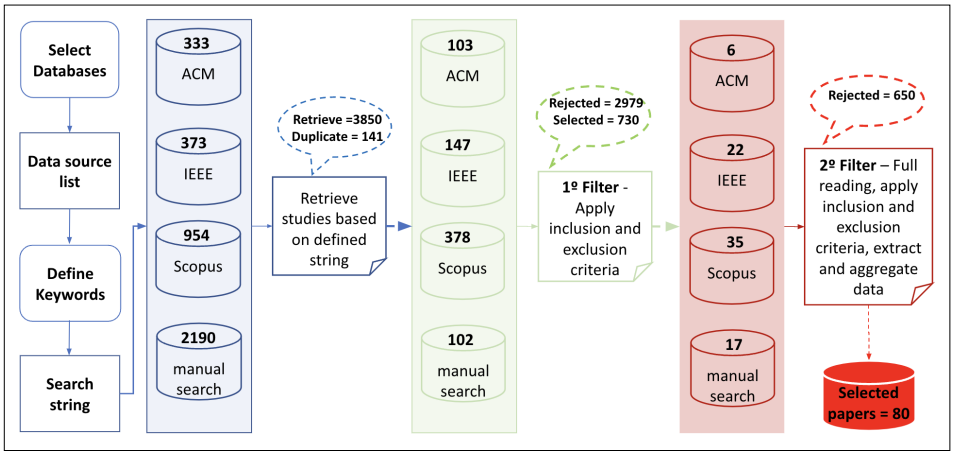


Fig. 1. Results of systematic mapping filters. Source: The authors.

4. Results

4.1. *Publication Trend*

This section presents the publication trends for the research topic investigated in this SMS. Fig. 2 shows the variation in the number of publications on adopting ALMs for teaching programming. During the research period, 2018 has the most significant number of publications. Ten studies were published in 2021, while only three were published in 2022. The period from 2019 to 2020 has 12 and nine publications, respec-

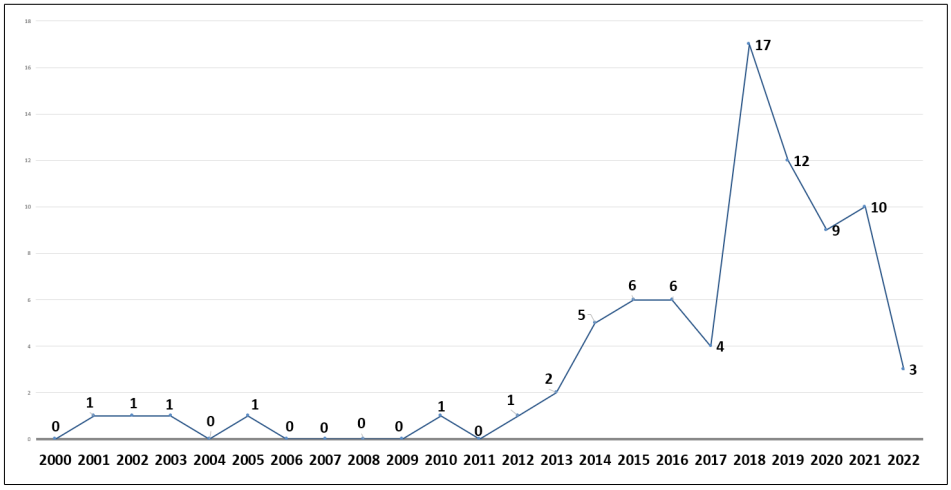


Fig. 2. Publication trend by year. Source: The authors.

Table 4
Events that resulted in more than two publications on the SMS theme. Source: The authors

ID	Publication venue	#Publications
01	Frontiers in Education Conference (FIE)	15
02	Technical Symposium on Computer Science Education (SIGCSE)	7
03	Brazilian Symposium on Informatics in Education (SBIE)	6
04	Workshop on Computer Education (WEI)	5
05	Brazilian Symposium on Games and Digital Entertainment (SBGames)	2
06	Global Engineering Education Conference (EDUCON)	2
07	International Conference on Learning and Teaching in Computing and Engineering (LaTICE)	2
08	Conference on Information Technology Education (SIG)	2
09	Others (places with only one publication)	39
-	**Total**	**80**

tively. Between 2013 and 2017, there was a variation between two and six publications. From 2001 to 2012, the number of publications varied between zero and one per year.

We observed decreased publications between 2020 and 2022, possibly due to the pandemic and the shift to remote learning. One possible reason for this could be the numerous planned studies on in-person teaching. However, in 2021, some strategies, such as those in publication S64, were adapted for emergency remote teaching. Given this scenario, it is clear that there is a significant number of publications on the adoption of ALMs for learning programming. Therefore, it is believed that the community is constantly researching the adoption of ALMs to support teaching practices.

The most common publication type is conference papers, with 43 publications. Workshops had 19 publications; finally, the journals had 18 studies published. To present venues for research publications related to adopting ALM in computing, we introduce Table 4, which lists events and journals and their respective number of publications. In this way, we aim to assist researchers new to the field.

We observed that the Frontiers in Education Conference (FIE), an important international conference that focuses on educational innovations and research in engineering and education in computing, leads in the development of new research insights and educational approaches and is the conference with the most significant number of publications of interest to this research. In addition, the Technical Symposium on Computer Science Education (SIGCSE) and the Brazilian Symposium on Informatics in Education (SBIE) presented seven publications each, and the Workshop on Computing Education (WEI) presented five publications.

4.2. *SQ1. Which ALMs were Addressed in the Publications?*

To answer SQ1, the ALMs reported in the publications were analyzed and classified by type, and 37 kinds of ALMs adopted for teaching programming were identified. Fig. 3 shows the types of ALMs mapped in this study. According to Katona and Kovari (2016),

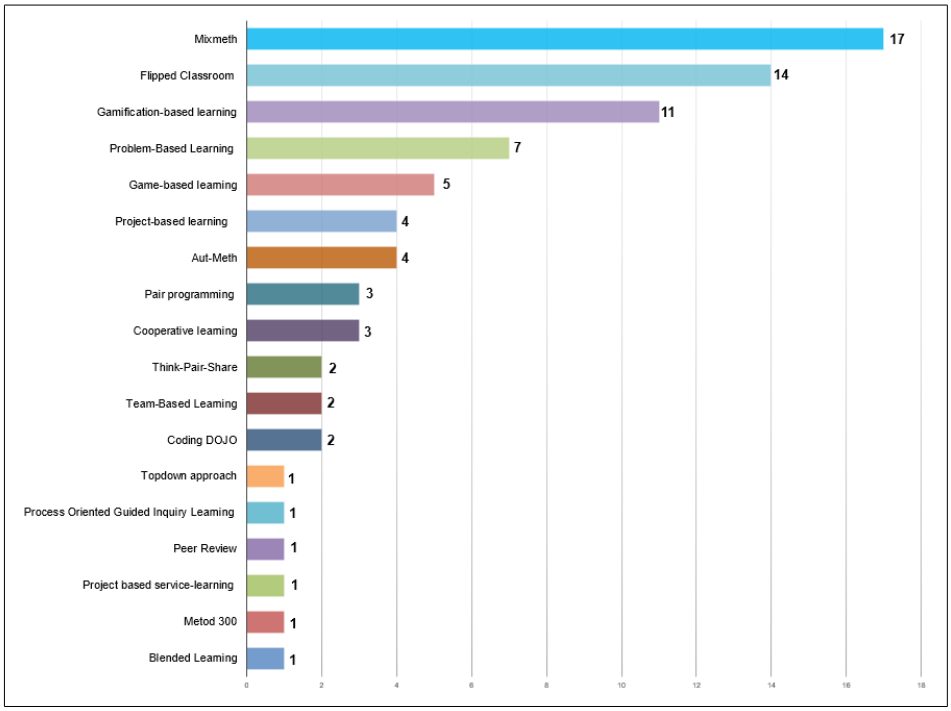


Fig. 3. Types ALMs adopted for teaching programming. Source: The authors.

numerous approaches have been aimed at enhancing students' learning achievements in recent decades through active learning methods. This particularly applies to programming-related courses, where students must practice regularly.

Among the ALMs mapped, we noticed 17 publications presenting approaches that combine more than one ALM. We named and classified them as "Mixed Methodologies" (MixMeth). See all the MixMeth in Table 5. In addition, four publications with proposals for new methodologies were classified as "Authors' Methodologies" (Aut-Meth), i.e., instructors adopt different teaching practices to explore active learning during the teaching schedule. These can be seen in publications S16, S17, S18, and S25.

The ALMs that were jointly adopted stand out with a percentage of 20.9% (17) of the mapped publications, as can be seen in study S12, in which the authors adopted the Flipped classroom (FC) and Problem-Based Learning (PBL) in a mixed way. The FC method uses information technology to invert traditional in-class activities into out-of-class activities and vice versa (Hendrik, 2019). The common practice of this approach is the students watch a pre-recorded lecturer video at home and then in the class meeting. They do a quiz or assignments related to the subject they learned before (Bergmann and Sams, 2012). Project-Based Learning (PBL) is an inclusive teaching approach that involves students investigating real-world problems. With this methodology, students formulate the questions and find solutions to these issues (dos Santos *et al.*, 2018). Therefore, the combination of active methodologies like FC and PBL can be highly ben-

Table 5
Methodologies adopted jointly. Source: The authors

ID	ALM	#Publications
1	Flipped Classroom + Project-Based Learning	S01
2	Mini-lecture + Live-coding + In-class coding	S03
3	Pair programming + Exercise-based learning	S05
4	Flipped Classroom + Problem-based Learning	S12, S15
5	Animated Flowchart with Example Think-Pair-Share	S16
6	Project-Based Learning + SCRUM	S23
7	Student Ownership of Learning + Flipped Classroom	S26
8	Pairing-based pedagogy - Pairing-Based Approach (Pair programming + Blended Learning	S27
9	Flipped Classroom + Team-Based Learning	S28
10	Process Oriented Guided Inquiry Learning + Pair Programming	S35
11	Process Oriented Guided Inquiry Learning + Pair Programming	S21
12	Game-Based Learning + Problem-Based Learning	S43
13	Lecture-based Learning + Problem-Based Learning + Peer Instruction	S46
14	Flipped Classroom + Gamification-Based Learning	S65
15	Blended teaching + Problem-Based Learning + Task-driven + Flipped classroom	S70
16	Learning by Collaboration, Flipped Classroom, Game-Based Learning	S73
17	Flipped Classroom, Peer Discussion, and Just-in-time	S76
18	Coding Dojo, Gamification, Problem-Based Learning, Flipped Classroom and Serious Games	S80

eficial for teaching programming due to the different contributions each one offers. FC method offers benefits such as pre-preparation, an emphasis on practical activities, and heightened interaction with the instructor. Meanwhile, Problem-Based Learning (PBL) promotes student-centered learning, knowledge application, and interpersonal skills development. This effective and engaging approach thoroughly equips students with real-world programming practice.

Notably, the Flipped Classroom and Problem-Based Learning methodologies were individually reported in 17.5% (14) and 9.8% (8) of the publications.

The S35 publication adopted Process-Oriented Guided Inquiry Learning (POGIL) and Pair Programming (PP) for teaching programming. POGIL is a student-centered learning approach that focuses on concept development in the framework of learning teams. Instead of passively listening to a traditional lecture, students work together in groups on specifically designed activities that guide students through the construction of course content (Hu and Shepherd, 2013). The pilot is responsible for typing at the computer or documenting a design in the PP process. The other partner, the co-pilot, observes the driver's work, looks for defects in the driver's position, and is an ever-ready brainstorming partner (Nagappan *et al.*, 2003). Adopting POGIL and PP methodologies can lead to notable enhancements in programming education. These improvements encompass active learning, the promotion of collaboration, the stimulation of critical thinking through guided inquiry, the provision of immediate feedback, ongoing code review, the encouragement of cooperative knowledge building, joint

problem-solving, and a deeper comprehension of algorithms. Consequently, incorporating these approaches into programming education can amplify student engagement, facilitate collaboration, cultivate problem-solving skills, and elevate the quality of generated code. Both methodologies are practical and can be employed in conjunction or separately, depending on the learning objectives and the specific requirements of the class. Thus, it can be seen that the mixed use of ALMs provided instructors with different possibilities to test combinations of ALMs jointly. In this way, different experiences of teaching practices are observed, as well as new opportunities for students to be motivated to learn actively.

We observed that 13.5% (11) of the analyzed studies adopted Gamification-Based Learning (GM). Gamification refers to integrating game elements into non-game contexts. This trend is gaining popularity among educational researchers due to its potential to reduce student boredom and increase active learning, engagement, and motivation (Kaya and Ercag, 2023). According to Venter (2020), GM is considered one of the most promising educational methodologies for this decade, as educators worldwide recognize that the proper design of gamified learning activities can significantly improve student productivity and creativity. Therefore, adopting the GM methodology in programming education innovates by making learning more engaging, practical, and motivating. GM is crucial for attracting and retaining students, developing programming and problem-solving skills, and preparing them for success in the tech industry. Adopting GM also provides significant opportunities, such as student engagement and motivation, promoting practical learning, fostering self-directed learning, and facilitating collaboration.

The Game-Based Learning (GBL) methodology appears in 6.1% (5) of the publications. The game-based approach is unique because it involves and excites students, allowing them to spend their time-solving problems. Additionally, GBL encourages the exploration of different problem-solving methods. In simple, fun games, the students may repeat the process just because they want a different outcome (Rajaravivarma, 2005). The methodology focuses on applying educational games designed to balance learning a specific competence with the gameplay (Qian and Clark, 2016). Currently, it is being adopted in computer science teaching in several areas, such as software engineering, programming, or security (Zhang-Kennedy and Chiasson, 2021).

Aut-Meth appears in 4.9% (4) of the publications, such as S26 and S32. The authors elaborated and used an ALM to explore collaboration and active learning in teaching programming. With the same percentage, Project-Based Learning (PjBL) appears in 4.9% (4) of the publications. PjBL is also an example of a student-centered methodology, through which students learn to build their own learning experiences independently (Paristiowati *et al.*, 2022). The Project-Based Learning (PjBL) methodology involves learning through projects. This methodology challenges students to take responsibility for their learning while promoting positive interdependence, individual accountability, social skills, and equal participation during project presentations. Students can benefit greatly from this learning approach by encouraging communication and leadership (Kholijah *et al.*, 2023).

Finally, 12 types of methodologies were cited by less than four publications: Cooperative Learning (CL) (3), Pair Programming (PP) (3), Team-Based Learning (TBL) (2), Think-Pair-Share (TPS) (2), Coding Dojo (Dojo) (2), Blended Learning (BL) (1), Peer Review (PR) (1), Project-Based Service Learning (PBSL) (1), Method 300 (M300) (1), Process-Oriented Guided Inquiry Learning (POGIL) (1), and Top-Down (TopD) (1).

CL is a widely-used educational approach that the instructors can apply to diverse subjects and populations (Beck and Chizhik, 2006). Also, it can develop computational thinking and knowledge of computational programming (Li *et al.*, 2023). PP is an active learning methodology that compares pair programming and solo programming. Its effectiveness is affected by compatibility factors such as students' skills, personality, and self-esteem (Xu and Correia, 2023). TBL develops critical thinking skills and problem-solving ability to solve problems individually and empowers students to solve complex issues (Sibley and Ostafichuk, 2023). TPS methodology encourages students to consider the problem's solution individually, share their answers with their partners in pairs, and present their solutions orally to the entire class (Hidayati *et al.*, 2023). Dojo is a hands-on workshop session widely used in classroom settings where students can practice programming in groups for collaborative learning. This methodology significantly improves students' skills in designing software and applying design patterns (Nasir, 2023).

BL combines in-person and online instruction for flexibility. It offers face-to-face learning while keeping students safe (Srivatanakul, 2023). PR is an active, authentic activity providing a distinct learning experience in the classroom. This approach demands that students engage in higher-level thinking as they analyze and evaluate the work of others. It is a commonly used technique in industry and is a genuine activity that can help prepare students for the workplace (Turner *et al.*, 2018). At PBSL, students can participate in projects that present challenging and holistic situations requiring them to apply their functional technology skills, critical thinking abilities, and interpersonal skills to understand the issues they must address. The learning experience is highly engaging as they work through the project and solve the problems they encounter (Brescia *et al.*, 2009).

M300 method can be defined as an innovative strategy of active learning, combining features of peer learning and mentoring techniques, which are widely used in active learning (de Castro Junior *et al.*, 2021). POGIL is a suitable pedagogical approach for teaching programming, software testing, and DevOps at the undergraduate level (Joshi and Lau, 2023). The TopD methodology is a pedagogical approach to software development and programming education. It begins with a broad view of the problem to be solved and gradually delves into specific implementation details. This approach is advantageous when teaching object-oriented programming, software architecture, and complex systems development, where organization and structure are vital to project success (Sung and Shirley, 2003). Table 6 shows the ALMs individually adopted per paper.

Table 6

Methodologies individually adopted per paper. Source: The authors

ID	ALM	#Publications
1	Blended Learning (BL)	S36
2	Cooperative Learning (CL)	S17, S32, S33, S77
3	Coding Dojo (DOJO)	S61, S63
4	Flipped Classroom (FC)	S2, S7, S8, S14, S24, S30, S37, S38, S40, S41, S42, S47, S50, S74
5	Game-Based Learning (GBL)	S11, S48, S51, S53, S55, S67
6	Gamification-Based Learning (GM)	S19, S21, S49, S54, S56, S57, S58, S60, S62, S68
7	Method 300 (M300)	S64
8	Programming Case Studies (PCS)	S18
9	Hybrid Two-Stage Model (HTSM)	S25
10	Problem-Based Learning (PBL)	S9, S10, S13, S52, S59, S75, S78, S80
11	Project-Based Service Learning (PBSL)	S44
12	Project-Based Learning (PjBL)	S4, S39, S66, S79
13	Process Oriented Guided Inquiry Learning (POGIL)	S71
14	Pair Programming (PP)	S22, S45, S72
15	Peer Review (PR)	S31
16	Team-Based Learning (TBL)	S6, S20
17	Top-Down (TopD)	S69
18	Think-Pair-Share (TPS)	S29, S34

4.3. *SQ2. Which Subjects were Mentioned in the Publications?*

To answer SQ2, we observed the contents and disciplines presented in the publications, as reported in the studies, and identified approximately 30 different disciplines used for teaching. Table 7 presents the ALMs used for teaching programming in different courses and classes in computing.

In the Introductory Programming class, different ALMs (PBSL, PjBL, PP, TBL, and TPS) have been adopted for the initial teaching of programming, as observed in

Table 7

Subject X Methodologies. Source: The authors.

ALM	Subject/content	Course	#Publication
FC	Data structures and OOP	Computer Engineering	S2
	Introduction to programming and algorithm	Software Engineering	S38
	Introduction to programming/linear data structures	Computer Science	S41
	OOP	Computer Programming	S8, S42
	Computer programming	Computer Science	S30, S37, S40
	Introductory programming	Computer Science and Information Technology, Information Systems	S7, S14, S24, S47, S50

Continued on next page

Table 7 – continued from previous page

ALM	Subject/content	Course	#Publication
MixMeth	Web programming	Informatics	S1
	Introductory programming	Management Information System	S26
	Computer programming	Computer Science	S35
	OOP	Computer Engineering, Software Engineering	S3, S15, S27, S65
	Introductory programming	Computer Engineering, Software Engineering and Information Systems Engineering	S5, S12, S23, S28, S43, S46
GM	Algorithm	Computer Science	S54
	Algorithm and data structures	Computer Science	S57
	OOP	Information Systems	S68
	Programming lab	Computer Science	S58
	Web programming	Information Systems	S62
	Introductory programming I and II	Computer Engineering, Computer Science	S56, S60, S19, S21, S49
GBL	Data Structures	Computer Science	S48
	Programming II	Computer Science	S53
	Programming Logic and Algorithm	Information Systems	S67
	Algorithms	Computer Science and Information Systems	S51
	Introductory programming	Computer science and Information Systems	S11, S55
PBL	OOP	Computer Engineering	S9
	Algorithms and programming I	Computer Engineering	S13
	OOP, data structures and software design	Computer Engineering	S10
	Programming paradigms	Software Engineering	S59
	Teaching programming	not mentioned	S52
AuthMeth	System Programming	Computer Science and Engineering	S16
	Programming	Computer Science	S18
	Introductory programming	Computer Science	S17, S25
PjBL	Introductory programming	Computer Engineering	S4
	Mobile development	Computer Science	S39
	OOP, data structures and systems design	Computer Engineering	S66
CL	Parallel programming	Computer Science	S32
	OOP	Informatics	S33
PP	Introductory Computer Science course	Computer Science	S22
	Mobile app development		S45
TBL	Introduction to systems programming	Computer Science	S6
	Introductory programming		S20
DOJO	Introductory programming, programming language, OOP	Computer Science	S63
	Algorithm		S61
TPS	Introductory programming	Computer Science	S29, S34
BL	Computer programming	Computer Science	S29, S36
M300	Algorithm and programming	Computer Science	S64
PBSL	Introductory programming	Computer Engineering	S44
PR	OOP	Computer Science	S31

Note: FC – Flipped Classroom; MixMeth – Mixed Methodologies; GM – Gamification-Based Learning; GBL – Game-Based Learning; PBL – Problem-Based Learning; Aut-Meth – Authors' Methodologies; PjBL – Project-Based Learning; CL – Cooperative Learning; PP – Pair Programming; TBL – Team-Based Learning; BL – Blended Learning; M300 – Method 300; PBSL – Project-Based Service Learning; PR – Peer Review.

publications S4, S22, S29, SS34, S39, S44, S45, and S66. Regarding the teaching of algorithms and data structures, the adoption of GM, GBL, FC, M300, and Dojo is observed, as observed in publications S11, S19, S21, S24, S30, S37 S48, S49, S51, S53-S58; S60-S64, S67 and S68.

In teaching computer programming, the BL, FC, and MixMeth are adopted, according to publications S1-S3, S5, S7, S8, S12, S14, S15, S23, S24, S26, S28, S30, S35, S37, S38, S40-S43, S46, S47, S50, S54, S61 and S65.

Finally, in classes such as Parallel Programming, Object-Oriented Programming (OOP), System Programming, Software Design, Teaching Programming, and Programming paradigms, the CL, Aut-Meth, PBL, and PR methodologies were mapped and are presented in publications S9, S10, S13, S16-S18, S25, S31-S33, S52, and S59.

4.4. *SQ3. Which Programming Languages were Reported in the Publications?*

To answer SQ3, we verified the programming languages reported in the studies. Table 8 summarizes the types of programming languages found in the publications, which are analyzed by the type of ALM used for their teaching. The publications S64, S65, S67, S70, S72, and S74 do not show which programming language was used.

Java is among the most used programming languages mentioned in 27 publications. The C++ and C languages are used in 12 and 10 publications. Finally, Python was mentioned in 11 publications. Not all publications cited which programming language was used, and some did not mention it clearly in the study. The following publications, S51, S53, S67 (GBL), S22 and S45 (PP), S6 (TBL), S64 (M300), S44 (PBSL), S31 (PR) and S54 (GM) are examples of this fact.

Table 8
Programming Language X Methodology. Source: The authors

ALM	Programming language	#Publication
FC	Java, C#, C,Python	S2, S7, S8, S14, S24, S30, S37, S38, S40, S41, S42, S47, S50
MixMeth	Javascript, PHP, Java, C++, C,Python	S1, S3, S5, S12, S15, S23, S26, S27, S28, S35, S43, S46, S65
GM	Java, C++, C,Python, PHP, Ruby	S19, S21, S49, S54, S56, S57, S58, S60, S62, S68
PBL	Java C, Python	S9, S10, S13, S52, S59
Maut	Assembly, C++, Java	S16, S17, S18, S25
PjBL	Python Java	S4, S39, S66
CL	C, C++, JAVA	S32,S 33
GBL	Python, Java	S48, S55
DOJO	C, Python, Java	S61, S62
BL	Java	S36
TBL	C++	S20

Note: FC – Flipped Classroom; MixMeth – Mixed Methodologies; GM – Gamification-Based Learning; PBL – Problem-Based Learning; Aut-Meth – Authors' Methodologies; PjBL – Project-Based Learning; CL – Cooperative Learning; GBL – Game-Based Learning; DOJO – Coding Dojo; BL – Blended Learning; TBL – Team-Based Learning.

4.5. *SQ4. What Type of Experimental Study was Carried out?*

Research and development in information technology and computer science rely heavily on empirical studies. These studies provide (i) the necessary foundation for making technical decisions, (ii) evaluating the efficiency of systems and solutions, and (iii) generating evidence-based knowledge to improve computing practices in different fields.

To answer SQ4, the types of studies were carried out: case studies, controlled experiments, surveys, and mixed methods.

Therefore, we observed that all the studies carried out were experimental. Table 9 presents the types of studies identified in the publications. Given this panorama, we observed that 87.76% of the studies carried out a case study, which evidences the instructors' concern regarding the applicability of the methodologies, technologies, and types of programming languages in daily teaching practice.

Table 9
Type of studies X Methodology. Source: The authors

ALM	Action research	Case study	Focus group	Interviews	Observations	Survey	#Publication
MixMeth		X		X		X	S1, S3, S5, S12, S15, S23, S26, S27, S28, S35, S43, S46, S65, S70, S73, S76, S80
FC		X				X	S2, S7, S8, S14, S24, S30, S37, S38, S40, S41, S42, S47, S50, S74
GM		X				X	S19, S21, S49, S54, S56, S57, S58, S60, S62, S68
PBL		X					S9, S10, S13, S52, S59, S75, S78, S80
GBL		X				X	S48, S51, S53, S55, S67
AuthMeth		X					S16, S17, S18, S25
PjBL		X					S4, S39, S66, S79
CL	X	X					S32, S33, S77
PP		X					S22, S45, S72
DOJO		X					S61, S63
TBL		X		X	X		S6, S20
TPS		X	X			X	S29, S34
BL		X					S36
M300		X					S64
PBSL		X					S44
PR		X					S31
POGIL		X					S71
TopD		X					S69

Note: MixMeth – Mixed Methodologies; FC – Flipped Classroom; GM – Gamification-Based Learning; PBL – Problem-Based Learning; GBL – Game-Based Learning; Aut-Meth – Authors' Methodologies; PjBL – Project-Based Learning; CL – Cooperative Learning; PP – Pair Programming; DOJO – Coding Dojo;TBL – Team-Based Learning; TPS – Think-Pair-Share; BL – Blended Learning; M300 – Method 300; PBSL – Project-Based Service Learning; PP – Pair Programming; POGIL – Process Oriented Guided Inquiry Learning; TopD – Top-Down.

In addition, we realized that the case study was the most used type of experiment and was used in conjunction with other types of investigation (e.g., surveys and interviews) as in publications S21, S29, S41, S46, and S48.

Observation, interviews, focus groups, and action-research experiments stood out in a smaller percentage. Our observations revealed that each technique was pivotal in research concerning adopting ALMs in programming education. The focus group approach provided an overview of group perspectives, allowing us to identify common trends and issues in studies S34. In contrast, studies S20 and S46 utilized the interview technique, provided a more in-depth exploration of individual experiences and revealed detailed and unique insights. Finally, research S77 using the action research technique enabled us to assess the implementation and evaluation of practical interventions, fostering continuous improvement in active teaching practices.

4.6. *SQ5. What Evaluation Metrics were Reported in the Publications?*

To answer SQ5, a qualitative analysis of the metrics was carried out about the ALMs adopted, which are presented from each accepted publication. The main objective of this analysis was to identify the metrics used by the instructors from the perspective of teaching to the adoption of ALMs. A list of all identified metrics was created to perform the qualitative analysis. Each of the metrics was listed, and based on the list, codes were created. Subsequently, these codes were analyzed and grouped according to their characteristics to form relevant concepts represented in this work through categories. It is noteworthy that a researcher-author performed the analysis. The identified metrics were then revised and discussed with another researcher-author with more than six years of experience in qualitative analysis.

Table 10 presents the main metrics, which are grouped according to the identified categories: Engagement, Performance, Motivation, Collaboration, and Interaction.

In the **Engagement** category, we observed that this metric generally represents why students felt more engaged in learning programming. Engagement refers to a work-related cognitive-affective state characterized by vigor, dedication, and absorption (Schaufeli and Bakker, 2003). The perception regarding engagement was identified when instructors adopted the following ALMs: GM (S19, S21, S54, S56, S62, S68), MixMeth (S15, S35), Auth-Meth (18), TBL (S20); TPS (S29), FC (S41), M300 (S64) and PBL (S78). Therefore, it can be seen that these ALMs contribute to awakening students to an active, creative, and collaborative posture, as they are engaged in teamwork, discuss issues during class, and seek to clarify their doubts.

The **Performance** category is related to performance in continuous assessment tasks such as key indicators, student progress, student grades after completing the course, rates, and averages obtained in activities, assessments, and final exams (Veerasamy *et al.*, 2020). In this category, the following ALMs stood out: MixMeth (S1, S5, S12, S15, S23, S26, S28, S35, S43, S73, S76, S80), FC (S14, S30, S36, S38, S40, S47, S50), PBL (S10, S58, S75, S80) and GM (S49, S54, S57, S60). These ALMs have significantly improved student performance due to new teaching strategies, which have shown con-

Table 10
Metrics X Methodology. Source: The authors

ALM	Engagement	Performance	Motivation	Collaboration	Interaction	#Publications
FC	X	X	X			S7, S14, S30, S36, S38, S40, S41, S47, S50, S74
MixMeth	X	X	X			S1, S5, S12, S15, S23, S26, S28, S35, S43, S65, S73, S76, S80
GM	X	X	X			S19, S21, S49, S54, S56, S57, S60, S62, S68, S21, S54, S56, S58, S60, S62, S68
GBL		X	X	X	X	S48, S11, S51, S55, S76
AutMeth	X	X	X			S16, S18, S17, S25
PjBL		X	X			S4, S66, S79
CL		X				S77
PP		X				S22, S46, S72
TBL	X	X		X		S6, S20
TPS	X	X				S29, S34
M300	X		X		X	S64
PBSL			X			S44
DOJO		X	X	X		S61, S63
PBL	X	X	X	X	X	S9, S10, S58, S75, S78, S80
TopD		X				S69
POGIL		X				S71

Note: FC – Flipped Classroom; MixMeth – Mixed Methodologies; GM – Gamification-Based Learning; GBL – Game-Based Learning; Aut-Meth – Authors' Methodologies; PjBL – Project-Based Learning; CL – Cooperative Learning; PP – Pair Programming; TBL – Team-Based Learning; TPS – Think-Pair-Share; M300 – Method 300; PBSL – Project-Based Service Learning; DOJO – Coding Dojo; PBL – Problem-Based Learning; TopD – Top-Down; POGIL – Process Oriented Guided Inquiry Learning.

siderable advantages in solving real problems while maintaining curiosity about technology (Wang *et al.*, 2019).

By definition, motivation explains the goals and how actively or intensely someone pursues them. This can be intrinsic motivation, which involves the individual in some task for the simple pleasure of performing it, or extrinsic motivation, which consists of the individual in activities for the rewards obtained by completing them or because such activities are necessary steps to achieve a specific objective (Souza and Bittencourt, 2019). The category **Motivation** is associated with how students felt when learning via the GM (S21, S54, S56, S58, S60, S62, S68), FC (S7, S40, S74), MixMeth (S15, S65), GBL (S51, S55) and Dojo (S61, S63) methodologies. We note that the motivation is reflected in an improvement in the student's attendance and class participation due to the challenges proposed to them to seek innovative ways of solving problems inside and outside the classroom.

Knowledge construction occurs via the exchange of experiences and the sharing of acquired knowledge. In this sense, it is observed that the DOJO (S61, S63), TBL (S20), JE (S11), and PBL (S9) were the methodologies that most contributed to the awakening of **Collaboration** among students and between students and instructors. In the case

of the DOJO, in addition to making the experience more fun, it promotes an inclusive, cooperative, and collaborative environment based on exchanging experiences and networking among participants (de Castro Junior *et al.*, 2021). This occurs because this ALM allows for improved classroom participation and knowledge exchange via collaboration in activities and discussions.

In the **Interaction**, the PBL (S9, S10), JE (S11), and M300 (S64) methodologies were the most reported to contributing to the awakening of interaction in the classroom, whether between students themselves or between the students and their instructors. We note that they all relate to improving or even awakening student interaction. They can contribute to developing professional skills such as broader communication, teamwork, and self-education. In addition, there is a discussion about improving programming skills such as problem-solving, understanding the basic functioning of programming languages, and the ability to read code (Nagai *et al.*, 2016).

Table 11 presents the metric **instructor's perception**. The instructor's perception metric is related to the subjective observations of instructors reported in the teaching and student learning studies. In this sense, the instructor's perception metric is related to their perception of knowledge and skill acquisition, students' perceptions of the effectiveness of studies, and students' views and performance, among others. Table 11 presents an overview of the reported perceptions since the perceptions regarding the students' effort are not objective (Aivaloglou and Meulen, 2021).

Table 11
Instructors' perception X Methodology. Source: The authors

Methodology	Instructors' perceptions	#Publication
MixMeth	Improvement of students' abilities, students' completion of a task.	S70
	Correcting errors and problem-solving within the given time frame.	S27
	Students' perceptions of the effectiveness.	S3
FC	Knowledge and skill acquisition.	S2
	Cognitive flexibility, problem-solving skills, flipped learning readiness levels in students' programming.	S24
CL	Peer evaluations and self-assessment.	S32
	Improvement in programming skills.	S33
PBL	Student behavior with a focus on the teaching-learning process, students' grades for the three PBL problems.	S13
	Theoretical evaluation (content), evaluation of the proposed solution (result), and evaluation of interpersonal skills.	S52
GBL	Willingness to solve problems, ability to generate alternatives, comparison between possible alternatives, evaluation of solutions.	S53
BL	Affection, skill, cognition.	S36
PjBL	Course organization and course quality, course difficulty level.	S39
PR	Students understood the concepts, and understanding was improving.	S31

Note: MixMeth – Mixed Methodologies; FC – Flipped Classroom; CL – Cooperative Learning; PBL – Problem-Based Learning; GBL – Game-Based Learning; BL – Blended Learning; PjBL – Project-Based Learning; PR Peer Review.

Given this scenario, it can be seen that the Performance metric highlights the methodologies MixMeth, FC, PBL, GM, and PjBL. However, we realized the GM and FC methodologies stand out regarding the Motivation metric. For the Collaboration and Interaction metrics, the DOJO and PBL stand out, respectively. Thus, there is an opportunity to improve instructional strategies for teaching, in addition to contributing to the understanding of taught concepts and the development of skills related to programming, which consequently contributes to the development of professional skills. Finally, the instructors' perception highlights the MixMeth, FC, CL, and PBL methodologies if we consider the advantages and construction of knowledge regarding group activities.

4.7. *SQ6. Which Technologies Were Used During the Teaching of Programming?*

To answer SQ6, we organized the technologies cited in the publications by the type of methodologies used for teaching programming. Table 12 presents the technologies reported in the publications.

Table 12
Technologies X Methodology. Source: The authors

Methodology	Technologies	#Publication
FC	Hands-on instruction, Tic-Tac-Toe, Grading, Tokens, Pearson MyProgrammingLab	S7
	Blackboard, videos, slides, textbooks	S8
	Video tutorials	S14
	App Inventor online editor, Edmodo, video	S24
	Video, interactive textbook, zyBooks platform	S30
	Video lectures, platforms online	S37
	Flash animations and video, Java Swing	S40
	Java Collections Framework and iterators, Eclipse, Java v1.7, JUnit v4, EclEmma, Jacoco, FindBugs, PMD, and CheckStyle, GitHub, Google Forms	S41
	Virtual learning environment	S42
	YouTube channel, video quizzes	S47
	MyProgrammingLab textbook, online quizzes, programming homework	S50
MixMeth	Google Classroom, Kahoot, video lectures	S1
	Stack Overflow, Javadoc or Google	S3
	CodeBlocks IDE, URI Judge Online, Sophia Learning tool	S12
	MOOC tool, PPT to the projection screen	S15
	Moodle	S23
	NSB AppStudio, commercial APIs (e.g., Google Maps, Yelp, Weather, etc.), Code Academy lessons, videos, Canvas	S26
	textbook, videos	S28
	Scratch game	S43
	Moodle, video from YouTube, Poll Everywhere tool	S46
	Moodle, the Multimedia Teaching-Learning Environment	S65

Continued on next page

Table 12 – continued from previous page

Methodology	Technologies	#Publication
GM	Interactive User Input, Cryptogram, Word Search, Puzzle Maker, Hangman	S19
	Framework de Werbach, UVa Online Judge	S54
	URI Online Judge	S57
	GameProgJF, Google Forms	S58
	code.org course, Kahoot, Socrative	S60
	cod[edu], Google Forms	S68
GBL	Textbook, video	S48
	Games: DSAsketch, Lightest and Heaviest, SAVG-Engine, Sorting Game, Sorting Casino, Sorting Game, Sortko	S51
	Games: Bullfrogs, An Eight-minute Empire, Carcassone, Metrocity, Resolution Inventory Social Problems	S53
	App Construct2	S67
PBL	Eclipse, NetBeans	S10
	Google Classroom, IDE JetBrains PyCharm	S52
PjBL	GUI tkinter	S4
	Video lectures, Canvas, Gitlab, Google App Engine, Google Cloud, CATME system	S39
	Junit	S66
CL	Github, Facebook, IntelliJ IDEA	S33
PP	Lectures, tutorials, demo sessions, homework assignments	S45
TBL	Quiz	S6
	Eclipse, NetBeans, JUnit, Javadoc	S20
TPS	Survey	S29
DOJO	IDE DevC++	S61
	Google Forms	S63

Note: FC – Flipped Classroom; MixMeth – Mixed Methodologies; GM – Gamification-Based Learning; GBL – Game-Based Learning; PBL – Problem-Based Learning; PjBL – Project-Based Learning; CL – Cooperative Learning; PP – Pair Programming; TBL – Team-Based Learning; TPS – Think-Pair-Share; DOJO – Coding Dojo.

It is essential to mention that not all publications presented the tools or technologies used. However, we noticed that the selected studies present different types of tools, ranging from devices known by the community, such as Google Classroom, Kahoot, video lectures, and GitHub, which were shown in publications S1, S33, and S52, to even lesscommon ones, such as tkinter GUI, App Inventor online editor, MyProgrammingLab, App Construct2, which were featured in S4, S24, S50, and S55. In addition, it is observed that not all publications reported or did not mention which technology was used in the study.

5. Discussion of the Results

We observed that instructors have been experimenting with different ALMs to improve their teaching abilities, which will reflect directly on the students' learning. In addition,

the community's concern regarding improvements in teaching programming is due to the needs and weaknesses still perceived in teaching. From this perspective, positive aspects are observed. Higher education has developed significantly over the last two decades. It has been influenced by two trends: advances in active learning methods and the integration of technology, which are much more than artifacts and applications.

In this context, the diverse scenario of ALMs experienced in teaching programming shows that the faculty seeks to motivate and engage students in programming studies, as it is known that teaching and learning programming is complex and challenging. In this context, it is observed that it is challenging to introduce innovations even when this would be advantageous and beneficial for teaching and learning programming, considering that teaching programming is still a challenge for instructors of computing courses (Raj *et al.*, 2018). However, adopting these ALMs makes it possible to minimize the challenges faced in the classroom for teaching and learning programming.

The results of this SMS are corroborated with the results of the literature, especially concerning the main ALMs mapped. The works by Berssanette and de Francisco (2021) and (Anicic and Stapic, 2022) present results that report adopting different ALMs in teaching and learning computer programming. These authors highlight methodologies that have been used by instructors in teaching programming, namely Coding Dojo (DOJO), Gamification (GM), Game-Based Learning (GBL), Project-Based Learning (PjBL), Problem-Based Learning (PBL), Flipped classroom (FC) and Peer Instruction (PI). The adoption of these ALMs reveals their concern for motivating and engaging students in programming classes. It is observed that instructors seek support in the ALMs to innovate in their teaching of programming.

Table 3 presents methodologies that were also listed in the research by Berssanette and de Francisco (2021) and Anicic and Stapic (2022), including approaches that instructors have implemented for active teaching and learning. Hendrik (2019) adopts two ALMs for teaching programming, the FC, which refers to the concept of role reversal in the classroom. The Flipped Classroom is "what is traditionally done in the classroom is now done at home, and what is traditionally done as homework is now done in the classroom" (Bergmann and Sams, 2012) and PjBL, which is "a teaching method that engages students in learning knowledge and skills through a structured extended inquiry process, complex real-life questions, and projected tasks" (Hallermann *et al.*, 2016).

We mapped four new methodologies implemented by the authors (S16, S17, S18, and S25) named Auth-Meth. The Auth-Meth are not widely used methodologies in the literature and are presented as new strategies for the teaching of programming. The work by Dol (2018) (S16) presents a combination of an animated flowchart with an example and TPS activities. The approach used to modify the TPS activities proved helpful in teaching algorithms. The work of Yuan and Cao (2019) (S25) shows a hybrid two-stage model in which a programming project is divided into two stages: the checkpoint stage (stage one) and the final submission stage (stage two) and the act of reviewing other people's code are found to improve student learning.

It is a challenge to plan classes that motivate students. However, motivation is considered an indispensable factor in carrying out any activity and, mainly, in learning. Faced with this challenge, ALMs are seen as an essential support and strategy for teaching

programming. In this context, Table 7 shows that the FC, MixMeth, GM, GBL, and PBL methodologies are more frequently addressed in the studies.

In this scenario, the MixMeth, GM, GBL, and PBL methodologies provide student learning that is generally based on projects and work in groups during their studies. According to Aivaloglou and Meulen (2021), there are several reasons for implementing group work, e.g., it offers students the opportunity to work on larger-scale software projects, and it can be used as an instructional strategy and is included in education because of its benefits for the domain-specific knowledge learning process. For Kirschner *et al.* (2018), there are also the benefits of collaboration when facilitating measures are taken, such as scripted learning environments, including rules for communication and coordination, in the classroom.

Table 12 summarizes the technologies that instructors have used. The FC, MixMeth, GM, and GBL methodologies use different technologies. It is observed that the technological support (whether digital or not) adopted for teaching programming was effective, mainly in implementing activities in the classroom, such as questionnaires and projects using different tools and applications. For Shokaliuk *et al.* (2020), the interaction with technologies and digital content provides a reflective and critical attitude in the face of its evolution and an ethical, safe, and responsible approach to using these tools. In this perspective, adopting ALMs and learning technologies, such as Kahoot or Google Classroom, is presented as effective in facilitating the teaching of programming. Even curious, open, and perspective in the face of its evolution, as well as an ethical, safe, and responsible approach to using these tools. In this perspective, adopting ALMs and learning tools (e.g., Kahoot or Google Classroom) is presented as effective in facilitating the teaching of programming.

In recent years, special attention has been focused on integrating digital technologies and games in education, and there is an increase in interest in using games as a tool to aid student learning (Grivokostopoulou *et al.*, 2016). In this context, the mapping results show that the studies used different games regarding GM and GBL methodologies, while methodologies like FC, MixMeth, PBL, and PjBL are used with online learning resources. The growing availability of online learning resources, such as tutorial web-sites (e.g., Codecademy.org, Kahn Academy), block programming environments (e.g., Scratch), and educational games (e.g., Swift Playgrounds), are popular choices for people to gain programming knowledge (Lee and Chiou, 2020).

ALMs and relevant technologies can aid instructors in teaching programming due to the possibility of involving students in classes. Students' engagement during their learning is essential for learning challenging subjects like computer programming. In particular, educational games have successfully taught introductory programming concepts (Lee and Chiou, 2020). However, even with the success of these resources, the student may encounter difficulties and not receive the necessary support to overcome the difficulties and may become frustrated (Lee and Chiou, 2020). Therefore, it is essential to look at student engagement, as it is crucial to student success (Marks, 2000) and, consequently, necessary for teaching programming.

Due to their unnatural syntax and semantics, computer programming languages are challenging for most first-year computer science students (Jeff and Nguyen, 2018). Giv-

en this, anattemptwas made to map the programminglanguages reported in the studies. Table 8 presents the various types of languages, and three types of programming language stand out for being the most used with most mapped ALMs. Java, Python, and C languages were the most reported in the studies, and these languages are among the main languages, according to surveys by Cass (2022).

Thus, using different languages with ALMs can significantly contribute to the programming teaching process and prove to be a viable alternative in teaching. Java is a popular language for developing Web applications. Java is the most-reported programming language in the studies and is used with the FC, MixMeth, GM, PBL, Auth-Meth, PjBL, CL, GBL, DOJO, and BL methodologies. Additionally, most studies reported using Python with the FC, MixMeth, GMm, PBL, PjBL, GBL, and DOJO methodologies. According to research by Cass (2022), since 2019, Python has been one of the main programming languages and at the top of the main programming languages until 2022. Another language that stood out in the studies was C, which was used with FC, MixMeth, PBL, CL, and Dojo methodologies. It can be used in different projects, such as creating applications. According to research by Cass (2022), C stands out among the main programming languages.

6. Why Are these Results Essential for an Educational Technology Proposal?

The results achieved in this SMS permitted us to identify and categorize the ALMs that instructors have adopted and revealed crucial positive evidence related to their use in teaching programming. On the other hand, it also shows that they are still little employed by instructors (Nguyen *et al.*, 2021). Lack of time for lesson planning (Eickholt, 2018; Michael, 2007), difficulty in complying with the entire content of the course (Eickholt, 2018), students' rejection of the use of new teaching methodologies, and lack of information on how to implement ALMs in classes (Tharayil *et al.*, 2018) are pointed out as to incorporate them into their teaching.

Based on that, we intend to develop an educational tool called CollabProg. CollabProg helps instructors to identify, select, adopt, discuss, comment, evaluate, and possibly collaborate with new (or existing) ALMs used during the teaching of programming. As a guide, we are using the Design Science Research (DSR) methodology (Wieringa, 2014; Vihavainen *et al.*, 2014) to help us develop CollabProg, a collaborative repository whose main objective is to aid instructors in adopting active methodologies while teaching programming content.

CollabProg will help the instructor to identify and choose ALMs that meet the pedagogical needs and follow their teaching context. In addition, it will provide a set of specific guidelines that will describe the steps for instructors to apply ALMs in the classroom. In this way, instructors will no longer need to search various books, articles, websites, or forums for ways to implement a specific ALM. The initial idea is for CollabProg to be available on a website on the Web so that instructors can access it. Fig. 4 shows the first version of CollabProg focusing on a specific active methodology, POGIL. Part 01 of

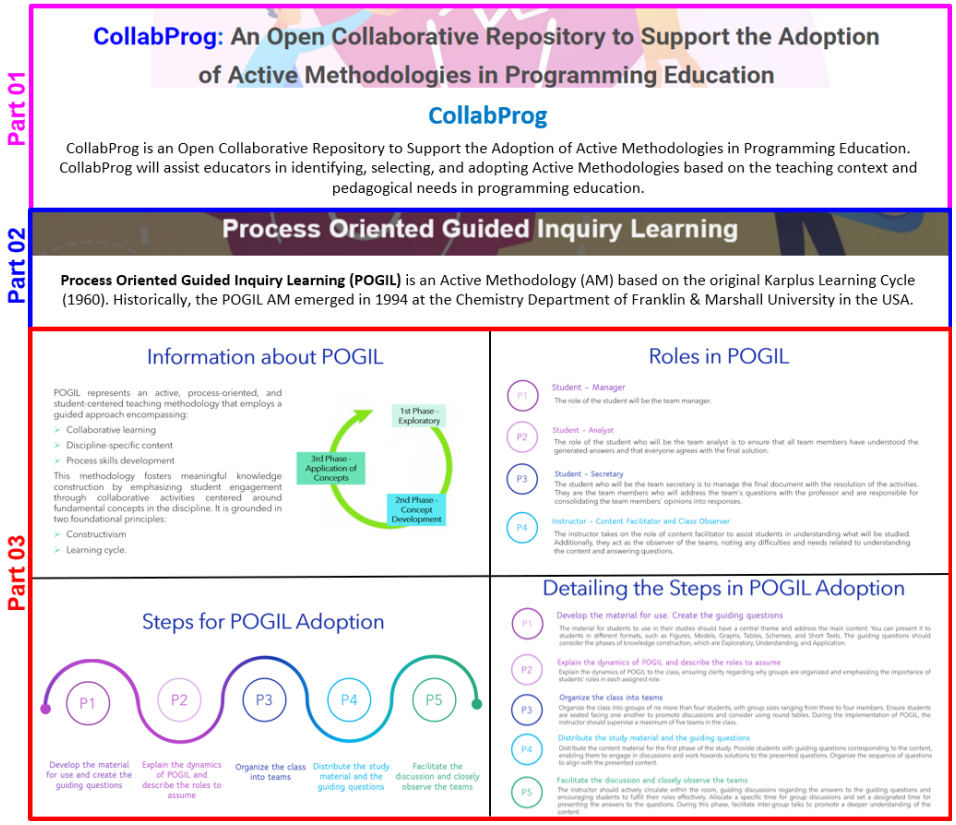


Fig. 4. First version of CollabProg. Source: The authors.

Fig. 4 provides a brief description of CollabProg, and Part 02 offers a concise overview of the chosen active methodology by the instructor, in this case, POGIL. Finally, Part 03 provides more detailed explanations of the methodology, including the roles within the method, the steps for adoption, and a breakdown of each step.

The website will contain further information to assist instructors in their teaching practice. The repository modules (menus) will displayed in sequence, and the instructor will not need to register to access the platform and have access to all its functions.

In CollabProg version 1, the repository is divided into three labeled menus. Each menu provides information for users to navigate, select, and adopt any available ALMs. Instructors can find a wealth of information on ALMs within CollabProg, including adoption examples, community-adopted tool options, real-world experiences, and feedback from other instructors. This platform provides valuable insights into both the positive and negative aspects of different ALMs. As a differential, unlike many other platforms, users don't need to register to access CollabProg – it's open to anyone.

In the main interface (Home), instructors will have access to **About**, which will present an overview of the CollabProg repository. In **Methodology**, a list of the ALMs

mapped from the SMS results will be presented. It is essential to highlight that not all methodologies identified here may be available on CollabProg. We will conduct a previous evaluation of all ALMs and, through pre-established evaluation criteria, a curation of methodologies with steps defined in the literature to direct their implementation in the classroom. This curatorship will be very important, as it will be through it that only methodologies that have well-defined steps and that can be reproduced by other instructors, regardless of their teaching context, will be highlighted.

In **Recommendation** (How to adopt menu), the instructor will provide characteristics about the class, the content to be taught, and the discipline, among others, so that CollabProg can recommend the ALM that best suits the scenario informed by the instructor. Based on CollabProg's recommendation, the intention is to present the step-by-step instructions for using the ALM, information, and the roles to be assumed by participants during the methodology implementation, suggestions for activities, and tool support options that are available and have been adopted by the community.

As it is a collatborative and open repository, in the **Register methodology** menu, the instructor will have an open space to share a new ALM or an adaptation of one already implemented or tested for teaching programming. The **Experiences** menu will be a space for the community to share their experiences, suggestions, and evaluations of ALMs in different educational settings. In addition, the results of the achieved learning objectives and the positive and negative points about the adopted ALM will also be presented. In this way, other instructors can consult the advantages or disadvantages of using a particular ALM, thus helping them choose the methodology. Finally, **Contact** will be the means of communication between the researchers involved in the development of the platform and the academic community, who will be able to get in touch via the authors' e-mails to report errors, problems, or suggestions for the repository.

To classify the ALMs that will be part of CollabProg, we intend to group the knowledge about each methodology in a conceptual model inspired by those proposed by Sobrinho *et al.* (2016). We will initially define the domain and scope of knowledge built from the SMS results. According to Sobrinho *et al.* (2016), the domain is the semantic representation and formalization of teaching methodologies based on active learning principles. This model's scope is to support instructors in teaching programming in higher education through organized and semantically represented knowledge, thus facilitating its dissemination and active methodologies. This way, we will structure the information collected from the ALMs in a conceptual model, represented using the class diagram shown in Fig. 5.

In the model, the class **Category** represents the category of active methodologies according to the approach of the method. This class is associated with the **Methodology** class, which represents the active methodologies that will compose CollabProg. As we observed in the SMS, the methodologies can be used together to improve or complement the positive results of teaching programming. The self-relationship represents this possibility in the Methodology class. The **Step** class represents the necessary steps for adopting methodologies. The **Activity** class describes the activities to be carried out in the steps for implementing the methodologies in the classroom, which can be planning the

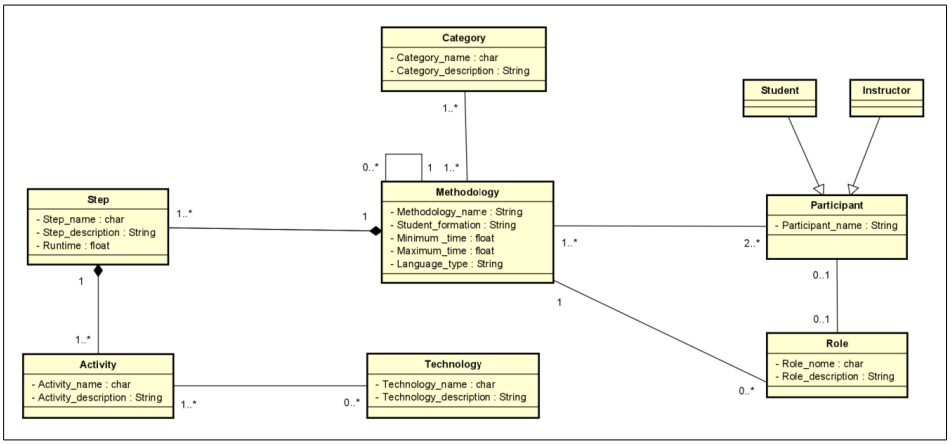


Fig. 5. Conceptual model of CollabProg. Source: The authors.

content and explaining the methodology and the roles, among others. The **Technology** class represents the possible educational technologies that can be used and employed for each activity, whether a virtual environment or a game. Finally, to define the roles to be followed and that exist in the methodologies, the **Participant** and **Role** classes are associated with each other and related to the Methodology class.

To better explain and develop the elaborated conceptual model, a recommendation system will be developed based on knowledge of the methodologies presented in the conceptual model. Thus, based on the answers provided by the instructors in the questionnaire, the recommendation system will provide a set of methodologies according to the needs of the instructor interested in applying them. This recommendation system will be part of CollabProg and will be available in the **Recommendation** menu, described in the information architecture.

Regarding the trusteeship of the contents that will be shared on CollabProg, in general, the perspective is that a screening process be carried out to guarantee the reliability of the contents presented so that there is adoption and effective use of ALMs in the teaching of programming. In addition, for curation, the researchers involved will propose criteria that will evaluate the contents made available in the repository to avoid any frustrations of the users who will use the repository.

To assess the feasibility of using and developing CollabProg, we intend to conduct quantitative experimental studies via questionnaires and using the Technology Acceptance Model (TAM) and semi-structured interviews. In addition, qualitative studies are planned that involve case studies, focus group sessions, and interviews with instructors in the area to understand the context in which instructors work (Manotas *et al.*, 2016). The goal is to conduct studies with instructors from public or private higher education institutions and in classes that deal with computer programming content, whether in courses for beginners or not.

We expected that CollabProg would be a technological aid that would bring together, in a single Internet portal, strategies on how to conduct the adoption of dif-

ferent ALMs for teaching programming and will provide examples, suggestions for activities, support options, and tools adopted by the computer science education community, as well as experiences on the adoption of methodologies in different scenarios, results achieved by other instructors and positive and negative points about the ALM adopted.

7. Threats to Validity

Despite the care in defining the SMS protocol as per Kitchenham (2012) and its systematic application, it can be observed that this research suffers from some well-known limitations and threats to its validity. However, to mitigate the impact of factors that may affect the validity of this SMS, several strategies were adopted for constructing the search string for selecting and extracting data from the publications. According to Ampatzoglou *et al.* (2019), several threats to validity can occur in an SMS. Among the most common is the search string construction, which we sought to mitigate using a string carefully constructed to include all potentially relevant publications.

In terms of threats to selecting relevant instructional units and data extraction, these were mitigated by the definition and documentation of a rigorous protocol. The careful establishment of inclusion and exclusion criteria and discussion among the authors until consensus was reached. As study inclusion/exclusion bias is a common threat to validity, an attempt was made to mitigate this threat by carrying out an inclusion and exclusion process by two researchers, who held weekly meetings to discuss each article, especially those that did not fit the criterion applied.

Finally, another prevalent threat in studies is data extraction bias, mitigated by defining possible answers for each question in the protocol before extraction. In addition, data extraction was performed by the first author, inferred when not explicitly indicated in the article, and carefully reviewed by the co-authors. Finally, selecting digital libraries and annals to search for publications is another validity threat we sought to mitigate. Therefore, to avoid this problem, we selected libraries and events that are known and widely used in computer science.

8. Conclusion and Future Work

After analyzing the data extracted from the publications selected for this research, the state of the art regarding adopting ALMs in teaching computer programming was characterized. It is essential to mention that this characterization can help in the development of new research since the selection of different methodologies that can be used and improved in teaching practice will, therefore, support the knowledge and construction of new research that aims to test or create new methods that help the instructors in teaching programming.

Thus, the importance of seeking strategies to support instructors in teaching and motivating students to learn programming is highlighted since this is a significant fac-

tor for successful instruction. This factor is especially relevant in collaborative learning contexts, where social interaction is critical in adopting ALMs (Serrano-Cámara *et al.*, 2014).

As future work, the aim is to curate and categorize the ALMs mapped here so that instructors can compose an open, collaborative repository in which they can identify, select, adopt, discuss, comment, evaluate, and possibly collaborate with new (or existing) ALMs are used while teaching programming. The repository will help the instructors identify and choose ALMs according to their teaching context to meet their pedagogical needs. Therefore, from the curation of the mapped ALMs, it will be possible to build and make available a set of step-by-step guidelines to aid instructors during the adoption of the ALMs. In this way, the instructors will not need to search various scientific articles or books for ways to carry out a particular ALM in the classroom.

Funding

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Finance Code 001. This work was partially supported by Amazonas State Research Support Foundation – FAPEAM – through the POSGRAD project. Williamson Silva thanks FAPERGS for the financial support granted through ARD/ARC Project (process 22/2551-0000606-0). Ivanilse Calderon thanks the Federal Institute of Rondônia (IFRO).

References

ACM, IEEE (2013). Computer science curricula 2013. `https://www.acm.org/binaries/content/assets/education/cs2013_web_final.pdf`

Agapito, J.L., Rodrigo, M., Mercedes, T. (2018). Investigating the impact of a meaningful gamification-based intervention on novice programmers' achievement. In: *International Conference on Artificial Intelligence in Education*, pp. 3–16. Springer.

Ahshan, R. (2021). A framework of implementing strategies for active student engagement in remote/online teaching and learning during the COVID-19 pandemic. *Education Sciences*, 11(9), 483.

Aivaloglou, E., Meulen, A.v.d. (2021). An empirical study of students' perceptions on the setup and grading of group programming assignments. *ACM Transactions on Computing Education (TOCE)*, 21(3), 1–22.

Aksit, F., Niemi, H., Nevgi, A. (2016). Why is active learning so difficult to implement: The Turkish case. *Australian Journal of Teacher Education (Online)*, 41(4), 94–109.

Alves, G., Rebouças, A., Scaico, P. (2019). Coding dojo como prática de aprendizagem colaborativa para apoiar o ensino introdutório de programação: Um estudo de caso. In: *Anais do XXVII Workshop sobre Educação em Computação*, pp. 276–290. SBC.

Amira, T., Lamia, M., Hafidi, M. (2019). Implementation and evaluation of flipped algorithmic class. *International Journal of Information and Communication Technology Education (IJICTE)*, 15(1), 1–12.

Ampatzoglou, A., Bibi, S., Avgeriou, P., Verbeek, M., Chatzigeorgiou, A. (2019). Identifying, categorizing and mitigating threats to validity in software engineering secondary studies. *Information and Software Technology*, 106, 201–230.

Anicic, K.P., Stapic, Z. (2022). Teaching Methods in Software Engineering: Systematic Review. *IEEE Software*.

Araújo, E.A., Furtado C C, J., Alexandre S H, G. (2020). Jogos de tabuleiros modernos para aprimorar a resolução de problemas em alunos de programação. In: *XIX Simposio Brasileiro de Jogos e Entretenimento Digital (SBgames 2020)*.

Astrachan, O.L., Duvall, R.C., Forbes, J., Rodger, S.H. (2002). Active learning in small to large courses. In: *32nd Annual Frontiers in Education* (Vol. 1), pp. 2–2. IEEE.

Avouris, N., Kaxiras, S., Koufopavlou, O., Sgarbas, K., Stathopoulou, P. (2010). Teaching introduction to computing through a project-based collaborative learning approach. In: *2010 14th Panhellenic Conference on Informatics*, pp. 237–241. IEEE.

Barnes, T., Powell, E., Chaffin, A., Lipford, H. (2008). Game2Learn: improving the motivation of CS1 students. In: *Proceedings of the 3rd international conference on Game development in computer science education*, pp. 1–5.

Battistella, P.E., Wangenheim, C.G.v., Wangenheim, A.v., Martina, J.E. (2017). Design and large-scale evaluation of educational games for teaching sorting algorithms. *Informatics in Education*, 16(2), 141–164.

Beck, L.L., Chizhik, A.W. (2006). Workshop Applying Cooperative Learning Methods in Teaching Computer Programming. In: *Proceedings. Frontiers in Education. 36th Annual Conference*, pp. 1–2. IEEE.

Ben-Ari, M. (2001). Constructivism in computer science education. *Journal of computers in Mathematics and Science Teaching*, 20(1), 45–73.

Bergmann, J., Sams, A. (2012). *Flip your classroom: Reach every student in every class every day*. International society for technology in Education, ???.

Berssanette, J.H., de Francisco, A.C. (2021). Active learning in the context of the teaching/learning of computer programming: A systematic review. *Journal of Information Technology Education. Research*, 20, 201.

Bittencourt, R.A., Rodrigues, C.A., Cruz, D.S.S. (2013). Uma experiência integrada de programaçao orientada a objetos, estruturas de dados e projeto de sistemas com pbl. In: *XXXIII Congresso da SBC–XXI WEI*.

Boudia, C., Bengueddach, A., Haffaf, H. (2019). Collaborative strategy for teaching and learning object-oriented programming course: A case study at Mostafa Stambouli Mascara University, Algeria. *Informatica*, 43(1).

Bowman, N.A., Jarratt, L., Culver, K., Segre, A.M. (2021). The impact of pair programming on college students' interest, perceptions, and achievement in computer science. *ACM Transactions on Computing Education*, 21(3), 1–19.

Brescia, W., Mullins, C., Miller, M.T. (2009). Project-based Service-Learning in an Instructional Technology Graduate Program. *International Journal for the Scholarship of Teaching and Learning*, 3(2), 2.

Brito, P., Fortes, R., Faria, F., Lopes, R.A., Santos, V., Magalhães, F. (2019). Programaçao competitiva como ferramenta de apoio ao ensino de algoritmos e estrutura de dados para alunos de ciência da computaçao. In: *Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação-SBIE)* (Vol. 30), p. 359.

Caceffo, R., Gama, G., Azevedo, R. (2018). Exploring active learning approaches to computer science classes. In: *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*, pp. 922–927.

Cao, L., Grabchak, M. (2019). Interactive preparatory work in a flipped programming course. In: *Proceedings of the ACM Conference on Global Computing Education*, pp. 229–235.

Casarotto, R.I., Bernardi, G., Cordenonsi, A.Z., Medina, R.D. (2018). Logirunner: um Jogo de Tabuleiro como Ferramenta para o Auxílio do Ensino e Aprendizagem de Algoritmos e Lógica de Programação. *RENOTE*, *16*(1).

Cass, S. (2022). The Top Programming Languages 2022: Python's still No. 1, but employers love to see SQL skills. *IEEE Spectrum*.

Chandrasekaran, S., Badwal, P., Thirunavukkarasu, G., Littlefair, G. (2016). Collaborative learning experience of students in distance education. In: *International Symposium on Project Approaches in EngineeringEducation* (Vol. 6), pp. 90–99.

Chao, P.-Y. (2016). Exploring students' computational practice, design and performance of problem-solving through a visual programming environment. *Computers & Education*, 95, 202–215.

Cohen, J. (1960). A coefficient of agreement for nominal scales. *Educational and psychological measurement*, 20(1), 37–46.

Corritore, C.L., Love, B. (2020). Redesigning an Introductory Programming Course to Facilitate Effective Student Learning: A Case Study. *Journal of Information Technology Education: Innovations in Practice*, 19, 091–135.

Costa, A.F.F., de Melo, A.F.M.F., Moreira, G.G., Carvalho, M.d.A., Lima, M.V.d.A. (2017). Aplicaçao de sala invertida e elementos de gamificaçao para melhoria do ensino-aprendizagem em programaçao orientada a objetos. TISE.

Creswell, J.W., Shope, R., Plano Clark, V.L., Green, D.O. (2006). How interpretive qualitative research extends mixed methods research. *Research in the Schools*, 13(1), 1–11.

da Silva, T.S.C., de Melo, J.C.B., Tedesco, P.C.d.A.R. (2018). Um modelo para promover o engajamento estudantil no aprendizado de programação utilizando gamification. *Revista Brasileira de Informática na Educação*, 26(03), 120.

da Silva Garcia, F.W., Oliveira, S.R.B. (2022). Aplicação de um Plano de Ensino para Disciplina de Algoritmos com Metodologias Ativas: Um Relato de Estudo de Caso Piloto. In: *Anais do XXXIII Simpósio Brasileiro de Informática na Educação*, pp. 301–310. SBC.

de Andrade, T.L., Rigo, S.J., Barbosa, J.L.V. (2021). Active methodology, educational data mining and learning analytics: A systematic mapping study. *Informatics in Education*, 20(2), 171.

de Azevêdo Silva, M.A., Dantas, A. (2014). KLouro: Um jogo educacional para motivar alunos iniciantes em programação. In: *Brazilian Symposium on Computers in Education* (Vol. 25), p. 702.

de Castro Junior, A.A., Cheung, L.M., Batista, E.J.S., de Lima, A.C. (2021). Uma Análise Preliminar da Aplicação do Método 300 em Turmas de Algoritmos e Programação. In: *AnaisdoXXIXWorkshopsobre-Educação em Computação*, pp. 171–180. SBC.

de Oliveira Fassbinder, A.G., Botelho, T.G.G., Martins, R.J., Barbosa, E.F. (2015). Applying flipped classroom and problem-based learning in a CS1 course. In: *2015 IEEE Frontiers in Education Conference (FIE)*, pp. 1–7. IEEE.

Desai, P., Meena, S., Giraddi, S., Desai, S., Hanchinamani, G. (2021). Transformation in Course Delivery Augmented with Problem-Based Learning and Tutorial. In: *2021 World Engineering Education Forum/ Global Engineering Deans Council (WEEF/GEDC)*, pp. 15–22. IEEE.

Dicheva, D., Hodge, A. (2018). Active learning through game play in a data structures course. In: *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*, pp. 834–839.

Dol, S.M. (2018). Animated flowchart with example followed by think-pair-share activity for teaching algorithms of engineering courses. In: *2018 IEEE Tenth International Conference on Technology for Education (T4E)*, pp. 186–189. IEEE.

dos Santos, S.C., Santana, E., Santana, L., Rossi, P., Cardoso, L., Fernandes, U., Carvalho, C., Torres, P. (2018). Applying PBL in teaching programming: an experience report. In: *2018 IEEE Frontiers in Education Conference (FIE)*, pp. 1–8. IEEE.

Drini, M. (2018). Using new methodologies in teaching computer programming. In: *2018IEEEIntegratedSTEM Education Conference (ISEC)*, pp. 120–124. IEEE.

Durak, H.Y. (2020). Modeling different variables in learning basic concepts of programming in flipped classrooms. *Journal of Educational Computing Research*, 58(1), 160–199.

Edwards, J.M., Fulton, E.K., Holmes, J.D., Valentin, J.L., Beard, D.V., Parker, K.R. (2018). Separation of syntax and problem solving in Introductory Computer Programming. In: *2018 IEEE Frontiers in Education Conference (FIE)*, pp. 1–5. IEEE.

Eickholt, J. (2018). Barriers to active learning for computer science faculty. *arXiv preprint arXiv:1808.02426*.

Elmaleh, J., Shankararaman, V. (2017). Improving student learning in an introductory programming course using flipped classroom and competency framework. In: *2017 IEEE Global Engineering Education Conference (EDUCON)*, pp. 49–55. IEEE.

Elnagar, A., Ali, M. (2012). A modified team-based learning methodology for effective delivery of an introductory programming course. In: *Proceedings of the 13th annual conference on Information technology education*, pp. 177–182.

Finger, A.F., da Silva, J.P.S., Ecar, M. (2021). Utilizando Aprendizado Baseado em Problemas para o Ensino de Paradigmas de Programação. In: *Anais do XXXII Simpósio Brasileiro de Informática na Educação*, pp. 135–144. SBC.

Freeman, S., Eddy, S.L., McDonough, M., Smith, M.K., Okoroafor, N., Jordt, H., Wenderoth, M.P. (2014). Active learning increases student performance in science, engineering, and mathematics. *Proceedings of the national academy of sciences*, 111(23), 8410–8415.

Gamage, L.N. (2021). A bottom-up approach for computer programming education. *Journal of Computing Sciences in Colleges*, 36(7), 66–75.

Garcia, F.W.D.S., Carvalho, E.D.C., Oliveira, S.R.B. (2021). Use of active methodologies for the development of a teaching plan for the algorithms subject. In: *2021 IEEE Frontiers in Education Conference (FIE)*, pp. 1–9. IEEE.

Garcia, F.W.d.S., Oliveira, S.R.B., Carvalho, E.d.C. (2022). A second experimental study the application of a teaching plan for the algorithms subject in an undergraduate course in computing using active methodologies. *Informatics in Education*, 22(2), 233–255.

Gonçalves, B., Nascimento, E., Monteiro, E., Portela, C., Oliveira, S. (2019). Elementos de Gamificação Aplicados no Ensino-Aprendizagem de Programação Web. In: *Anais do XXVII Workshop sobre Educação em Computação*, pp. 1–10. SBC.

Grivokostopoulou, F., Perikos, I., Hatzilygeroudis, I. (2016). An educational game for teaching search algorithms. In: *International Conference on Computer Supported Education* (Vol. 3), pp. 129–136. SCITEPRESS.

Hallermann, S., Larmer, J., Mergendoller, J.R. (2016). *PBL in the elementary grades: step-by-step guidance, tools and tips for standards-focused K-5 projects*. Buck Institute for Education, ???.

Hativa, N. (2001). *Teaching for effective learning in higher education*. Springer Science & Business Media, ???.

Hayashi, Y., Fukamachi, K.-I., Komatsugawa, H. (2015). Collaborative learning in computer programming courses that adopted the flipped classroom. In: *2015 International Conference on Learning and Teaching in Computing and Engineering*, pp. 209–212. IEEE.

Heckman, S.S. (2015). An empirical study of in-class laboratories on student learning of linear data structures. In: *Proceedings of the Eleventh Annual International Conference on International Computing Education Research*, pp. 217–225.

Hendrik, H. (2019). Flipping Web Programming Class: Student's Perception and Performance. In: *Proceedings of the 11th International Conference on Engineering Education (ICEED)*, pp. 31–45.

Herala, A., Vanhala, E., Nikula, U. (2015). Object-oriented programming course revisited. In: *Proceedings of the 15th Koli Calling Conference on Computing Education Research*, pp. 23–32.

Hidayati, N., Hariyadi, T., Praheto, B., Kusnita, S., Darmuki, A. (2023). The effect of cooperative learning model with think pair share type on speaking skill. *International Journal of Instruction*, 16(3), 935–950.

Hijon-Neira, R., Velazquez-Iturbide, A., Pizarro-Romero, C., Carriço, L. (2014). Serious games for motivating into programming. In: *2014 IEEE Frontiers in Education Conference (FIE) Proceedings*, pp. 1–8. IEEE.

Hu, H.H., Shepherd, T.D. (2013). Using POGIL to help students learn to program. *ACM Transactions on Computing Education (TOCE)*, 13(3), 1–23.

Hu, H.H., Shepherd, T.D. (2014). Teaching CS 1 with POGIL activities and roles. In: *Proceedings of the 45th ACM technical symposium on Computer science education*, pp. 127–132.

Imbulpitiya, A., Kodagoda, N., Gamage, A., Suriyawansa, K. (2020). Using active learning integrated with pedagogical aspects to enhance student's learning experience in programming and related concepts. In: *International Conference on Interactive Collaborative Learning*, pp. 218–228. Springer.

Jeff, B., Nguyen, K. (2018). ADL-Algorithmic design language. In: *2018 International Conference on Computational Science and Computational Intelligence (CSCI)*, pp. 651–654. IEEE.

Jonassen, D., Davidson, M., Collins, M., Campbell, J., Haag, B.B. (1995). Constructivism and computer-mediated communication in distance education. *American journal of distance education*, 9(2), 7–26.

Jonsson, H. (2015). Using flipped classroom, peer discussion, and just-in-time teaching to increase learning in a programming course. In: *2015 IEEE Frontiers in Education Conference (FIE)*, pp. 1–9. IEEE.

Joshi, A., Schmidt, M., Panter, S., Jain, A. (2020). Evaluating the benefits of team-based learning in a systems programming class. In: *2020 IEEE Frontiers in Education Conference (FIE)*, pp. 1–7. IEEE.

Joshi, N., Lau, S.-K. (2023). Effects of process-oriented guided inquiry learning on approaches to learning, long-term performance, and online learning outcomes. *Interactive Learning Environments*, 31(5), 3112–3127.

Kane, L. (2007). Educators, learners and active learning methodologies. *International journal of lifelong education*.

Katona, J., Kovari, A. (2016). A brain–computer interface project applied in computer engineering. *IEEE Transactions on Education*, 59(4), 319–326.

Kaya, O.S., Ercag, E. (2023). The impact of applying challenge-based gamification program on students' learning outcomes: Academic achievement, motivation and flow. *Education and Information Technologies*, 1–26.

Kelleher, C., Pausch, R. (2005). Lowering the barriers to programming: A taxonomy of programming environments and languages for novice programmers. *ACM Computing Surveys (CSUR)*, 37(2), 83–137.

Kholijah, G., Rarasati, N., Sormin, C., Aryanto, F. (2023). Project Based Learning Model in Computer Programming Courses at Mathematics Student. *IJER (Indonesian Journal of Educational Research)*, 8(1), 36–42.

Kinnunen, P., Malmi, L. (2006). Why students drop out CS1 course? In: *Proceedings of the second international workshop on Computing education research*, pp. 97–108.

Kirschner, P.A., Sweller, J., Kirschner, F., Zambrano R, J., *et al.*(2018). From cognitive load theory to collaborative cognitive load theory. *International Journal of Computer-Supported Collaborative Learning*, 13(2), 213–233.

Kitchenham, B.A. (2012). Systematic review in software engineering: where we are and where we should be going. In: *Proceedings of the 2nd international workshop on Evidential assessment of software technologies*, pp. 1–2.

Kong, S.-C., Lai, M., Sun, D. (2020). Teacher development in computational thinking: Design and learning outcomes of programming concepts, practices and pedagogy. *Computers & Education*, 151, 103872.

Kothiyal, A., Murthy, S., Iyer, S. (2014). Think-pair-share in a large CS1 class: does learning really happen? In: *Proceedings of the 2014 conference on Innovation & technology in computer science education*, pp. 51–56.

Kuhrmann, M., Fernández, D.M., Daneva, M. (2017). On the pragmatic design of literature studies in software engineering: an experience-based guideline. *Empirical software engineering*, 22(6), 2852–2891.

Kumar, M., Renumol, V., Murthy, S. (2018). Flipped classroom strategy to help underachievers in java programming. In: *2018 International Conference on Learning and Teaching in Computing and Engineering (LaTICE)*, pp. 44–49. IEEE.

Kurkovsky, S. (2013). Mobile game development: improving student engagement and motivation in introductory computing courses. *Computer Science Education*, 23(2), 138–157.

Lacher, L.L., Jiang, A., Zhang, Y., Lewis, M.C. (2018). Including Coding Questions in Video Quizzes for a Flipped CS1. In: *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*, pp. 574–579.

Lang, J., Nugent, G.C., Samal, A., Soh, L.-K. (2006). Implementing CS1 with embedded instructional research design in laboratories. *IEEE Transactions on Education*, 49(1), 157–165.

Lee, M.J., Chiou, J. (2020). Animated hints help novices complete more levels in an educational programming game. *Journal of computing sciences in colleges*, *35*(8).

Li, W., Liu, C.-Y., Tseng, J.C. (2023). Effects of the interaction between metacognition teaching and students' learning achievement on students' computational thinking, critical thinking, and metacognition in collaborative programming learning. *Education and Information Technologies*, 1–25.

Lister, R., Adams, E.S., Fitzgerald, S., Fone, W., Hamer, J., Lindholm, M., McCartney, R., Moström, J.E., Sanders, K., Seppälä, O., *et al.*(2004). A multi-national study of reading and tracing skills in novice programmers. *ACM SIGCSE Bulletin*, 36(4), 119–150.

Loftsson, H., Matthíasdóttir, Á. (2019). Using flipped classroom and team-based learning in a first-semester programming course: An experience report. In: *2019 IEEE International Conference on Engineering, Technology and Education (TALE)*, pp. 1–6. IEEE.

Luxton-Reilly, A., Albluwi, I., Becker, B.A., Giannakos, M., Kumar, A.N., Ott, L., Paterson, J., Scott, M.J., Sheard, J., Szabo, C. (2018). Introductory programming: a systematic literature review. In: *Proceedings Companion of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education*, pp. 55–106.

Manotas, I., Bird, C., Zhang, R., Shepherd, D., Jaspan, C., Sadowski, C., Pollock, L., Clause, J. (2016). An empirical study of practitioners' perspectives on green software engineering. In: *Proceedings of the 38th international conference on software engineering*, pp. 237–248.

Marks, H.M. (2000). Student engagement in instructional activity: Patterns in the elementary, middle, and high school years. *American educational research journal*, 37(1), 153–184.

Mayfield, C., Moudgalya, S.K., Yadav, A., Kussmaul, C., Hu, H.H. (2022). POGIL in CS1: Evidence for Student Learning and Belonging. In: *Proceedings of the 53rd ACM Technical Symposium on Computer Science Education V. 1*, pp. 439–445.

McCracken, M., Almstrum, V., Diaz, D., Guzdial, M., Hagan, D., Kolikant, Y.B.-D., Laxer, C., Thomas, L., Utting, I., Wilusz, T. (2001). Report by the ITiCSE 2001 Working Group on Assessment of Programming Skills of First-year CS Students. *Distribution*, 33(4), 125–180.

Melo, S., Soares Neto, C.d.S. (2017). Game of code: desenvolvimento e avaliação de uma atividade gamificada para disciplinas de programação. In: *XVI Simposio Brasileiro de Jogos e Entretenimento Digital (SBgames 2017)*.

Mendes, E., Wohlin, C., Felizardo, K., Kalinowski, M. (2020). When to update systematic literature reviews in software engineering. *Journal of Systems and Software*, 167, 110607.

Michael, J. (2007). Faculty perceptions about barriers to active learning. *College teaching*, 55(2), 42–47.
Michaličková, V. (2021). Using Online Forums to Promote Collaborative Learning in Introductory Programming Courses. In: *7th International Conference on Higher Education Advances (HEAd'21)*, pp. 145–152. Editorial Universitat Politècnica de València.
Nagai, W., Izeki, C., Dias, R. (2016). Experiência no uso de ferramentas online gamificadas na introdução à programação de computadores. In: *Anais do Workshop de Informática na Escola* (Vol. 22), pp. 301–310.
Nagappan, N., Williams, L., Ferzli, M., Wiebe, E., Yang, K., Miller, C., Balik, S. (2003). Improving the CS1 experience with pair programming. *ACM Sigcse Bulletin*, 35(1), 359–362.
Nakamura, W.T., de Oliveira, E.C., de Oliveira, E.H., Redmiles, D., Conte, T. (2022). What factors affect the UX in mobile apps? A systematic mapping study on the analysis of app store reviews. *Journal of Systems and Software*, 193, 111462.
Nasir, U. (2023). Using Architectural Kata in Software Architecture Course: An Experience Report. In: *Proceedings of the 5th European Conference on Software Engineering Education*, pp. 215–219.
Nguyen, K.A., Borrego, M., Finelli, C.J., DeMonbrun, M., Crockett, C., Tharayil, S., Shekhar, P., Waters, C., Rosenberg, R. (2021). Instructor strategies to aid implementation of active learning: a systematic literature review. *International Journal of STEM Education*, 8, 1–18.
Özyurt, H., Özyurt, Ö. (2018). Analyzing the effects of adapted flipped classroom approach on computer programming success, attitude toward programming, and programming self-efficacy. *Computer Applications in Engineering Education*, 26(6), 2036–2046.
Paristiowati, M., Rahmawati, Y., Fitriani, E., Satrio, J.A., Putri Hasibuan, N.A. (2022). Developing Preservice Chemistry Teachers' Engagement with Sustainability Education through an Online Project-Based Learning Summer Course Program. *Sustainability*, 14(3), 1783.
Park, E.L., Choi, B.K. (2014). Transformation of classroom spaces: Traditional versus active learning classroom in colleges. *Higher Education*, 68(5), 749–771.
Parsons, P. (2011). Preparing computer science graduates for the 21st Century. *Teaching Innovation Projects*, 1(1).
Petri, G., von Wangenheim, C.G. (2017). How games for computing education are evaluated? A systematic literature review. *Computers & education*, 107, 68–90.
Pollock, L., Jochen, M. (2001). Making parallel programming accessible to inexperienced programmers through cooperative learning. *ACM SIGCSE Bulletin*, 33(1), 224–228.
Qian, M., Clark, K.R. (2016). Game-based Learning and 21st century skills: A review of recent research. *Computers in human behavior*, 63, 50–58.
Rahman, F. (2018). Integrating project-based learning in mobile development course to enhance student learning experience. In: *Proceedings of the 19th Annual SIG Conference on Information Technology Education*, pp. 1–6.
Raj, A.G.S., Patel, J., Halverson, R. (2018). Is More Active Always Better for Teaching Introductory Programming? In: *2018 International Conference on Learning and Teaching in Computing and Engineering (LaT-ICE)*, pp. 103–109. IEEE.
Rajaravivarma, R. (2005). A games-based approach for teaching the introductory programming course. *ACM SIGCSE Bulletin*, 37(4), 98–102.
Raposo, E.H.S., Dantas, V. (2016). O Desafio da Serpente-Usando gamification para motivar alunos em uma disciplina introdutória de programação. In: *Brazilian Symposium on Computers in Education* (Vol. 27), p. 577.
Ribeiro, A.L., Bittencourt, R.A. (2018). A pbl-based, integrated learning experience of object-oriented programming, data structures and software design. In: *2018 IEEE Frontiers in Education Conference (FIE)*, pp. 1–9. IEEE.
Ribeiro, A.L., Bittencourt, R.A. (2019). A case study of an integrated programming course based on PBL. In: *2019 IEEE Frontiers in Education Conference (FIE)*, pp. 1–9. IEEE.
Rosiene, C.P., Rosiene, J.A. (2015). Flipping a programming course: The good, the bad, and the ugly. In: *2015 IEEE Frontiers in Education Conference (FIE)*, pp. 1–3. IEEE.
Safana, A.I., Nat, M. (2019). Students' Perception of a Blended Learning Approach in an African Higher Institution. *J. Univers. Comput. Sci.*, 25(5), 515–540.
Schaufeli, W.B., Bakker, A.B. (2003). Utrecht work engagement scale preliminary manual version 1.1. *Occupational Health Psychology Unit, Utrecht University*.
Scherer, A.P.Z., Mór, F.N. (2020). Uso da técnica Coding DOJO em aulas de programação de computadores. In: *Anais do XXVIII Workshop sobre Educação em Computação*, pp. 6–10. SBC.

Seeling, P. (2016a). Evolving an introductory programming course: impacts of student self-empowerment, guided hands-on times, and self-directed training. In: *2016 IEEE Frontiers in Education Conference (FIE)*, pp. 1–5. IEEE.

Seeling, P. (2016b). Switching to blend-Ed: Effects of replacing the textbook with the browser in an introductory computer programming course. In: *2016 IEEE Frontiers in Education Conference (FIE)*, pp. 1–5. IEEE.

Serrano-Cámara, L.M., Paredes-Velasco, M., Alcover, C.-M., Velazquez-Iturbide, J.Á. (2014). An evaluation of students' motivation in computer-supported collaborative learning of programming concepts. *Computers in human behavior*, 31, 499–508.

Seyam, M., McCrickard, D.S., Niu, S., Esakia, A., Kim, W. (2016). Teaching mobile application development through lectures, interactive tutorials, and Pair Programming. In: *2016 IEEE Frontiers in Education Conference (FIE)*, pp. 1–9. IEEE.

Shokaliuk, S.V., Bohunenko, Y.Y., Lovianova, I.V., Shyshkina, M.P. (2020). Technologies of distance learning for programming basics on the principles of integrated development of key competences. In: *CTE Workshop Proceedings* (Vol. 7), pp. 548–562.

Sibley, J., Ostafichuk, P. (2023). *Getting started with team-based learning*. Taylor & Francis, ???.

Sobral, S.R. (2020). Two different experiments on teaching how to program with active learning methodologies: a critical analysis. In: *2020 15th Iberian Conference on Information Systems and Technologies (CISTI)*, pp. 1–7. IEEE.

Sobral, S.R. (2021a). Project based learning with peer assessment in an introductory programming course.

Sobral, S.R. (2021b). Strategies on teaching introducing to programming in higher education. In: *World Conference on Information Systems and Technologies*, pp. 133–150. Springer.

Sobral, S.R. (2021c). Teaching and learning to program: Umbrella review of introductory programming in higher education. *Mathematics*, 9(15), 1737.

Sobrinho, H., Castro, L., Nogueira, A., Harada, E., Gadelha, B. (2016). Organizando o conhecimento sobre técnicas de aprendizagem colaborativas. *Nuevas Ideas em Informatica Educativa*, 12, 152–156.

Souza, S.M., Bittencourt, R.A. (2019). Motivation and engagement with pbl in an introductory programming course. In: *2019 IEEE Frontiers in Education Conference (FIE)*, pp. 1–9. IEEE.

Souza, S.M., Bittencourt, R.A. (2020). Report of a CS1 Course for Computer Engineering Majors Based on PBL. In: *2020 IEEE Global Engineering Education Conference (EDUCON)*, pp. 837–846. IEEE.

Souza, S.M., Bittencourt, R.A. (2021). Sentiments and Performance in an Introductory Programming Course Based on PBL. In: *2021 IEEE Global Engineering Education Conference (EDUCON)*, pp. 831–840. IEEE.

Srivatanakul, T. (2023). Emerging from the pandemic: instructor reflections and students' perceptions on an introductory programming course in blended learning. *Education and Information Technologies*, 28(5), 5673– 5695.

Steinmacher, I., Silva, M.A.G., Gerosa, M.A., Redmiles, D.F. (2015). A systematic literature review on the barriers faced by newcomers to open source software projects. *Information and Software Technology*, 59, 67–85.

Stephan, J., Oliveira, A., Renhe, M.C. (2020). O uso de jogos para apoiar o ensino e aprendizagem de programação. In: *Anais do XXXI Simpósio Brasileiro de Informática na Educação*, pp. 381–390. SBC.

Suarez-Escalona, R., Estrada-Dominguez, J., Infante-Alcantara, L., Cavazos-Salazar, R., Treviño-Rodriguez, F. (2022). Active Learning Implementation as Digital Education Strategy During the COVID-19. In: *13th International Multi-Conference on Complexity, Informatics and Cybernetics, IMCIC 2022*, pp. 63–68.

Sulaiman, S. (2020). Pairing-based approach to support understanding of object-oriented concepts and programming. *Int. J. Adv. Sci. Eng. Inf. Technol*, *10*(4).

Sung, K., Shirley, P. (2003). A top-down approach to teaching introductory computer graphics. In: *ACM SIGGRAPH 2003 Educators Program*, pp. 1–4.

Tao, Y., Nandigam, J. (2016). Programming case studies as context for active learning activities in the classroom. In: *2016 IEEE Frontiers in Education Conference (FIE)*, pp. 1–4. IEEE.

Tenenberg, J., Fincher, S. (2005). Students designing software: a multi-national, multi-institutional study. *Informatics in Education*, 4(1), 143–162.

Tharayil, S., Borrego, M., Prince, M., Nguyen, K.A., Shekhar, P., Finelli, C.J., Waters, C. (2018). Strategies to mitigate student resistance to active learning. *International Journal of STEM Education*, 5(1), 1–16.

Topalli, D., Cagiltay, N.E. (2018). Improving programming skills in engineering education through problem-based game projects with Scratch. *Computers & Education*, 120, 64–74.

Turner, S.A., Pérez-Quiñones, M.A., Edwards, S.H. (2018). Peer review in CS2: Conceptual learning and high-level thinking. *ACM Transactions on Computing Education (TOCE)*, 18(3), 1–37.

Turpen, C., Dancy, M., Henderson, C. (2016). Perceived affordances and constraints regarding instructors' use of Peer Instruction: Implications for promoting instructional change. *Physical Review Physics Education Research*, 12(1), 010116.

Unterkalmsteiner, M., Gorschek, T., Islam, A.M., Cheng, C.K., Permadi, R.B., Feldt, R. (2011). Evaluation and measurement of software process improvement—a systematic literature review. *IEEE Transactions on Software Engineering*, 38(2), 398–424.

Veerasamy, A.K., D'Souza, D., Apiola, M.-V., Laakso, M.-J., Salakoski, T. (2020). Using early assessment performance as early warning signs to identify at-risk students in programming courses. In: *2020 IEEE Frontiers in Education Conference (FIE)*, pp. 1–9. IEEE.

Venter, M. (2020). Gamification in STEM programming courses: State of the art. In: *2020 IEEE Global Engineering Education Conference (EDUCON)*, pp. 859–866. IEEE.

Vihavainen, A., Airaksinen, J., Watson, C. (2014). A systematic review of approaches for teaching introductory programming and their influence on success. In: *Proceedings of the Tenth Annual Conference on International Computing Education Research*, pp. 19–26.

Wang, G., Zhao, H., Guo, Y., Li, M. (2019). Integration of flipped classroom and problem based learning model and its implementation in university programming course. In: *2019 14th International Conference on Computer Science & Education (ICCSE)*, pp. 606–610. IEEE.

West, R.E., Waddoups, G., Graham, C.R. (2007). Understanding the experiences of instructors as they adopt a course management system. *Educational Technology Research and Development*, 55, 1–26.

Wieringa, R.J. (2014). Design science methodology for information systems and software engineering.

Xie, S., Hu, C., Wu, W., Fan, L., Xiong, Y., Tao, J. (2021). Blended Practical Teaching of Object Oriented Programming Based on PBL and Task Driven. In: *2021 5th International Conference on Education and E-Learning*, pp. 125–128.

Xu, F., Correia, A.-P. (2023). Adopting distributed pair programming as an effective team learning activity: a systematic review. *Journal of Computing in Higher Education*, 1–30.

Yang, F.-C.O., Lai, H.-M., Wang, Y.-W. (2023). Effect of augmented reality-based virtual educational robotics on programming students' enjoyment of learning, computational thinking skills, and academic achievement. *Computers & Education*, 195, 104721.

Yang, S., Park, H., Choi, H. (2021). Impact of Active Learning on Object-Oriented Programming Instruction: Transforming from 3D to Text-based coding. In: *2021 IEEE Integrated STEM Education Conference (ISEC)*, pp. 252–255. IEEE.

Yuan, H., Cao, Y. (2019). Hybrid pair programming-a promising alternative to standard pair programming. In: *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*, pp. 1046–1052.

Zayapragassarazan, Z., Kumar, S. (2012). Active learning methods. *Online Submission*, 19(1), 3–5.

Zhang, L., Niu, J. (2022). Research to Practice in Computer Programming Course using Flipped Classroom. In: *2022 IEEE Frontiers in Education Conference (FIE)*, pp. 1–7. IEEE.

Zhang-Kennedy, L., Chiasson, S. (2021). A systematic review of multimedia tools for cybersecurity awareness and education. *ACM Computing Surveys (CSUR)*, 54(1), 1–39.

M.I. Calderon Ribeiro is currently pursuing a Ph.D. degree in informatics with the Federal University of Amazonas (UFAM). Her research interests include software engineering education, active learning strategies, and related topics. She is an associate professor at the Federal Institute Rondônia (IFRO – Porto Velho North Zone Campus).

W. Silva received a Ph.D. in Informatics from the Institute of Computing of the Federal University of Amazonas (UFAM). He is currently an Adjunct Professor of Software Engineering at the Federal University of Pampa (UNIPAMPA). He is also a member of the LESSE Research Group (Laboratory of Empirical Studies in Software Engineering), the Steering Committee (2022-2023 and 2023-2024) of the Special Committee on Information Systems (CESI) of the Brazilian Computer Society (SBC), and is part of the Active Methodologies Interest Group linked to the Special Committee on Computing Education. His research interests include Software Engineering, Empirical Software Engineering, Software Quality, Computing Education Research, Usability, User Experience, Machine Learning, and Human-Centered Machine Learning.

E.L. Feitosa received a degree in data processing from the Federal University of Amazonas (UFAM) in 1998, a master's degree in computer science from the Federal University of Rio Grande do Sul (UFRGS), in 2001, and the Ph.D. degree in computer science from the Federal University of Pernambuco (UFPE). He is an Associate Professor with the Institute of Computing (IComp), UFAM. He is also a Researcher and a Leader with the Emerging Technologies and System Security (ETSS) Research Group. He holds a position as a Research Fellow with the Networking and Emerging Technologies Research Group.

Appendix A

Table 13 presents the relevant publications for this systematic mapping.

Table 13
Selected publications

ID	Publication title	Authors/year
S01	Flipping Web Programming Class: Student's Perception and Performance	Hendrik (2019)
S02	Flipped Classroom Strategy to Help Underachievers in Java Programming	Kumar *et al.* (2018)
S03	Is More Active Always Better for Teaching Introductory Programming?	Raj *et al.* (2018)
S04	Teaching Introduction to Computing through a project-based collaborative learning approach	Avouris *et al.* (2010)
S05	Separation of syntax and problem-solving in Introductory Computer Programming	Edwards *et al.* (2018)
S06	Evaluating the Benefits of Team-Based Learning in a Systems Programming Class	Joshi *et al.* (2020)
S07	Evolving an introductory programming course: Impacts of student self-empowerment, guided hands-on times, and self-directed training	Seeling (2016a)
S08	Flipping a Programming Course: the Good, the Bad, and the Ugly	Rosiene and Rosiene (2015)
S09	A Case Study of an Integrated Programming Course Based on PBL	Ribeiro and Bittencourt (2019)
S10	A PBL-Based, Integrated Learning Experience of Object-Oriented Programming, Data Structures and Software Design	Ribeiro and Bittencourt (2018)
S11	Serious Games for Motivating into Programming	Hijon-Neira *et al.* (2014)
S12	Applying Flipped Classroom and Problem-Based Learning in a CS1 Course	de Oliveira Fassbinder *et al.* (2015)
S13	Report of a CS1 Course for Computer Engineering Majors Based on PBL	Souza and Bittencourt (2020)
S14	Improving Student Learning in an Introductory Programming Course Using Flipped Classroom and Competency Framework	Elmaleh and Shankararaman (2017)
S15	Integration of Flipped Classroom and Problem-Based Learning Model and its Implementation in University Programming Course	Wang *et al.* (2019)
S16	Animated Flowchart with Example Followed by Think-Pair-Share Activity for Teaching Algorithms of Engineering Courses	Dol (2018)
S17	Active Learning in Small to Large Courses	Astrachan *et al.* (2002)
S18	Programming Case Studies as Context for Active Learning Activities in the Classroom	Tao and Nandigam (2016)
S19	A Games-Based Approach for Teaching the Introductory Programming Course	Rajaravivarma (2005)
S20	A Modified Team-Based Learning Methodology for Effective Delivery of an Introductory Programming Course	Elnagar and Ali (2012)
S21	Mobile game development: Improving student engagement and motivation in introductory computing courses	Kurkovsky (2013)
S22	Improving the CS1 Experience with Pair Programming	Nagappan *et al.* (2003)
S23	Two different experiments on teaching how to program with active learning methodologies: critical analysis	Sobral (2020)
S24	Modeling Different Variables in Learning Basic Concepts of Programming in Flipped Classrooms	Durak (2020)

Continued on next page

Table 13 – continued from previous page

ID	Publication title	Authors/year
S25	Hybrid Pair Programming – A Promising Alternative to Standard Pair Programming	Yuan and Cao (2019)
S26	Redesigning an introductory programming course to facilitate effective student learning: a case study	Corritore and Love (2020)
S27	Pairing-Based Approach to Support Understanding of Object-Oriented Concepts and Programming	Sulaiman (2020)
S28	Using Flipped Classroom and Team-Based Learning in a First-Semester Programming Course: An Experience Report	Loftsson and Matthíasdóttir (2019)
S29	Effect of Think-Pair-Share in a Large CS1 Class: 83 Sustained Engagement	Kothiyal *et al.* (2014)
S30	Interactive Preparatory Work in a Flipped Programming Course	Cao and Grabchak (2019)
S31	Peer Review in CS2: Conceptual Learning and High-Level Thinking	Turner *et al.* (2018)
S32	Making Parallel Programming Accessible to Inexperienced Programmers through Cooperative Learning	Pollock and Jochen (2001)
S33	Collaborative Strategy for Teaching and Learning Object-Oriented Programming Course: A Case Study at Mostafa Stambouli Mascara University, Algeria	Boudia *et al.* (2019)
S34	Think-Pair-Share in a Large CS1 Class: Does Learning Really Happen?	Kothiyal *et al.* (2014)
S35	Teaching CS 1 with POGIL Activities and Roles	Hu and Shepherd (2014)
S36	Students' Perception of a Blended Learning Approach in an African Higher Institution	Safana and Nat (2019)
S37	Implementation and Evaluation of Flipped Algorithmic Class	Amira *et al.* (2019)
S38	Analyzing the effects of adapted flipped classroom approach on computer programming success, attitude toward programming, and programming self-efficacy	Özyurt and Özyurt (2018)
S39	Integrating Project-Based Learning in Mobile Development Course to Enhance Student Learning Experience	Rahman (2018)
S40	Collaborative Learning in Computer Programming Courses That Adopted The Flipped Classroom	Hayashi *et al.* (2015)
S41	An Empirical Study of In-Class Laboratories on Student Learning of Linear Data Structures	Heckman (2015)
S42	Object-oriented programming course revisited	Herala *et al.* (2015)
S43	Improving programming skills in engineering education through problem-based game projects with Scratch	Topalli and Cagiltay (2018)
S44	Using New Methodologies in Teaching Computer Programming	Drini (2018)
S45	Teaching Mobile Application Development through Lectures, Interactive Tutorials, and Pair Programming	Seyam *et al.* (2016)
S46	Exploring Active Learning Approaches to Computer Science Classes	Caceffo *et al.* (2018)
S47	Including Coding Questions in Video Quizzes for a Flipped CS1	Lacher *et al.* (2018)
S48	Active Learning through Game Play in a Data Structures Course	Dicheva and Hodge (2018)
S49	Investigating the Impact of a Meaningful Gamification-Based Intervention on Novice Programmers' Achievement	Agapito *et al.* (2018)
S50	Switching to Blend-Ed: Effects of Replacing the Textbook with the Browser in an Introductory Computer Programming Course	Seeling (2016b)
S51	Design and Large-scale Evaluation of Educational Games for Teaching Sorting Algorithms	Battistella *et al.* (2017)
S52	Applying PBL in Teaching Programming: na Experience Report	dos Santos *et al.* (2018)
S53	Modern board games to improve problem solving in programming students	Araújo *et al.* (2020)

Continued on next page

Table 13 – continued from previous page

ID	Publication title	Authors/year
S54	Game of Code: development and evaluation of a gamified activity for programming disciplines	Melo and Soares Neto (2017)
S55	KLouro: An educational game to motivate beginner students in programming	de Azevêdo Silva and Dantas (2014)
S56	The Snake Challenge – Using gamification to motivate students in an introductory programming course	Raposo and Dantas (2016)
S57	Competitive Programming as a tool to support the teaching of algorithms and data structure for Computer Science students	Brito *et al.* (2019)
S58	The Use of Games to Support the Teaching and Learning of Programming	Stephan *et al.* (2020)
S59	Using Problem-Based Learning to Teach Programming	Finger *et al.* (2021)
S60	Experience in Using Gamified Online Tools in Introduction to Computer Programming	Nagai *et al.* (2016)
S61	Use of the Coding DOJO technique in computer programming classes	Scherer and Mór (2020)
S62	Gamification Elements Applied in Web Programming Teaching-Learning	Gonçalves *et al.* (2019)
S63	Coding Dojo as a Collaborative Learning Practice to Support Introductory Programming Teaching: A Case Study	Alves *et al.* (2019)
S64	A Preliminary Analysis of the Application of Method 300 in Algorithms and Programming Classes	de Castro Junior *et al.* (2021)
S65	Application of Inverted Room and Gamification Elements to Improve Teaching-Learning in Object Oriented Programming	Costa *et al.* (2017)
S66	An Integrated Experience of Object Oriented Programming, Data Structures and Systems Design with PBL	Bittencourt *et al.* (2013)
S67	Logirunner: A Board Game as a Tool to Aid the Teaching and Learning of Algorithms and Logic Programming	Casarotto *et al.* (2018)
S68	A Model to Promote Student Engagement in Programming Learning Using Gamification	da Silva *et al.* (2018)
S69	A Bottom-Up Approach for Computer Programming Education	Gamage (2021)
S70	Blended Practical Teaching of Object Oriented Programming Based on PBL and Task Driven	Xie *et al.* (2021)
S71	POGIL in CS1: Evidence for Student Learning and Belonging	Mayfield *et al.* (2022)
S72	The Impact of Pair Programming on College Students' Interest, Perceptions, and Achievement in Computer Science	Bowman *et al.* (2021)
S73	Impact of Active Learning on Object-Oriented Programming Instruction	Yang *et al.* (2021)
S74	Research to Practice in Computer Programming Course using Flipped Classroom	Zhang and Niu (2022)
S75	Transformation in Course Delivery Augmented with Problem-Based Learning and Tutorial	Desai *et al.* (2021)
S76	Using Flipped Classroom, Peer Discussion, and Just-in-time Teaching to Increase Learning in a Programming Course	Jonsson (2015)
S77	Using Online Forums to Promote Collaborative Learning in Introductory Programming Courses	Michaličková (2021)
S78	Sentiments and Performance in an Introductory Programming Course Based on PBL	Souza and Bittencourt (2021)
S79	Project Based Learning with Peer Assessment in an Introductory Programming Course	Sobral (2021a)
S80	Application of a Teaching Plan for the Discipline of Algorithms with Active Methodologies: A Report of a Pilot Case Study	da Silva Garcia and Oliveira (2022)

Appendix D

XIII Congresso Brasileiro de Informática na Educação (CBIE 2024)
XXXV Simpósio Brasileiro de Informática na Educação (SBIE 2024)

Um Survey sobre o Uso de Metodologias Ativas de Aprendizagem no Ensino de Programação em Universidades Brasileiras

Ivanilse Calderon[1,3], Ana Carolina Oran[1], Eduardo Feitosa[1], Williamson Silva[2]

[1]Instituto de Computação (IComp) – Universidade Federal do Amazonas (UFAM)
Manaus, AM – Brasil

[2]Programa de Pós-Graduação em Engenharia de Software (PPGES) -
Universidade Federal do Pampa (UNIPAMPA) - Alegrete, RS - Brasil

[3]Instituto Federal de Educação, Ciência e Tecnologia de Rondônia (IFRO)
Campus Porto Velho Zona Norte - Porto Velho, RO - Brasil

{[1,3]ivanilse.calderon,[1]ana.oran,efeitosa}@icomp.ufam.edu.br

[2]williamsonsilva@unipampa.edu.br

Abstract. *Teaching programming is challenging because it requires students to develop abstraction, problem-solving, and logical reasoning skills. There is evidence that Active Learning Methodologies (ALMs) can facilitate the efficient development of these skills. This paper describes the results of a survey conducted with 102 teachers from different regions of Brazil, which summarized evidence on the use of ALMs in teaching programming. The results were obtained from 22 states, with the highest participation from the North region (37.2%) and a predominance of instructors working in public institutions (77.5%). The results indicated that 78.4% of instructors already use or are using ALMs, and the three most adopted ALMs are Problem-Based Learning, Gamification, and Project-Based Learning.*

Resumo. *Ensinar programação é desafiador devido à necessidade de desenvolver habilidades como abstração, resolução de problemas e raciocínio lógico nos estudantes. Há evidências de que as Metodologias Ativas de Aprendizagem (MAAs) podem facilitar o desenvolvimento dessas competências de forma eficiente. Este artigo apresenta os resultados de um survey conduzido com 102 docentes de diferentes regiões do Brasil que sumarizariou evidências sobre o uso das MAAs no ensino de programação. Os resultados foram obtidos de 22 unidades federativas, com maior participação proveniente da região Norte (37,2%) e uma predominância de docentes atuando em instituições públicas (77,5%). Os resultados indicaram que 78,4% dos docentes já utilizaram ou estão utilizando MAAs e as mais adotadas são Problem-Based Learning, Gamificação e Project-Based Learning.*

1. Introdução

O ensino na área da Computação enfrenta desafios significativos, exigindo um equilíbrio entre os conhecimentos teóricos e abordagens de aprendizagem práticas e aplicadas

[dos Santos et al. 2020]. Dada a relevância da Computação no cotidiano, o ensino de programação tornou-se um desafio ainda maior [Eickholt 2018]. Ensinar programação é uma tarefa complexa, pois os estudantes ao final das disciplinas necessitam compreender como utilizar diferentes tecnologias de forma eficaz [Liao e Ringler 2023]. Contudo, aprender a programar, especialmente no início dos cursos, é desafiador para os estudantes. Muitos enfrentam dificuldades ao planejar e escrever programas, e alguns consideram os conteúdos de programação difíceis de compreender [Okonkwo e Ade-Ibijola 2023]. Em geral, os estudantes têm dificuldades em aprender a programar devido à falta de compreensão de conceitos fundamentais para escrever programas simples [Corritore e Love 2020].

A abordagem centrada no professor, típica das aulas tradicionais, é muitas vezes ineficaz para o desenvolvimento de competências importantes, pois tende a levar os estudantes a absorver passivamente as informações apresentadas [Caceffo et al. 2018]. Como resultado, muitos estudantes desistem das disciplinas ou mesmo do próprio curso [Sobral 2021b, Garcia et al. 2021]. No entanto, esse cenário tem mudado nas últimas décadas, impulsionado pelo contínuo avanço tecnológico e por novas abordagens pedagógicas. Um exemplo dessa evolução são as Metodologias Ativa de Aprendizagem (MAAs), amplamente discutidas e adotadas no ensino de programação [Sobral 2021a]. Essas metodologias promovem a participação ativa dos estudantes no processo de aprendizagem, contribuindo para o desenvolvimento efetivo de competências práticas. As MAAs visam capacitar os estudantes a lidar com os desafios do mercado de trabalho, desenvolver maior autonomia na resolução de problemas e melhorar a comunicação [Garcia et al. 2021].

Este artigo descreve um *survey* conduzido com docentes de cursos de Computação no Brasil, visando investigar a adoção das MAAs no ensino de programação. O *survey* examinou percepções dos docentes sobre o uso dessas metodologias, assim como as dificuldades e desafios enfrentados em sala de aula. Os resultados oferecem um panorama da adoção das MAAs no Brasil e revelam as percepções dos docentes ao ensinarem disciplinas de programação. Por meio deste estudo, busca-se fornecer *insights* valiosos para a melhoria das práticas pedagógicas na educação em programação, contribuindo para um ensino mais eficaz e alinhado às necessidades do mercado e dos estudantes.

2. Fundamentação Teórica

O ensino de programação tem se tornado importante devido à crescente relevância da Computação no cotidiano. No entanto, os docentes enfrentam diversos desafios, pois os estudantes precisam entender a sintaxe e a semântica das linguagens de programação, e também desenvolver habilidades como a capacidade cognitiva para abstrair problemas, resolver desafios, exercitar o raciocínio e o pensamento lógico [Sharma et al. 2022]. Muitos estudantes enfrentam dificuldades no início dos cursos ao projetar e escrever programas simples, e alguns consideram a programação uma disciplina complexa [Okonkwo e Ade-Ibijola 2023]. A falta de compreensão de conceitos fundamentais é um obstáculo, resultando em baixo desempenho, frustração, falta de engajamento, dentre outros fatores [Corritore e Love 2020].

Esses desafios refletem nas altas taxas de evasão nos cursos de Computação. Algumas instituições de ensino superior relatam taxas de evasão de até 50%, e a média global de aprovação em cursos introdutórios de Ciência da Computação é de cerca de 68% [Penney et al. 2023]. Isso é atribuído, em parte, às técnicas instrucio-

nais e pedagógicas atuais, altas expectativas dos docentes e a falta de suporte aos ingressantes [Beaubouef e Mason 2005, Luxton-Reilly 2016, Denny et al. 2011]. Diante desses desafios, as MAAs vêm ganhando destaque entre os docentes [Parsons 2011, Berssanette e de Francisco 2021, Calderon et al. 2024]. As MAAs têm sido cada vez mais adotadas em salas de aula por serem centradas nos estudantes e por promoverem maior envolvimento na aprendizagem. A adoção de MAAs tem implicações práticas bem-sucedidas, proporcionando aos estudantes desafios semelhantes aos que enfrentarão no mercado de trabalho [Garcia et al. 2021].

As MAAs combinam participação ativa do estudante, aprendizagem experimental e aprendizagem pela ação, tornando o estudante mais responsável na aprendizagem, o que resulta em maior motivação e satisfação [Imbulpitiya et al. 2020]. As vantagens do aprendizado ativo sobre o aprendizado passivo incluem a participação efetiva dos estudantes na construção da sua aprendizagem [Bacich e Moran 2018] e o estímulo à autonomia, que apoia no desenvolvimento das habilidades relacionadas à resolução de problemas [Witt et al. 2018]. Portanto, algumas MAAs têm sido implementadas no ensino de programação em cursos de graduação em Computação, visando capacitar os estudantes para os desafios do mercado de trabalho, desenvolver maior autonomia na resolução de problemas e melhorar a comunicação [Garcia et al. 2021]. Vale ressaltar ainda que a implementação bem-sucedida das MAAs exige conhecimento e planejamento cuidadoso por parte dos docentes. Compreender as diferentes estratégias, bem como seus sucessos e fracassos, é importante para docentes que desejam incorporar novas metodologias e estratégias de aprendizagem ativa em suas aulas de programação.

3. Trabalhos Relacionados

Nesta seção, são discutidos trabalhos que exploram diferentes abordagens e metodologias educacionais, com ênfase na eficácia da aprendizagem ativa e MAAs nos currículos de Ciência da Computação. Esses estudos oferecem uma visão abrangente das tendências e desafios atuais na educação.

Hassan e Puteh (2017) conduziram um *survey* sobre o uso da Aprendizagem Ativa Habilitada por Tecnologia nas práticas de ensino e aprendizagem para melhorar a qualidade dos estudantes de engenharia. Elahi *et al.* (2016) conduziram uma revisão sobre estratégias de aprendizagem ativa recentes, agrupando-as em duas dimensões distintas: personalização, ou seja, se os itens selecionados pelo sistema são diferentes para diferentes usuários, e hibridização, ou seja, se o aprendizado ativo é guiado por um único critério (heurística) ou por múltiplos critérios. O trabalho de Bishop e Verleger (2013) conduziram uma revisão da literatura sobre o uso de sala de aula invertida. Os resultados desta pesquisa mostram que a maioria dos estudos realizados até o momento exploram as percepções dos estudantes e utilizam projetos em grupo. Os relatos sobre as percepções dos estudantes sobre a sala de aula invertida são variadas, mas, em geral, são positivos.

Suo *et al.* (2021) conduziram um *survey* com docentes para compreender sobre a inclusão e adoção do uso de metodologias ativas no conteúdos de Computação Paralela e Distribuída (PDC) em currículos dos cursos de Ciência da Computação. Enquanto que Wiggins *et al.* (2017) conduziram um *survey* para obter uma visão mais holística da experiência dos estudantes em sala de aula a partir do uso das metodologias ativas. A pesquisa conduzida por Villas-Boas *et al.* (2012) visou determinar o estado da arte da

Aprendizagem Ativa no Ensino de Engenharia no Brasil e os esforços de pesquisa nesta área, bem como mapear os pesquisadores envolvidos neste tipo de abordagem de ensino-aprendizagem. Os autores relataram que as estratégias ativas de aprendizagem aplicadas nas escolas de engenharia do Brasil já apresentam resultados, que muitas vezes indicam a necessidade de realinhamento em suas concepções iniciais, bem como na organização do currículo do curso. Lima *et al.* (2020) conduziram um *survey* para diagnosticar o uso de metodologias ativas no processo de ensino-aprendizagem das disciplinas da Engenharia de Software nas instituições de ensino brasileiras. Como resultados, os autores relataram que apesar do aumento na aplicação destas e os benefícios produzidos pelas mesmas, os resultados indicam alguns obstáculos que tornam difícil o seu uso. Diante deste cenário, observa-se que estas pesquisas oferecem uma rica contribuição para a comunidade científica ao explorar e avaliar diversas metodologias educacionais que buscam melhorar a qualidade do ensino e a preparação dos estudantes para os desafios profissionais.

4. Método de pesquisa

O objetivo deste estudo foi compreender a percepção dos docentes sobre a adoção e uso das MAAs e compreender sobre as dificuldades e/ou desafios enfrentados ao utilizar estas MAAs em sala de aula no ensino de programação. Para alcançar este objetivo, foi adotado o método Pesquisa de Opinião (*survey*), empregando um questionário *online* como abordagem para coleta das percepções dos docentes. Segundo Kitchenham e Pfleeger (2008), um *survey* é um método de pesquisa utilizado para sumarizar e compreender as características investigadas a partir de ampla população de indivíduos. O público-alvo deste *survey* são docentes de instituições de ensino superior que têm experiência no ensino de programação empregando algum tipo de metodologia ativa de aprendizagem. Em relação ao *design* da coleta de dados, o questionário aplicado foi classificado como um recorte transversal, no qual os docentes participantes forneceram suas informações em relação às suas experiências e visão dentro de um determinado contexto [Oliveira et al. 2017].

4.1. Planejamento

O processo, desde a definição dos objetivos até a elaboração do questionário e a aplicação para obter dados válidos, foi inspirado nas quatro etapas propostas por Kitchenham e Pfleeger (2008) para assegurar a consistência e confiabilidade do estudo. Na primeira etapa, foram investigados trabalhos de revisão e mapeamento sistemático da literatura já realizados (ver Seção 3). Isso ajudou os pesquisadores a compreender melhor o estado da arte e as tendências atuais das MAAs, os desafios frequentes e as boas práticas percebidas pelos docentes, bem como compreender algumas motivações iniciais. Na segunda etapa (*Design* do Questionário), definiu-se a população-alvo, a questão de pesquisa, o método de coleta e os critérios de análise. Esses elementos foram estabelecidos para direcionar o estudo e garantir que os dados coletados fossem relevantes para o contexto da pesquisa. Na terceira etapa (Aplicação do Questionário), realizou-se a coleta de dados, iniciando com um estudo piloto para validar o questionário. Em seguida, o questionário foi disponibilizado e divulgado entre o público-alvo. Na quarta e última etapa (Documentação e divulgação dos resultados), o *survey* foi documentado e compartilhado utilizando a ferramenta Google Forms. Os resultados foram analisados com o auxílio de outros três pesquisadores especialistas, visando garantir a precisão da análise e a divulgação dos resultados.

4.2. *Design* do questionário

Para a construção do questionário, foram seguidas as diretrizes sugeridas por Coelho *et al.* (2019), assegurando que as fossem elaboradas seguindo uma ordem lógica e encadeada. As perguntas são apresentadas na Tabela 1.

Tabela 1. Perguntas criadas para o *survey*.

ID	Questão
Q01	Perguntas referentes ao perfil (gênero, estado que leciona, titulação, etc.) e experiência dos participantes (lecionando computação, lecionando aulas de programação, linguagens adotadas).
Q02	Você adota algum tipo de MAA para o ensino de programação?
Q03	Há quanto tempo você vem adotando MAA no ensino de programação?
Q04	Quais MAAs você já utilizou em suas aulas de programação?
Q05	Qual é a sua motivação para a adoção das MAAs para o ensino de programação?
Q06	Quais os pontos positivos percebidos em relação a adoção das MAAs para o ensino de programação?
Q07	Você já sentiu dificuldade em adotar as MAAs?
Q08	Quais os principais desafios e pontos negativos enfrentados em relação ao uso de MAAs no ensino de programação?

As perguntas do questionário foram organizadas em diferentes tipos para investigar a adoção de MAAs no ensino de programação. Inicialmente, são coletadas informações sobre o perfil e a experiência dos participantes, como cargo atual e tempo de adoção de MAAs. Em seguida, os participantes são questionados sobre a adoção específica de MAAs para ensinar programação, seguido pela identificação das metodologias já utilizadas e em quais disciplinas foram aplicadas. Também são explorados os tipos de ferramentas e plataformas utilizadas no processo de ensino. Além disso, são investigadas motivações, benefícios percebidos, dificuldades enfrentadas e desafios relacionados à adoção dessas metodologias. As questões finais abordam métodos de avaliação da aprendizagem dos estudantes e as *soft skills* percebidas como desenvolvidas durante o processo de ensino de programação. Essa estrutura visa compreender amplamente a implementação e impacto das MAAs no contexto específico do ensino de programação. As respostas das perguntas do *survey* podem ser: perguntas fechadas de escolha única e perguntas de múltipla escolha. Em Q1, Q2 e Q3, os participantes selecionam uma única opção que melhor descreve seu perfil e experiência. A partir da Q4, os participantes foram apresentados a uma lista de opções em que poderiam escolher as que melhor refletem sua realidade ou perspectiva.

Além das perguntas usadas para coletar as informações dos participantes, uma seção inicial foi apresentada contendo o objetivo do estudo, o Termo de Consentimento Livre e Esclarecido (TCLE) [1] com informações sobre os direitos dos participantes e garantia de anonimato, além de uma questão solicitando a concordância dos participantes para participar do estudo. Vale ressaltar que o presente estudo está dispensado de apresentação de Comitê de Ética, por enquadrar-se na categoria *Pesquisa de Opinião pública com participantes não identificáveis*, conforme o Ofício Circular No. 17/2022/CONEP/SECNS/MS, de julho de 2022 e Ofício Circular No. 12/2023/CONET/SECNS/DGIP/SE/MS.

4.3. Aplicação do questionário

Para coletar as respostas dos docentes em diferentes regiões, utilizou-se o Google *Forms* para disponibilizar o questionário *on-line*. O *link* do questionário foi compartilhado em diversos grupos de interesse no tema e em redes sociais, possibilitando a identificação de

[1] https://figshare.com/s/bd04d980e05a30de434c

potenciais respondentes dispostos a participar do estudo. O questionário foi enviado por meio dos e-mails institucionais, grupos de WhatsApp, abrangendo o quantitativo de 107 instituições de ensino superior. O período de coleta foi de 28 de novembro de 2023 a 17 de maio de 2024.

4.4. Documentação e Divulgação dos resultados

Foi realizada uma análise descritiva dos dados de forma univariada [Nardi 2018]. Os dados foram tabulados e os gráficos foram gerados com o auxílio do *Microsoft Excel*. Para possibilitar a transparência e a reprodutibilidade do processo científico [Mendez et al. 2020] seguido pelos pesquisadores, todos os dados utilizados durante a análise podem ser acessados por meio deste link[2]. Por fim, a divulgação dos resultados à comunidade, está sendo realizada a partir da publicação deste trabalho.

5. Análise dos resultados

5.1. Visão geral dos participantes

O estudo obteve a participação de 102 docentes de 21 Estados e o Distrito Federal, totalizando 22 unidades federativas participantes. Apresentando o seguinte cenário: Norte: 37,2%, Nordeste: 14,9%, Sudeste: 11,8%, Sul: 9,7% e o Centro-Oeste: 6,9%. A maior participação dos docentes estão nas regiões Norte, sendo Amazonas (23,5%), Rondônia (21,6%) e Acre (7,8%), seguido pelos Estados de Minas Gerais (6,9%) e Alagoas (5,9%).

Analisando o perfil dos docentes, notou-se que 62,7% (64) se declararam como Homem e 37,3% (38) se declararam como Mulher. Em relação às instituições de ensino, 77,5% (70) dos docentes atuam em instituições públicas, 19,6% (20) em instituições privadas e 2,9% (3) em instituições comunitárias, o que reflete a predominância do setor público na oferta educacional. Quanto à titulação, 50% (51 docentes) possuem mestrado, seguido por 34,3% (35) com doutorado, 5,9% (6) com graduação e apenas 2,9% (3) possuem especialização.

Quanto à experiência em sala de aula, observou-se que a maioria dos docentes possui 10 anos de experiência (11,8%), seguido por 11 anos (6,9%), indicando uma trajetória significativa no ensino. No que diz respeito à experiência em lecionar programação, 9,8% dos docentes têm 5 anos de experiência e 8,8% entre 2 e 10 anos, refletindo uma diversidade de níveis de experiência. Além disso, dentre as linguagens mencionadas encontram-se Python (66.3%), C++ (40,6%), C (35,6%), Java (35,6%), Java Script (11,9%), Pascal (9,9%) e C (5,9%).

5.2. Experiência sobre o uso das Metodologias Ativas de Aprendizagem

A Figura 1 apresenta o percentual de docentes que utilizam MAAs (Parte 1) e tempo de adoção dessas metodologias no ensino de programação (Parte 2).

Sobre a adoção das MAAs (Q2), os resultados mostram que 78,4% dos docentes informaram que usa ou usou algum tipo de MAA no ensino de programação. Ainda constatou-se que (Q3): 21,2% dos participantes têm utilizado MAAs por três anos; 17,5% aplicam essas metodologias há quatro anos, 16,2% utilizam MAAs há cinco anos, 15% que começaram a utilizá-las há dois anos e 8,8% adotam essas práticas há oito anos. Esses

[2]https://figshare.com/s/0027c75e6b77f6c2c849

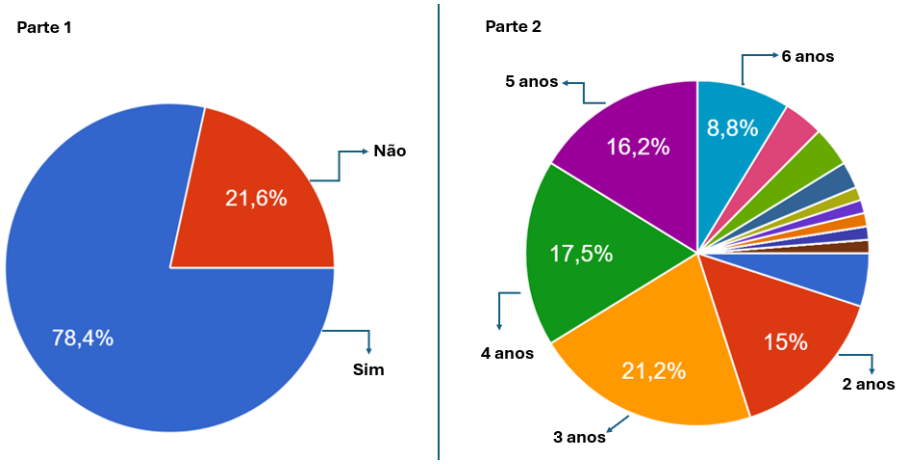


Figura 1. Percentual de docentes que adotam as MAAs e o tempo de adoção.

dados indicam uma crescente adoção das MAAs no ensino de programação, com muitos docentes já implementando essas práticas há vários anos. Os resultados evidenciam que os docentes brasileiros usam as MAAs como facilitadores do aprendizado, guiando e apoiando os estudantes em seu processo de descoberta e construção do conhecimento.

A Figura 2 apresenta o cenário em relação as MAAs empregadas pelos docentes no ensino de programação (Q4). As MAAs **sempre** utilizadas nas aulas de programação incluem: *Problem-Based Learning* (26), Gamificação (19), *Project-Based Learning* (14), *Coding Dojo* (19) e *Peer Review* (5). As MAAs **quase sempre** utilizadas são: *Problem-Based Learning* (26), Gamificação (21), *Project-Based Learning* (20), *Pair Programming* (13), *Team-Based Learning* (11) e *Coding Dojo* (9). As MAAs **algumas vezes** adotadas pelos docentes incluem: *Project-Based Learning* (14), *Team-Based Learning* (12), Gamificação (12), *Problem-Based Learning* (11), *Pair Programming* (8), *Coding Dojo* (8), *Peer Review* (5). Os resultados demonstram que o uso diversificado e frequente de MAAs nas aulas de programação aponta para uma mudança significativa nas estratégias de ensino, buscando melhorar o engajamento e o desempenho dos estudantes por meio de MAAs mais participativas e colaborativas.

5.3. Percepções dos docentes sobre a adoção e o uso das MAAs na prática docente

Ao analisar as percepções dos docentes sobre a adoção e o uso das MAAs na prática docente, observa-se que este é um aspecto importante para o aprimoramento contínuo da qualidade do ensino. Compreender as motivações, pontos positivos e negativos, e os desafios enfrentados pelos educadores ao implementar MAAs permite identificar áreas de sucesso e oportunidades de melhoria.

A análise das principais motivações (Q5) apontadas pelos docentes para adotar MAAs no ensino de programação revela uma gama de fatores que influenciam essa decisão (ver motivações na Tabela 2). As principais motivações destacadas pelos docentes foram: o aumento do engajamento dos estudantes (56,3%), a adequação do conteúdo à realidade e prática de ensino (55%) e a possibilidade de inovação na prática docente e possibilidade dos estudantes criarem, adaptarem e modificarem algoritmos ou códigos (cada uma com 46,3%). Esses resultados refletem a busca por métodos de ensino mais eficazes e envolventes, que proporcionem uma experiência de aprendizado mais significativa

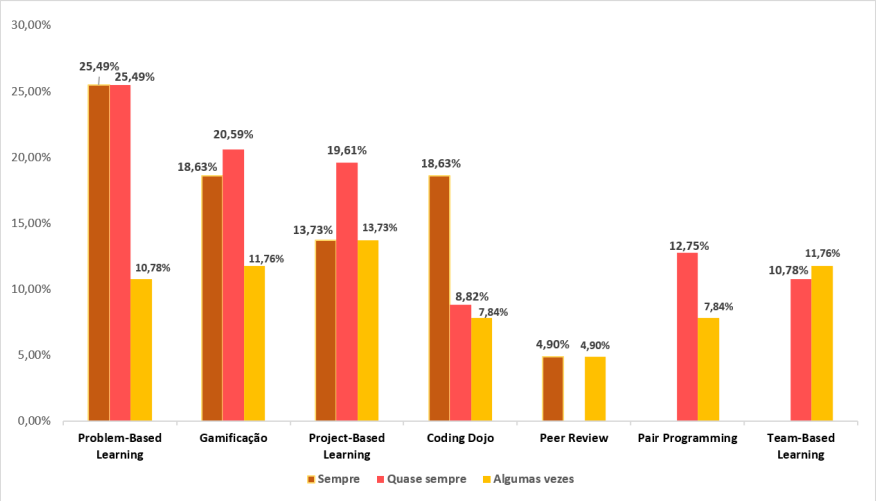


Figura 2. Frequência de uso das MAAs nas aulas de programação.

e relevante para os estudantes. Entendemos que esses aspectos apontam para a importância de abordagens de ensino mais práticas e voltadas para a aplicação do conhecimento em contextos reais, preparando os estudantes não apenas para compreender os conceitos teóricos, mas também para aplicá-los de forma eficaz no mercado de trabalho.

Tabela 2. Motivações mais apontadas pelos docentes

ID	Motivação	Porcentagem (%)
M01	Engajamento dos estudantes para o aprendizado da programação é maior	56,3
M02	Adequação do conteúdo de acordo com a sua realidade e prática de ensino	55,0
M03	Possibilidade de inovação na prática docente	46,3
M04	Possibilidade dos estudantes criarem, adaptarem e modificarem algoritmos ou códigos	46,3
M05	A substituição das aulas expositivas	43,8
M06	Possibilidade de desenvolver habilidades para a prática profissional	41,3
M06	Prática de ensino ativa que dinamiza as aulas	40,0
M07	Possibilidade de realizar avaliações curtas e frequentes	32,5
M08	Adaptação às habilidades e necessidades dos estudantes	27,5
M09	Possibilidade de construção colaborativa do conhecimento	27,5
M10	Metodologias tradicionais de ensino não permitem ao professor meios para aprimorar o ensino dos conteúdos nas disciplinas de programação	26,3
M11	Articulação entre os conteúdos com aplicação no dia a dia do estudante	26,3

Além disso, as motivações relacionadas à dinamização das aulas (40%), à realização de avaliações curtas e frequentes (32,5%) e à adaptação às habilidades e necessidades dos estudantes (27,5%) destacam a importância de uma abordagem personalizada e flexível no ensino de programação. Essa abordagem permite que os docentes atendam às diferentes necessidades e estilos de aprendizagem dos estudantes, promovendo um ambiente de aprendizado mais inclusivo e eficaz. Observa-se que as motivações dos docentes refletem um desejo de promover uma educação mais envolvente, prática e relevante para os estudantes, preparando-os de forma mais eficaz para os desafios do mercado de trabalho e incentivando seu desenvolvimento acadêmico e profissional.

A Tabela 3 apresenta as Percepções Positivas (PP) reportadas pelos docentes em relação à adoção das MAAs no ensino de programação (Q06). A análise dos pontos positivos apresentam uma série de benefícios percebidos para os estudantes e o ambiente de sala de aula. A motivação dos estudantes para aprender os conteúdos (86,3%) é destacada

como o principal benefício, sugerindo que as MAAs podem estimular um interesse mais profundo e significativo nos temas abordados. Além disso, o engajamento dos estudantes em sala de aula (78,8%) foi o segundo benefício destacado pelos docentes. Outros benefícios bem avaliados é a melhoria da capacidade dos estudantes em relação à leitura de código (70%) e o entendimento do funcionamento das instruções de programação (65%), em que os docentes percebem que as MAAs podem contribuir para o desenvolvimento de habilidades práticas e técnicas importantes em programação. A colaboração entre os estudantes durante o aprendizado do conteúdo (46,3%) e a resolução de desafios propostos de forma individual ou em grupo (33,8%) destacam a importância das MAAs na promoção do trabalho em equipe e na troca de conhecimentos entre os estudantes.

Os docentes responderam também que as MAAs melhoram o desempenho individual (51,2%) e no desempenho da turma (45%), sugerindo que as MAAs podem contribuir para o alcance de melhores resultados acadêmicos. A capacidade dos estudantes em desenvolver habilidades, gerar alternativas para solução de problemas, realizar avaliações das soluções encontradas e realizar divisão dos problemas em módulos menores são aspectos que indicam o fortalecimento das competências cognitivas e analíticas dos estudantes. Os resultados indicam que a adoção das MAAs é amplamente percebida como benéfica pelos docentes, especialmente em termos de aumento da motivação e engajamento dos estudantes. Estes aspectos são importantes para a aprendizagem ativa e participativa, elementos centrais para a eficácia do ensino de programação.

Tabela 3. Percepções positivas relatadas pelos docentes para adotar MAAs.

ID	Categoria	Porcentagem (%)
PP01	Motivação para aprender conteúdos	86,3
PP02	Engajamento dos estudantes em sala de aula	78,8
PP03	Melhoria da capacidade de leitura de código	70,0
PP04	Melhoria no entendimento das instruções de programação	65,0
PP05	Melhoria no desempenho individual	51,2
PP06	Colaboração entre estudantes durante o aprendizado	46,3
PP07	Melhoria no desempenho da turma	45,0
PP08	Desafios resolvidos individual ou em grupo	33,8
PP09	Melhoria na interação entre os estudantes	33,8
PP10	Melhoria no desenvolvimento de habilidades	32,5
PP11	Capacidade dos estudantes em gerar alternativas para solução de problemas	31,3
PP12	Melhoria na participação dos estudantes em sala de aula	30,0
PP13	Melhoria na interação entre professor e estudantes	28,7
PP14	Capacidade dos estudantes em realizar avaliação das soluções	27,5
PP15	Capacidade dos estudantes em dividir problemas em módulos menores	27,5
PP16	Compartilhamento do conhecimento entre os estudantes	25,0
PP17	Disposição dos estudantes para resolver problemas	21,3
PP18	Aplicação da teoria nas atividades práticas	18,8
PP19	Capacidade dos estudantes em realizar comparação entre alternativas	15,0

Apesar dos benefícios das MAAs apontadas acima, muitos docentes enfrentam dificuldades ou barreiras na sua adoção em sala de aula. Compreender os aspectos que mais impactam negativamente nas práticas em sala de aula é importante. Para tanto, os participantes foram questionados sobre as dificuldades encontradas ao adotar MAAs (Q07). Os resultados indicam que 53,8% dos docentes afirmam que **às vezes** enfrentam dificuldades, 27,5% relatam enfrentar dificuldades **frequentemente**, 10% raramente encontram dificuldades, 5% **nunca** tiveram dificuldades, e 3,7% **sempre** enfrentam obstáculos na adoção das MAAs. Alinhado à questão acima, os participantes também foram questionados sobre os principais desafios e pontos negativos enfrentados na aplicação das MAAs.

Ao mapear esse cenário por meio da questão Q08, evidenciou-se que a falta de formação docente específica para a aplicação das MAAs (64,6%), a dificuldade em implementar todas as fases das MAAs (58,2%), a carência de suporte tecnológico para facilitar a compreensão e adoção das metodologias (58,2%), a escassez de informações sobre a implementação e adoção das MAAs (55,7%) e a falta de conhecimento ou domínio sobre como implementá-las (55,7%) são os principais obstáculos. Outros desafios incluem a dificuldade em conciliar o uso de tecnologias durante a implementação das MAAs (54,4%), a falta de comprometimento dos estudantes com os estudos prévios necessários (53,2%), a restrição de tempo por parte dos docentes para o planejamento das aulas que adotam MAAs (51,9%), e a ausência de formação pedagógica específica para o ensino com a utilização dessas metodologias (48,1%). Todos esses desafios refletem a complexidade do ambiente educacional atualmente, evidenciam as lacunas na preparação dos docentes. Isso também ressalta a necessidade urgente de investimentos em capacitação profissional e suporte institucional para promover uma implementação eficaz das MAAs no ensino de programação.

6. Discussão dos Resultados

A adoção das MAAs no ensino de programação é importante para promover um aprendizado significativo e eficaz entre estudantes da Computação. Estudos indicam um aumento considerável no engajamento dos estudantes quando essas metodologias são implementadas nas aulas [Acharya e Gayana 2021, Bacich e Moran 2018]. Esse aumento de engajamento se manifesta em maior participação, motivação e interesse dos estudantes. A adoção das MAAs pelos docentes é, portanto, fundamental para assegurar um ensino de qualidade e preparar os estudantes para os desafios do século XXI. Os resultados do questionário evelam que as MAAs mais frequentemente adotadas no ensino de programação incluem *Problem-Based Learning*, Gamificação, *Project-Based Learning*, *Coding Dojo* e *Peer Review*. Comparando esses resultados com a literatura, evidenciamos que há uma convergência em relação às MAAs mais adotadas no ensino de programação. Tanto os resultados do *survey* quanto a revisão sistemática conduzida por Bersanete e Francisco (2021) destacam *Project-Based Learning* como uma das principais metodologias adotadas pelos docentes.

Além disso, os resultados deste trabalho apresentam que as linguagens de programação mais utilizadas pelos docentes são Python (66.3%), C++ (40,6%), C (35,6%), Java (35,6%), Java Script (11,9%), Pascal (9,9%) e C (5,9%). Calderon *et al.* (2021) evidenciaram que linguagem de programação C é a mais utilizada pelos docentes para ensinar programação, seguida por Java e Python. Em seu trabalho mais recente, Calderon *et al.* (2024) relataram que Java está entre as linguagens mais utilizadas, seguida por C++, C e Python. Esses resultados estão alinhados com os relatados na literatura, ressaltando a adaptação curricular às necessidades acadêmicas e profissionais e sustentando a flexibilidade necessária no ensino de programação para preparar estudantes para múltiplos contextos da indústria de tecnologia. Os resultados deste trabalho indicam ainda que a adoção das MAAs é amplamente percebida como benéfica pelos docentes, especialmente em termos de aumento da motivação e engajamento dos estudantes. Liao e Ringler (2023) relatam as percepções positivas dos docentes em relação às MAAs, sugerindo que estão sempre abertos a inovar em suas práticas pedagógicas e a adotar abordagens mais centradas no estudante para melhorar a experiência de aprendizagem dos estudantes em cursos

de Computação. Esses autores destacam que os docentes adotam as MAAs principalmente devido ao aumento da motivação e engajamento dos estudantes e à importância de envolver estes em atividades práticas e projetos relevantes, proporcionando uma conexão mais direta entre o aprendizado teórico e sua aplicação prática. A colaboração entre os estudantes é fortalecida, pois muitas MAAs incentivam o trabalho em equipe e a troca de ideias. Os estudantes sentem-se mais preparados para enfrentar os desafios do mercado de trabalho após passarem por experiências de aprendizado baseadas em MAAs, desenvolvendo não apenas habilidades técnicas, mas também habilidades interpessoais e de resolução de problemas.

Os resultados também revelam um cenário complexo e multifacetado de desafios enfrentados pelos docentes ao adotar MAAs no ensino de programação. A análise dos dados revelou dificuldades enfrentadas pelos docentes, incluindo a falta de formação específica para a aplicação das MAAs, a dificuldade em implementar todas as fases necessárias, a carência de suporte tecnológico, a escassez de informações sobre a implementação e a falta de conhecimento ou domínio sobre como aplicá-las, conforme relatado também no trabalho de Lima *et al.* (2020). Esses desafios refletem a complexidade do ambiente educacional e a necessidade de apoio e desenvolvimento profissional contínuo para os docentes no contexto do ensino de programação. A escassez de tempo para planejamento das aulas sugerem a necessidade de estratégias de gestão de sala de aula e de tempo mais eficazes. Conforme Sobral (2020) e Lima *et al.* (2020), a implementação de MAAs pode exigir tempo e esforço adicionais por parte dos docentes para planejar e executar as atividades de forma eficaz. Kovařík *et al.* (2022) afirmam que a preparação de atividades de aprendizagem ativas pode demandar mais tempo e esforço comparado a preparação de aulas tradicionais.

O suporte institucional e à adaptação curricular não devem ser subestimados. As instituições precisam fornecer os recursos necessários e criar um ambiente que encoraje a experimentação e a inovação pedagógica. A falta de formação específica é um obstáculo crítico para a implementação eficaz dessas metodologias [Calderon et al. 2024, Berssanette e de Francisco 2021]. Segundo Kovarik *et al.* (2022), a adoção bem-sucedida de MAAs requer um certo nível de treinamento e desenvolvimento profissional para garantir que os docentes possam projetar e facilitar atividades de aprendizagem eficazes. Sem a formação adequada, os docentes podem se sentir despreparados para adotar e integrar as MAAs de forma sistemática em seu contexto de ensino. Além disso, a carência de suporte tecnológico e de informações sobre a implementação das MAAs destaca a necessidade de infraestrutura adequada e recursos educacionais. Calderon *et al.* (2024) relatam que a implementação bem-sucedida das MAAs em sala de aula requer tanto o conhecimento pedagógico quanto a disponibilidade de ferramentas tecnológicas. Há, portanto, a necessidade de artefato educacionais que apoiem os docentes na adoção de MAAs no ensino de programação, mininizando as barreiras percebidos pelos docentes [Berssanette e de Francisco 2021, Calderon et al. 2024].

A falta de comprometimento e resistência por parte dos estudantes são desafios. Alguns estudantes podem resistir ao aprendizado ativo devido às mudanças na abordagem de ensino tradicional, exigindo maior participação e envolvimento [Liao e Ringler 2023]. No entanto, ao longo do tempo, muitos estudantes percebem os benefícios dessas estratégias e se tornam mais receptivos a elas, contribuindo para um ambiente de aprendizagem

mais dinâmico [Kovarik et al. 2022, Eickholt 2018]. Kovarik *et al.* (2022) afirmam que os estudantes geralmente se beneficiam das MAAs, pois essas abordagens promovem uma aprendizagem mais envolvente, colaborativa e significativa.

A adoção da MAAs no ensino das disciplinas introdutórias desempenham um papel importante na formação dos estudantes, estabelecendo as bases conceituais e práticas necessárias para sua jornada acadêmica e profissional. A implementação dessas MAAs desde o início do curso pode proporcionar uma introdução mais envolvente e significativa aos conceitos fundamentais da computação, preparando-os para desafios mais avançados. Por outro lado, se as MAAs estiverem sendo adotadas em disciplinas mais avançadas, isso pode indicar uma ênfase na aplicação prática dos conhecimentos adquiridos e no desenvolvimento de habilidades específicas para situações do mundo real. Em ambos os casos, compreender a presença das MAAs nas disciplinas do curso se faz necessário para avaliar o impacto dessas abordagens no ensino de computação e para orientar futuras iniciativas de melhoria curricular e pedagógica.

7. Considerações Finais e Trabalho Futuros

Este estudo apresentou o resultado de um *survey* realizado com 102 docentes de diferentes regiões do Brasil, cujo objetivo foi compreender a percepção dos docentes sobre a adoção e uso das MAAs e compreender sobre as dificuldades e/ou desafios enfrentados ao utilizar estas MAAs em sala de aula no ensino de programação. De acordo com os resultados obtidos, as MAAs incentivam uma compreensão prática dos conceitos, promovem o desenvolvimento de habilidades de programação e fortalecem competências consideradas importantes, como trabalho em equipe, comunicação e pensamento crítico. Além disso, os docentes evidenciaram que essas metodologias preparam os estudantes para os desafios do mercado de trabalho, onde a capacidade de aplicar conhecimentos em projetos práticos e resolver problemas complexos é fundamental. Assim, as MAAs não apenas melhoram a experiência de aprendizagem, mas também capacitam os estudantes para uma carreira bem-sucedida.

Por outro lado, os desafios identificados são igualmente importantes e sugerem a necessidade de uma abordagem multifacetada para a sua superação. A falta de formação específica e de suporte tecnológico, bem como a dificuldade em implementar todas as fases das MAAs, indicam que os docentes precisam de mais recursos e apoio para integrar essas metodologias de forma eficaz em suas práticas pedagógicas. A falta de comprometimento dos estudantes e a restrição de tempo para o planejamento também são questões que precisam ser abordadas nas políticas institucionais e estratégias de ensino que valorizem e facilitem a adoção das MAAs. Para superar essas barreiras, faz-se necessário investir em formação contínua para os docentes, disponibilizar recursos tecnológicos adequados e fornecer suporte institucional, assegurando a implementação eficaz das MAAs e maximizando seus benefícios no ensino de programação. Como trabalhos futuros espera-se investigar estratégias específicas para mitigar os desafios identificados na adoção das MAAs no ensino de programação e a percepção dos estudantes sobre o uso de MAAs em sala de aula. Estudos adicionais podem explorar a eficácia de diferentes abordagens de ensino que facilitem a implementação das MAAs, bem como avaliar o impacto de iniciativas institucionais na motivação e no comprometimento dos estudantes.

Agradecimentos

A presente pesquisa foi realizada com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Código de Financiamento 001 e parcialmente financiado pela Fundação de Amparo à Pesquisa do Estado do Amazonas – FAPEAM – por meio do projeto POSGRAD. Williamson Silva agradece pelo apoio financeiro da FAPERGS (Projeto ARD/ARC) - processo n. 22/2551-0000606. Ivanilse Calderon agradece ao Grupo de Pesquisa em Tecnologias e Educação em Computação (GPComp) e ao Instituto Federal de Educação, Ciência e Tecnologia de Rondônia (IFRO) pelo apoio financeiro - processo n. 23243.004494/2024-20, Programa Institucional de Incentivo à Qualificação (PIQ), Edital n.34/2024/REIT-DGP/IFRO.

Referências

Acharya, S. e Gayana, M. (2021). Enhanced learning and improved productivity of students' using project based learning approaches for programming courses. *Journal of Engineering Education Transformations*, 34:524–530.

Bacich, L. e Moran, J. (2018). *Metodologias ativas para uma educação inovadora: uma abordagem teórico-prática*. Penso Editora.

Beaubouef, T. e Mason, J. (2005). Why the high attrition rate for computer science students: some thoughts and observations. *ACM SIGCSE Bulletin*, 37(2):103–106.

Berssanette, J. H. e de Francisco, A. C. (2021). Active learning in the context of the teaching/learning of computer programming: A systematic review. *Journal of Information Technology Education. Research*, 20:201.

Bishop, J. e Verleger, M. A. (2013). The flipped classroom: A survey of the research. Em *2013 ASEE annual conference & exposition*, páginas 23–1200.

Caceffo, R., Gama, G., e Azevedo, R. (2018). Exploring active learning approaches to computer science classes. Em *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*, páginas 922–927.

Calderon, I., Silva, W., e Feitosa, E. (2021). Um mapeamento sistemático da literatura sobre o uso de metodologias ativas durante o ensino de programação no brasil. *Anais do XXXII Simpósio Brasileiro de Informática na Educação*, páginas 1152–1161.

Calderon, I., Silva, W., e Feitosa, E. (2024). Active learning methodologies for teaching programming in undergraduate courses: A systematic mapping study. *Informatics in Education*, 23(2):279–322.

Coelho, J. A., Souza, G. H., e Albuquerque, J. (2020). Desenvolvimento de questionários e aplicação na pesquisa em informática na educação. *Metodologia de Pesquisa em Informática na Educa\cão: Abordagem Quantitativa de Pesquisa. Porto Alegre: SBC. Série Metodologia de Pesquisa em Informática na Educa\cão*, 2.

Corritore, C. L. e Love, B. (2020). Redesigning an introductory programming course to facilitate effective student learning: A case study. *Journal of Information Technology Education: Innovations in Practice*, 19:091–135.

Denny, P., Luxton-Reilly, A., Tempero, E., e Hendrickx, J. (2011). Understanding the syntax barrier for novices. Em *Proceedings of the 16th annual joint conference on Innovation and technology in computer science education*, páginas 208–212.

dos Santos, S. C., Reis, P. B., Reis, J. F., e Tavares, F. (2020). Two decades of pbl in teaching computing: a systematic mapping study. *IEEE transactions on education*, 64(3):233–244.

Eickholt, J. (2018). Barriers to active learning for computer science faculty. *arXiv preprint arXiv:1808.02426*.

Elahi, M., Ricci, F., e Rubens, N. (2016). A survey of active learning in collaborative filtering recommender systems. *Computer Science Review*, 20:29–50.

Garcia, F. W. D. S., Carvalho, E. D. C., e Oliveira, S. R. B. (2021). Use of active methodologies for the development of a teaching plan for the algorithms subject. Em *2021 IEEE Frontiers in Education Conference (FIE)*, páginas 1–9. IEEE.

Hassan, N. F. e Puteh, S. (2017). A survey of technology enabled active learning in teaching and learning practices to enhance the quality of engineering students. *Advanced Science Letters*, 23(2):1104–1108.

Imbulpitiya, A., Kodagoda, N., Gamage, A., e Suriyawansa, K. (2020). Using active learning integrated with pedagogical aspects to enhance student's learning experience in programming and related concepts. Em *International Conference on Interactive Collaborative Learning*, páginas 218–228. Springer.

Kitchenham, B. A. e Pfleeger, S. L. (2008). Personal opinion surveys. Em *Guide to advanced empirical software engineering*, páginas 63–92. Springer.

Kovarik, M. L., Robinson, J. K., e Wenzel, T. J. (2022). Why use active learning? Em *Active Learning in the Analytical Chemistry Curriculum*, páginas 1–12. ACS Publications.

Liao, Y.-C. e Ringler, M. (2023). Backward design: Integrating active learning into undergraduate computer science courses. *Cogent Education*, 10(1):2204055.

Lima, J. V. V., Silva, C. A. D., de Alencar, F. M. R., e Santos, W. B. (2020). Metodologias ativas como forma de reduzir os desafios do ensino em engenharia de software: diagnóstico de um survey. Em *Anais do XXXI Simpósio Brasileiro de Informática na Educação*, páginas 172–181. SBC.

Luxton-Reilly, A. (2016). Learning to program is easy. Em *Proceedings of the 2016 ACM Conference on Innovation and Technology in Computer Science Education*, páginas 284–289.

Mendez, D., Graziotin, D., Wagner, S., e Seibold, H. (2020). *Open Science in Software Engineering*, páginas 477–501. Springer International Publishing, Cham.

Nardi, P. M. (2018). *Doing Survey Research: A Guide to Quantitative Methods*. Routledge.

Okonkwo, C. W. e Ade-Ibijola, A. (2023). Synthesis of nested loop exercises for practice in introductory programming. *Egyptian Informatics Journal*, 24(2):191–203.

Oliveira, M., Oliveira, S. R. B., e Meira, S. (2017). Condução de uma fábrica de software e o processo de aprendizagem em cursos de graduação de ti: Uma aplicação de um survey sobre a percepção da importância. Em *SBIE*, volume 28.

Parsons, P. (2011). Preparing computer science graduates for the 21st century. *Teaching Innovation Projects*, 1(1).

Penney, J., Pimentel, J. F., Steinmacher, I., e Gerosa, M. A. (2023). Anticipating user needs: Insights from design fiction on conversational agents for computational thinking. Em *International Workshop on Chatbot Research and Design*, páginas 204–219. Springer.

Sharma, V., Bhagat, K. K., Huang, H.-H., e Chen, N.-S. (2022). The design and evaluation of an ar-based serious game to teach programming. *Computers & Graphics*, 103:1–18.

Sobral, S. R. (2020). Two different experiments on teaching how to program with active learning methodologies: A critical analysis. Em *2020 15th Iberian Conference on Information Systems and Technologies (CISTI)*, páginas 1–7. IEEE.

Sobral, S. R. (2021a). Project based learning with peer assessment in an introductory programming course.

Sobral, S. R. (2021b). Strategies on teaching introducing to programming in higher education. Em *World Conference on Information Systems and Technologies*, páginas 133–150. Springer.

Suo, X., Glebova, O., Liu, D., Lazar, A., e Bein, D. (2021). A survey of teaching pdc content in undergraduate curriculum. Em *2021 IEEE 11th Annual Computing and Communication Workshop and Conference (CCWC)*, páginas 1306–1312. IEEE.

Villas-Boas, V., Neto, O. M., Campos, L. C., e Aguiar, B. (2012). A survey of active learning in brazilian engineering schools. Em *Proceedings: Active Learning Engineering Education Workshop*.

Wiggins, B. L., Eddy, S. L., Wener-Fligner, L., Freisem, K., Grunspan, D. Z., Theobald, E. J., Timbrook, J., e Crowe, A. J. (2017). Aspect: A survey to assess student perspective of engagement in an active-learning classroom. *CBE—Life Sciences Education*, 16(2):ar32.

Witt, D. T., Kemczinski, A., e dos Santos, L. M. (2018). Resolução de problemas: Abordagens aplicadas no ensino de computação. *Anais do Computer on the Beach*, páginas 731–740.

Appendix E

Repositório Colaborativo para apoiar a adoção de Metodologias Ativas no Ensino de Programação

Ivanilse Calderon Ribeiro
Instituto de Computação (IComp)
Universidade Federal do Amazonas
Manaus, AM - Brasil
ivanilse.calderon@icomp.ufam.edu.br

Williamson Silva
Departamento de Ciência da Computação
Universidade Estadual do Paraná
Apucarana, PR - Brasil
williamson.silva@gmail.com

Eduardo Luzeiro Feitosa
Instituto de Computação (IComp)
Universidade Federal do Amazonas
Manaus, AM -Brasil
efeitosa@icomp.ufam.edu.br

O ensino de programação é um processo complexo [1], pois requer que os estudantes desenvolvam ao longo da aprendizagem diferentes habilidades, tais como capacidade de abstração, resolução de problemas, raciocínio e pensamento lógico [2-4]. Tradicionalmente, o ensino de programação se dá por meio de aulas expositivas combinadas com exercícios que descrevem problemas ao qual os estudantes devem solucionar [2,5-6]. Essa forma de ensino vem recebendo diversas críticas, uma vez que a transmissão do conhecimento é realizada de forma passiva [7-9]. Visando minimizar este problema, docentes tentam adaptar ou empregar novas estratégias de ensino para proporcionar um ambiente de aprendizagem desafiador e engajador para os estudantes [5,10].

Neste sentido, o uso de Metodologias Ativas (MAs) vêm ganhando destaque entre os docentes [11-12]. De acordo com Koening [13], as MAs baseiam-se na teoria Construtivista, em que a aprendizagem é responsabilidade do estudante. As MAs criam situações de aprendizagem para que os estudantes construam conhecimentos sobre os conteúdos aprendidos, desenvolvendo a capacidade crítica e a reflexão sobre as práticas que realizam, bem como explorando atitudes, valores pessoais e aprendam-fazendo (*learning by doing*) [12,14, 15].

Apesar das evidências positivas em relação às MAs, a adoção por parte dos docentes ainda é relativamente baixa [16-18]. Isso vem ocorrendo devido às diversas barreiras que os professores enfrentam na adoção das MAs, tais como: (a) falta de tempo para o planejamento das aulas adotando MAs [16-17]; (b) dificuldade de cumprir todo o conteúdo da disciplina [16,19]; (c) rejeição por parte dos estudantes em relação à utilização de novas metodologias de ensino; (d) falta de informação sobre como implementar as MAs nas aulas [3,19].

Dado o contexto apresentado, esta pesquisa tem como objetivo apoiar a adoção das MAs para o ensino de programação, minimizando as barreiras e/ou desafios enfrentados pelos docentes. Esta pesquisa está sendo guiada pela metodologia de *Design Science Research* (DSR) [20-21] para delimitar o problema de pesquisa, o desenvolvimento, a avaliação e evolução do artefato. A proposta inicial é desenvolver um repositório colaborativo aberto em que os docentes possam identificar, selecionar, adotar, discutir, comentar, avaliar e possivelmente colaborar com (novas ou não) MAs utilizadas durante o ensino de programação.

O repositório auxiliará o docente na identificação e escolha de MA(s) de acordo com o seu contexto de ensino e que atenda às suas necessidades pedagógicas. O repositório também disponibilizará um conjunto de *guidelines* que contará com o passo a passo para guiar os docentes durante a adoção das MAs. Desta forma, os docentes não precisarão buscar, em vários artigos científicos ou livros, formas de como conduzir uma determinada MA em sala de aula. Com a elaboração do repositório, estas informações ficarão disponíveis em apenas um único lugar. Além disso, em razão da proposição de novas MAs, o repositório será colaborativo e aberto à comunidade acadêmica. Assim, docentes poderão contribuir com MAs adotadas, avaliando positivamente ou não o uso de uma determinada MA. Esta avaliação permitirá que os demais docentes possam compartilhar com a comunidade docente suas experiências de uso de uma MA. Isso ajudará os demais docentes durante o processo de adoção ou não de uma determinada MA. É importante mencionar que o repositório está na fase de ideação, ou seja, ainda está sendo realizada a identificação e o mapeamento das MAs, materiais de apoio, informações e artefatos que poderão ajudar os professores durante a adoção de uma MA. A coleta e curadoria destas informações apoiarão na concepção e no desenvolvimento do artefato proposto (repositório colaborativo aberto).

Por fim, para avaliar a viabilidade de uso e evoluir o repositório, pretende-se conduzir estudos experimentais quantitativos (questionários – Modelo de Aceitação de Tecnologia, *surveys*) e qualitativos (estudos de caso, entrevistas e sessões de grupo focal) com docentes que ministram disciplinas de programação. Espera-se que, a partir do uso do repositório, algumas barreiras enfrentadas pelos docentes durante a adoção de MAs sejam minimizadas, uma vez que a literatura confirma que docentes e pesquisadores em Educação estão obtendo resultados significativamente melhores ao experimentar novas intervenções e abordagens pedagógicas durante o processo de ensino-aprendizagem [22].

AGRADECIMENTOS

Os autores agradecem ao apoio financeiro fornecido de forma indireta das seguintes instituições de ensino: Universidade Federal do Amazonas (UFAM), Instituto Federal de Educação, Ciência e

EduComp'21, Abril 26–30, 2021, Jataí, Goiás, Brasil (On-line)

Tecnologia de Rondônia (IFRO) / Campus Porto Velho Zona Norte e da Universidade Estadual do Paraná (UNESPAR) / Campus Apucarana.

REFERÊNCIAS

[1] Andrew Luxton-Reilly, Ibrahim Albluwi, Brett A. Becker, Michail Giannakos, Amruth N. Kumar, Linda Ott, James Paterson, Michael James Scott, Judy Sheard, and Claudia Szabo. 2018. Introductory programming: a systematic literature review. In Proceedings Companion of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education, pp. 55-106.

[2] Adalbert Gerald Soosai Raj, Jignesh Patel, and Richard Halverson. 2018. Is More Active Always Better for Teaching Introductory Programming? In2018 International Conference on Learning and Teaching in Computing and Engineering (LaTICE), IEEE, 103–109.

[3] Maria Ivanilse Calderon Ribeiro, and Odette Mestrinho Passos. 2020. A Study on the Active Methodologies Applied to Teaching and Learning Process in the Computing Area. In *IEEE Access*, (2020), 219083–219097.

[4] Chandra Turpen, Melissa Dancy, and Charles Henderson. 2016. Perceived affordances and constraints regarding instructors' use of Peer Instruction: Implications for promoting instructional change. In *Physical Review Physics Education Research* 12, 1 (2016), 010116.

[5] Essi Lahtinen, Kirsti Ala-Mutka, and Hannu-Matti Järvinen. 2005. A study of the difficulties of novice programmers. In *ACM SIGCSE Bulletin* 37, 3 (2005), 14–18.

[6] Diego Teixeira Witt, and Avanilde Kemczinski. 2020. Metodologias de Aprendizagem Ativa Aplicadas à Computação: Uma Revisão da Literatura. In *Informática na educação: teoria & prática* 23, 1 (2020).

[7] Lilian Bacich and José Moran. Metodologias Ativas para uma Educação Inovadora: Uma Abordagem Teórico-Prática. Penso Editora.

[8] Chicon P. M., C. R. Quaresma, and S. B. Garcês. 2019. Aplicação do Método de ensino *Peer Instruction* para o Ensino de Lógica de Programação com acadêmicos do Curso de Ciência da Computação. In *Anais do 5º Seminário Nacional de Inclusão Digital (SENID)*. Cruz Alta: UNICRUZ, 02-10.

[9] Aline Diesel, Alda Leila Santos Baldez, and Silvana Neumann Martins. 2017. Os princípios das metodologias ativas de ensino: uma abordagem teórica. In Revista *Thema* 14, 1 (2017), 268–288.

[10] Shreenath Acharya and MN Gayana. 2021. Enhanced Learning and Improved Productivity of Students' using Project Based Learning Approaches for Programming Courses. In *Journal of Engineering Education Transformations* 34, (2021), 524–530.

[11] Ronney Moreira Castro and Sean Siqueira. 2019. Técnicas Alternativas de Ensino (Aprendizagem Ativa) para Disciplinas da Computação: Um Mapeamento Sistemático no Contexto Brasil. In Anais do Workshop de Informática na Escola (Vol. 25, No. 1, pp. 1409-1413.

[12] Diego Teixeira Witt, Avanilde Kemczinski, and Luciane Mulazani dos Santos. 2018. Resolução de problemas: Abordagens aplicadas no ensino de computação. In *Anais do Computer on the Beach* (2018), 731–740.

[13] Kathleen M Koenig. 2020. Personal response systems: Making an informed choice. In *Active Learning in College Science Journal*. Springer, 123–139.

[14] Jose Moran. 2021. Avanços e desafios na educação híbrida. Educação transformadora. Retrieved January 20, 2021 from http://www2.eca.usp.br/moran/.

[15] Paul Parsons. 2011. Preparing computer science graduates for the 21st Century. Teaching Innovation Projects 1, 1 (2011).

[16] Jesse Eickholt. 2018. Barriers to active learning for computer science faculty. arXiv preprint arXiv:1808.02426 (2018).

[17] Joel Michael. 2007. Faculty perceptions about barriers to active learning. In *College Teaching* 55, 2 (2007), 42–47.

[18] Elisa L Park and Bo Keum Choi. 2014. Transformation of classroom spaces: Traditional versus active learning classroom in colleges. In Higher *Education* 68, 5 (2014), 749–771.

[19] Williamson Silva, Bruno Gadelha, Igor Steinmacher, and Tayana Conte. 2020. Towards an open repository for teaching software modeling applying active learning strategies. In *2020 IEEE/ACM 42nd International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET*), IEEE 162–172.

[20] Alan Hevner and Samir Chatterjee. 2010. Design science research in information systems. In Design research in information systems. Springer, 9–22.

[21] Roel J Wieringa. 2014. Design science methodology for information systems and software engineering. Springer.

[22] Arto Vihavainen, Jonne Airaksinen, and Christopher Watson. 2014. A systematic review of approaches for teaching introductory programming and their influence on success. In Proceedings of the tenth annual conference on International computing education research, 19–26.

Appendix F

CollabProg: Um Repositório Colaborativo Aberto para Apoiar na Adoção de Metodologias Ativas no Ensino de Programação

Ivanilse Calderon
ivanilse.calderon@icomp.ufam.edu.br
Instituto de Computação (IComp) - Universidade Federal do Amazonas
Manaus, Amazonas, BR

Williamson Silva
williamsonsilva@unipampa.edu.br
Departamento de Engenharia de Software - Universidade Federal do Pampa (UNIPAMPA)
Alegrete, Rio Grande do Sul, BR

Eduardo Feitosa
efeitosa@icomp.ufam.edu.br
Instituto de Computação (IComp) - Universidade Federal do Amazonas
Manaus, Amazonas, BR

RESUMO

O ensino de programação é um desafio, pois requer que o docente direcione o estudante ao desenvolvimento de diferentes habilidades, tais como abstração do mundo real, resolução de problemas, raciocínio lógico. No entanto, a abordagem tradicional de ensino utilizada não é eficaz para isso. Nesse sentido, as Metodologias Ativas (MAs) vêm sendo adotadas pelos docentes, pois possibilitam o desenvolvimento de habilidades, reflexão sobre as práticas realizadas, explorar atitudes, valores pessoais e o aprender-fazendo. O objetivo desta pesquisa é apoiar os docentes na adoção de MAs no ensino de programação. A metodologia utilizada nesta pesquisa é baseada nas diretrizes do *Design Science Research* que guiará a condução dos estudos, a criação e avaliação do artefato proposto. A principal contribuição para a base de conhecimento é o próprio repositório colaborativo aberto para apoiar o docente na adoção de MAs no ensino de programação, a metodologia da pesquisa usada neste trabalho e o *design* dos estudos experimentais conduzidos.

CCS CONCEPTS

• **Social and professional topics** → Computing education.

PALAVRAS-CHAVE

Ensino de programação, Metodologias Ativas, Computação

1 CARACTERIZAÇÃO DO PROBLEMA

O ensino de programação ainda é um grande desafio para os docentes, uma vez que requer que os estudantes compreendam de forma correta conceitos abstratos [9, 12]. Além disso, os docentes se deparam com a necessidade de motivar e despertar nos estudantes diferentes habilidades ao longo do ensino de programação, tais como a capacidade de abstração, a resolução de problemas e, o raciocínio e pensamento lógico [15, 21].

Nesse sentido, o uso de Metodologias Ativas (MAs) vêm ganhando destaque entre os docentes [7, 27]. Diferente da abordagem tradicional de ensino, as MAs possibilitam que os estudantes assumam um papel ativo na aprendizagem, tendo suas experiências, saberes e opiniões valorizadas como ponto de partida para construção do conhecimento [7], os estudantes constroem conhecimentos sobre os conteúdos aprendidos, desenvolvem a capacidade crítica e passam a refletir sobre suas práticas, sobre os valores pessoais e o aprender-fazendo (*learning by doing*) [19, 27].

Apesar das evidências positivas em relação às MAs no ensino de programação, a adoção por parte dos docentes ainda é relativamente baixa [9, 20], devido às diversas barreiras que os docentes enfrentam na adoção das MAs, tais como: (a) falta de tempo para o planejamento das aulas adotando MAs [9, 18]; (b) dificuldade de cumprir todo o conteúdo da disciplina [9, 23]; (c) rejeição por parte dos estudantes em relação à adoção de novas metodologias de ensino; (d) falta de informação sobre como implementar as MAs nas aulas [23]; (e) os grupos de alunos são grandes e heterogêneos [13]; e (f) falta de suporte tecnológico, um artefato para apoiar o docente na adoção de MAs no ensino de programação.

Diante desse cenário, observa-se pesquisas interessadas no uso das MAs para apoiar os docentes em suas práticas na Computação [22, 27]. No entanto, poucas pesquisas apresentam soluções ou suporte tecnológico para apoiar o docente a minimizar as barreiras e/ou os desafios enfrentados na adoção das MAs para o ensino de programação, logo, observa-se lacunas para serem pesquisadas.

O problema tratado nesta pesquisa está relacionado com a melhoria do processo de ensino de programação em Computação. Neste contexto, este trabalho está sendo guiado pela seguinte questão de pesquisa: **Como minimizar as barreiras e/ou desafios enfrentados pelos docentes durante a adoção de metodologias ativas no ensino de programação em Computação?**

Este artigo está organizado da seguinte forma: A Seção 2 discorre sobre a fundamentação teórica. A Seção 3 discute os trabalhos relacionados. A Seção 4 apresenta a metodologia da pesquisa e métodos para avaliar os resultados. Por fim, a Seção 5 detalha o estado atual da pesquisa.

2 FUNDAMENTAÇÃO TEÓRICA

As disciplinas de lógica de programação e programação de computadores, bem como o estudo de linguagens de programação, apresentam um nível de dificuldade alto para muitos estudantes, exigindo grande esforço [10]. Isso ocorre porque tais disciplinas exigem do estudante conhecimentos prévios em lógica, matemática, leitura e interpretação de texto, abstração de ideias e outras habilidades [4].

A literatura ainda destaca que o processo de ensino em disciplinas de programação de computadores é um processo complexo devido às necessidades dos estudantes desenvolverem diferentes habilidades [15, 21]. Além disso, nas salas de aula os grupos de

EduComp'22, Abril 24-29, 2022, Feira de Santana, Bahia, Brasil (On-line)

estudantes são grandes e heterogêneos, portanto, é difícil planejar as aulas de forma que seja benéfica para todos [13].

Diante desse cenário, as MAs vêm sendo utilizadas pelos docentes para o ensino de programaçao [7, 27], pois possibilitam uma mudança de paradigma de aprendizagem, em que o estudante sai do papel de agente passivo, que apenas escuta e recebe o conteúdo que é transmitido pelo docente, e passa para o papel de agente ativo, tornando-se o responsável por sua própria aprendizagem [8, 19]. As MAs são estratégias de ensino centradas na participação efetiva dos estudantes e auxiliam na construção do processo de aprendizagem de forma flexível, interligada e híbrida [7]. As MAs também englobam a concepção do processo de ensino e aprendizagem que considera a participação efetiva dos estudantes na construção da sua aprendizagem, valorizando as diferentes formas pelas quais eles podem ser envolvidos nesse processo para que aprendam, em seu próprio ritmo, tempo e estilo [3]. Além disso, podem estimular a motivação e a autônomia do estudante e ajudar no desenvolvimento das habilidades relacionadas à resolução de problemas [27].

3 TRABALHOS RELACIONADOS

A literatura evidencia os esforços despendidos pela academia para mitigar os desafios do processo de ensino na Computação. Como diferencial e critério de inovação, esta pesquisa destaca-se pela criação de um repositório colaborativo aberto para apoiar na adoção de MAs no ensino de programação. Os repositórios digitais surgiram conjuntamente ao avanço da tecnologia, quando muda-se a forma de armazenar informações [17]. Apresenta-se a seguir tecnologias utilizadas para apoiar os docentes na utilização de MAs na prática docente.

O trabalho de Castro e Siqueira [6] apresenta um portal chamado ALCASYSTEM, que recomenda técnicas de MAs para disciplinas da área de Computação. Para que o docente adote determinada MA é necessário selecionar opções no portal para obter recomendações de artigos para leitura. Logo, a demanda de tempo e opções para seleção, poderá desmotivar o docente na adoção das técnicas.

Silva et al. [24] apresenta um repositório aberto para auxiliar no ensino de modelagem de software empregando estratégias de aprendizagem ativa, chamado OpenSMALS, que disponibiliza um conjunto de métodos e atividades, baseados em estratégias ativas de aprendizagem, para docentes dos cursos de Engenharia de Software (ES). Logo, contexto da pesquisa é mais especifico.

O trabalho de Lima et al.[14] mostra um guia preliminar para seleção assertiva de MAs no ensino de ES, que fornece aos docentes uma ferramenta para auxiliar na escolha assertiva das MAs. Observa-se que o guia é específico para ES, traz apenas dez tipos de MAs, é disponibilizado em arquivo digital e não permite interação entre a comunidade.

Ahshan [1] mostra um *framework* para implementar estratégias para o envolvimento ativo dos estudantes no ensino remoto durante a pandemia do COVID-19, que traz atividades/estratégias para garantir o envolvimento ativo dos estudantes com foco no contexto remoto, contudo, não apresenta experiências e/ou avaliações de outros docentes em relação às atividades/estratégias apresentadas.

A literatura apresenta pesquisas que abordam a utilização de MAs no ensino em diferentes contextos. No entanto, não há trabalho que apresente ao docente um repositório colaborativo aberto para apoiar na adoção de MAs no ensino de programação. Diante disso, como diferencial e inovação, o CollabProg será um apoio tecnológico, disponível na internet que apresentará um conjunto de *guidelines* para apoiar o docente na adoção de MAs no ensino de programação. Destaca-se ainda como inovador por possibilitar a comunidade academica compartilhar experiências, por ser um ambiente colaborativo e aberto. Além disso, apresentará vários tipos de MAs e técnicas associadas a diversos recursos pedagógicos para o ensino de programação. Os docentes não precisarão buscar em vários artigos científicos ou livros estratégias para adoção das MAs, pois tais informações ficarão disponíveis em apenas um único lugar.

4 METODOLOGIA DA PESQUISA E MÉTODOS PARA AVALIAR OS RESULTADOS

Para delimitar o problema de pesquisa, o desenvolvimento, a avaliação e evolução do artefato proposto, esta pesquisa é guiada pela metodologia *Design Science Research* (DSR) [11, 26]. O DSR enfatiza a conexão entre conhecimento e prática [25] e vem sendo utilizada em pesquisas educacionais [2]. Esta Seção apresenta a visão geral da metodologia da pesquisa (Figura 1), os estudos experimentais para avaliar o artefato propostos e os resultados alcançados.

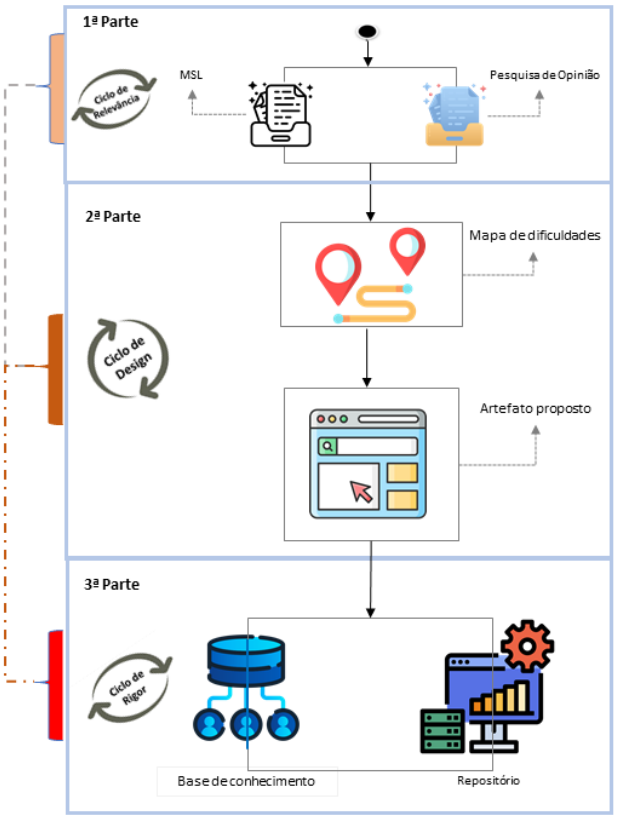


Figura 1: Visão geral da metodologia da pesquisa.

Na 1ª parte da pesquisa (**Ciclo de Relevância**) é definido o problema a ser investigado, compreendido o contexto da investigação, estabelecida a motivação para a solução do problema e os critérios de aceitação para a avaliação final dos resultados da pesquisa. Para isso, serão conduzidos três estudos exploratórios: **a)** Um Mapeamento Sistemático da Literatura (MSL) para buscar na literatura evidências sobre as MAs e as áreas de conhecimento em que elas estão sendo empregadas para o ensino na Computação; **b)** Um segundo MSL será conduzido para sumarizar os tipos de MAs e quais as evidências experimentais que existem na adoção de tais MAs para o ensino de programação; e **c)** Uma Pesquisa de Opinião com docentes que lecionam na área da Computação, para identificar as MAs adotadas, as barreiras e os desafios enfrentados em relação a adoção das MAs.

Na 2ª parte da pesquisa (**Ciclo de *Design***) será desenvolvido, evoluído e avaliado o **CollabProg**, um Repositório Colaborativo Aberto para Apoiar na Adoção de Metodologias Ativas no Ensino de Programação. O CollabProg apoiará o docente na identificação, escolha e adoção das MAs de acordo com o contexto de ensino e que atenda às necessidades pedagógicas no ensino de programação. Para isso, este repositório disponibilizará um conjunto de *guidelines* que contará com o passo a passo para guiar os docentes durante a adoção das MAs nas aulas de programação. O CollabProg será colaborativo e aberto à comunidade acadêmica, em razão da proposição de novas MAs frente aos diferentes cenários de ensino, para que os docentes possam identificar, selecionar, adotar, discutir, comentar, avaliar e possivelmente colaborar com as MAs utilizadas durante o ensino de programação e, possibilitará o compartilhamento de experiências e avaliações em relação a adoção das MAs, contribuindo assim, com a comunidade acadêmica no processo de adoção das MAs durante o ensino de programação.

Como um apoio tecnológico, o CollabProg reunirá em um único repositório estratégias de como conduzir a adoção de diferentes tipos de MAs para o ensino de programação, disponibilizará ao docente um passo a passo objetivo e prático, juntamente com exemplos, sugestões de atividades, opções de suporte ferramental disponível e adotado pela comunidade, experiências sobre a adoção das MAs em cenários diferentes, os resultados alcançados, os pontos positivos e negativos sobre a MA adotada. Assim, os docentes não precisarão buscar em vários artigos científicos ou livros como utilizar determinada MA. Em relação à curadoria dos conteúdos que serão compartilhados no CollabProg, de modo geral, a perspectiva é que seja feito um processo de triagem que visará garantir a confiabilidade dos conteúdos apresentados, para que se tenha a adoção efetiva das MAs no ensino de programação. Além disso, para a curadoria, os pesquisadores envolvidos irão propor critérios que avaliarão os conteúdos que serão disponibilizados no repositório para evitar frustações dos usuários que utilizarão do repositório.

Para avaliar a viabilidade de uso e evoluir o CollabProg, pretende-se conduzir estudos experimentais quantitativos, por meio de questionários, utilizando o *Technology Accepentace Model* (TAM) e entrevistas semiestruturadas. Além disso, planeja-se realizar estudos qualitativos por meio de estudos de caso, sessões de grupo focal e entrevistas com os docentes da área para obter uma compreensão ampla do contexto em que os docentes atuam [16]. O objetivo é realizar os estudos com docentes de Instituições de Ensino Superior Públicas ou privadas no Brasil e em disciplinas que tratam o conteúdo de programação de computadores, seja em turma iniciante ou não.

Também, será formulado e evoluído um modelo de dificuldades em relação adoção das MAs no ensino de programação para avaliar e validar o CollabProg. Este modelo de dificuldades será consolidado a partir dos resultados dos estudos experimentais realizados. O objetivo é projetar um modelo a partir da perspectiva e experiências dos docentes da área e que lecionam as disciplinas de programação. Para avaliar o modelo será utilizada uma pesquisa de opinião realizada com os docentes, a evolução se dará a partir das perspectivas e avaliação dos próprios docentes e, por fim, será validado por meio do uso do modelo pelos docentes. Assim, espera-se que o CollabProg seja avaliado a partir de diferentes experiências, necessidades e contextos do ensino de programação.

A 3ª parte da pesquisa (**Ciclo de Rigor**) refere-se principalmente à geração e o uso de conhecimento [11]. Nesta parte, os principais fundamentos estão relacionados ao conhecimento sobre a adoção de MAs para o ensino de programação na Computação, às estratégias para adoção das MAs, o MSL, os estudos experimentais, às análises qualitativa e quantitativa, ao grupo focal, à entrevista, dentre outros. Em relação à geração de conhecimento, a principal contribuição é o próprio CollabProg, o conjunto de estratégias para a adoção das MAs e, um modelo de dificuldades na adoção da MAs, na perspectiva dos docentes que ensinam programação.

Em se tratando das limitações da pesquisa, pode-se considerar que o repositório será avaliado na perspectiva de apoiar o docente em suas práticas no ensino de programação. Contudo, embora sejam analisados diferentes bases de dados e conduzidos diferentes estudos, provavelmente não serão alcançadas todas a percepções dos docentes sobre a adoção das MAs, o que não inviabiliza sua avaliação a partir da perspectiva do estudante. Em relação aos cenários e respectivos conteúdos que serão compartilhados no repositório, não é possível afirmar que representarão todos os cenários do ensino de programação. Por fim, em relação aos dados coletados, pode-se apontar a subjetividade na classificação dos mesmos, que busca-se mitigar com um processo de análise e interpretação rigoroso.

5 ESTADO ATUAL DA PESQUISA

Como o objetivo de apoiar os docentes na adoção de MAs no ensino de programação, a pesquisa iniciou em março de 2020. Para conduzir estudos experimentais junto à comunidade acadêmica, submeteu-se o projeto da pesquisa para a avaliação da Comissão de Ética em Pesquisa (CEP) da Universidade Federal do Amazonas (UFAM), o projeto foi aprovado pela CEP sob o parecer n.4.694.031. A Tabela 1 apresenta o cronograma e as atividades da pesquisa.

Até o presente período, finalizou-se a Atividade 1 da pesquisa. Os resultados alcançados no estudo realizado trouxeram evidências sobre a adoção de 6 tipos de MAs em 35 cursos e conteúdos diferentes da área da Computação, além disso, apresentou as percepções dos estudantes em relação a adoção das MAs no ensino. Os resultados completos estão em Calderon et al. [22].

A Atividade 2 está em desenvolvimento. Parte dos resultados desta atividade encontra-se em Calderon et al. [5], em que os autores apresentam um MSL sobre o uso de MAs no contexto brasileiro e percebeu-se que as MAs que mais se destacam como suporte ao

ensino na Computação são as MAs Jogos Educacionais e a Gamificação. É importante mencionar que o estudo conduzido na Atividade 2, está em fase de conclusão, pois os resultados locais alcançados estão sendo expandidos com os resultados obtidos em conferências e periódicos internacionais.

Tabela 1: Cronograma das atividades da pesquisa proposta.

Id	Atividades	2020	2021	2022	2023
1	Conduzir 1ª MSL - MAs na Computação	x	x		
2	Conduzir 2º MSL - MAs no ensino de programação		x	x	
3	Pesquisa de opinião com os docentes			x	
4	Construir, avaliar e validar o mapa de dificuldades			x	
5	Construir, avaliar e validar o repositório proposto			x	x

O estudo da Atividade 3 está em fase de planejamento, passando atualmente por análise e avaliação das questões e do formulário da pesquisa de opinião. Com a condução deste estudo, espera-se alcançar evidências sobre a adoção das MAs, considerando as práticas docentes, experiênicas e diferentes contextos em que estão sendo adotas as MAs no ensino de programação na área da Computação.

Pretende-se iniciar a Atividade 4 após a consolidação dos resultados dos estudos experimentais realizados nas Atividades 3. Por fim, é importante mencionar que a Atividade 5, está em fase ideação, ou seja, ainda está sendo realizada a identificação e o mapeamento das MAs, materiais de apoio, informações e artefatos que poderão apoiar o docente na adoção de uma MA. Assim, a coleta e curadoria destas informações apoiarão na concepção e no desenvolvimento do CollabProg.

6 AGRADECIMENTOS

Ao Instituto de Computação da Universidade Federal do Amazonas (UFAM), a Fundação de Amparo à Pesquisa do Estado do Amazonas (FAPEAM) - POSGRAD 2017 (Resolução 002 / 2016), a Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código Financeiro 001, ao Instituto Federal de Educação, Ciência e Tecnologia de Rondônia (IFRO) / Campus Porto Velho Zona Norte e a Universidade Federal do Pampa (UNIPAMPA - Alegrete) pelo apoio.

REFERÊNCIAS

[1] Razzaqul Ahshan. 2021. A framework of implementing strategies for active student engagement in remote/online teaching and learning during the COVID-19 pandemic. *Education Sciences* 11, 9, 483.

[2] Alan César Belo Angeluci, Gabriela Leal Redigolo, Paulo Sergio Felix da Silva, and Patrícia Jaqueline Arakaki. 2020. DESIGN SCIENCE RESEARCH COMO MÉTODO PARA PESQUISAS EM TIC NA EDUCAÇÃO. In *Anais do CIET: EnPED: 2020-(Congresso Internacional de Educação e Tecnologias| Encontro de Pesquisadores em Educação a Distância)*.

[3] Lilian Bacich and José Moran. 2018. *Metodologias ativas para uma educação inovadora: uma abordagem teórico-prática.* Penso Editora.

[4] Nara Martini Bigolin, Sidnei Renato Silveira, Cristiano Bertolini, Iara Carnevale de Almeida, Marlise Geller, Fábio José Parreira, Guilherme Bernardino da Cunha, and Ricardo Tombesi Macedo. 2020. Metodologias Ativas de Aprendizagem: um relato de experiência nas disciplinas de programação e estrutura de dados. *Research, Society and Development* 9, 1, e74911648–e74911648.

[5] Ivanilse Calderon, Williamson Silva, and Eduardo Feitosa. 2021. Um Mapeamento Sistemático da Literatura sobre o uso de Metodologias Ativas durante o Ensino de Programação no Brasil. In *Anais do XXXII Simpósio Brasileiro de Informática na Educação*. SBC, 1152–1161.

[6] Ronney Moreira de Castro and Sean Siqueira. 2019. ALCASYSTEM-Um Portal com Técnicas de Aprendizagem Ativa para Disciplinas da Área da Computação. In *Anais dos Workshops do Congresso Brasileiro de Informática na Educação*, Vol. 8. 1243.

[7] Ronney Moreira de Castro and Sean Siqueira. 2019. Técnicas alternativas de ensino (aprendizagem ativa) para disciplinas da computação: Um mapeamento sistemático no contexto brasil. In *Anais do Workshop de Informática na Escola*, Vol. 25. 1409–1413.

[8] Aline Diesel, Alda Leila Santos Baldez, and Silvana Neumann Martins. 2017. Os princípios das metodologias ativas de ensino: uma abordagem teórica. *Revista Thema* 14, 1, 268–288.

[9] Jesse Eickholt. 2018. Barriers to active learning for computer science faculty. *arXiv preprint arXiv:1808.02426*.

[10] Laís Freire, Jarbele Coutinho, Verônica Lima, and Náthalee Lima. 2019. Uma Proposta de Encontros de Tutoria Baseada em Metodologias Ativas para Disciplinas de Programação Introdutória. In *Anais dos Workshops do Congresso Brasileiro de Informática na Educação*, Vol. 8. 298.

[11] Alan Hevner and Samir Chatterjee. 2010. Design science research in information systems. In *Design research in information systems*. Springer, 9–22.

[12] Asanthika Imbulpitiya, Nuwan Kodagoda, Anjalie Gamage, and Kushnara Suriyawansa. 2019. Using active learning integrated with pedagogical aspects to enhance student's learning experience in programming and related concepts. In *International Conference on Interactive Collaborative Learning*. Springer, 218–228.

[13] Essi Lahtinen, Kirsti Ala-Mutka, and Hannu-Matti Järvinen. 2005. A study of the difficulties of novice programmers. *Acm sigcse bulletin* 37, 3, 14–18.

[14] José Lima, Fernanda Alencar, and Wylliams Santos. 2021. A Preliminary Guide for Assertive Selection of Active Methodologies in Software Engineering Education. In *Brazilian Symposium on Software Engineering*. 170–179.

[15] Andrew Luxton-Reilly, Ibrahim Albluwi, Brett A Becker, Michail Giannakos, Amruth N Kumar, Linda Ott, James Paterson, Michael James Scott, Judy Sheard, and Claudia Szabo. 2018. Introductory programming: a systematic literature review. In *Proceedings Companion of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education*. 55–106.

[16] Irene Manotas, Christian Bird, Rui Zhang, David Shepherd, Ciera Jaspan, Caitlin Sadowski, Lori Pollock, and James Clause. 2016. An empirical study of practitioners' perspectives on green software engineering. In *2016 IEEE/ACM 38th International Conference on Software Engineering (ICSE)*. IEEE, 237–248.

[17] Rodrigo Medeiros, Marcos Doarte, José Viterbo, Cristiano Maciel, and Clodis Boscarioli. 2021. Uma Análise Comparativa entre Repositórios de Recursos Educacionais Abertos para a Educação Básica. In *Anais do XXXII Simpósio Brasileiro de Informática na Educação*. SBC, 213–224.

[18] Joel Michael. 2007. Faculty perceptions about barriers to active learning. *College teaching* 55, 2, 42–47.

[19] José Morán. 2015. Mudando a educação com metodologias ativas. *Coleção mídias contemporâneas. Convergências midiáticas, educação e cidadania: aproximações jovens* 2, 1, 15–33.

[20] Elisa L Park and Bo Keum Choi. 2014. Transformation of classroom spaces: Traditional versus active learning classroom in colleges. *Higher Education* 68, 5, 749–771.

[21] Adalbert Gerald Soosai Raj, Jignesh Patel, and Richard Halverson. 2018. Is More Active Always Better for Teaching Introductory Programming?. In *2018 International Conference on Learning and Teaching in Computing and Engineering (LaTICE)*. IEEE, 103–109.

[22] Maria Ivanilse Calderon Ribeiro and Odette Mestrinho Passos. 2020. A Study on the Active Methodologies Applied to Teaching and Learning Process in the Computing Area. *IEEE Access* 8, 219083–219097.

[23] Williamson Silva, Bruno Gadelha, Igor Steinmacher, and Tayana Conte. 2020. Towards an open repository for teaching software modeling applying active learning strategies. In *2020 IEEE/ACM 42nd International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET)*. IEEE, 162–172.

[24] Williamson Alison Freitas Silva et al. 2020. OPENSMALS: um repositório aberto para auxiliar no ensino de modelagem de software empregando estratégias de aprendizagem ativa.

[25] Roel Wieringa. 2009. Design science as nested problem solving. In *Proceedings of the 4th international conference on design science research in information systems and technology*. 1–12.

[26] Roel J Wieringa. 2014. *Design science methodology for information systems and software engineering*. Springer.

[27] Diego Teixeira Witt, Avanilde Kemczinski, and Luciane Mulazani dos Santos. 2018. Resolução de problemas: Abordagens aplicadas no ensino de computação. *Anais do Computer on the Beach*, 731–740.

Appendix G

CollabProg: Um Repositório Colaborativo Aberto para Apoiar na Adoção de Metodologias Ativas no Ensino de Programação

Ivanilse Calderon[1,3], Klissia Reis[1], Saulo Silva[1], Hélio Endrio[1], Roseno Silva[1], Miquéias Viana[1], Symon Cristhian[1], Williamson Silva[2], Eduardo Feitosa[1]

[1]Instituto de Computação (IComp) – Universidade Federal do Amazonas (UFAM) Manaus, AM – Brasil

[2]Laboratory of Empirical Studies in Software Engineering (LESSE) - Departamento de Engenharia de Software - Universidade Federal do Pampa (UNIPAMPA) - Alegrete, RS - Brasil

[3]Instituto Federal de Educação, Ciência e Tecnologia de Rondônia (IFRO) Campus Porto Velho Zona Norte - Porto Velho, RO - Brasil

{[1,3]ivanilse.calderon,[1]efeitosa}@icomp.ufam.edu.br

[2]williamsonsilva@unipampa.edu.br

Abstract. *CollabProg is an open collaborative repository, available on the internet, which brings together, in a single environment, different types of Active Methodologies (AMs) for teaching programming and aims to support teachers in the adoption of AMs in teaching programming. The methodology used in this research is based on the Design Science Research guidelines that guided the conducting the studies, creating and evaluating the proposed artifact. The AMs presented by CollabProg went through a screening process, which established quality criteria in relation to the detailing and adoption of the methodology in teaching programming. In this way, the reliability of the contents presented was guaranteed, in order to have the effective adoption of AMs in programming teaching.*
Keywords *Teaching programming, Active Methodologies, Computing*

Resumo. *O CollabProg é um repositório colaborativo aberto, disponível na internet, que reúne, em único ambiente diferentes tipos de Metodologias Ativas (MAs) para o ensino de programação e tem por objetivo apoiar os docentes na adoção de MAs no ensino de programação. A metodologia utilizada nesta pesquisa é baseada nas diretrizes do Design Science Research que guiou a condução dos estudos, a criação e avaliação do artefato proposto. As MAs apresentadas pelo CollabProg passaram por um processo de triagem, que estabeleceu critérios de qualidade em relação ao detalhamento e adoção da metodologia no ensino de programação. Deste modo, garantiu-se a confiabilidade dos conteúdos apresentados, a fim de se ter a adoção efetiva das MAs no ensino de programação.*
Palavras-chave: *Ensino de programação, Metodologias Ativas, Computação*

CollabProg

Autores: Ivanilse Calderon[1], Klissia Reis [2], Saulo Silva [3], Hélio Endrio [4], Roseno Silva[5], Miquéias Viana[6], Symon Cristhian[7], Williamson Silva[8], Eduardo Feitosa[9].

Um Repositório Colaborativo Aberto para Apoiar na Adoção de Metodologias Ativas no Ensino de Programação

Contexto

O **ensino** de **programação** é um processo complexo [1].

Requer **estímulos** para o **desenvolvimento** de diferentes **habilidades** [2,3].

Comumente, o **ensino** se dá por meio de **metodologias tradicionais** [4,5].

Motivação

Os docentes buscam **adaptar** ou **empregar novas estratégias** pedagógicas para um **ambiente de ativo** [6,7].

As **Metodologias Ativas (MAs)** para o ensino vêm ganhando destaque entre os docentes [9, 10,11].

No entanto, existem **uma série de barreiras** que impedem a adoção das MAs em sala de aula [12,13, 14].

Algumas das barreiras enfrentadas pelos docentes

Barreiras

- ✓ **Falta de tempo** para o planejamento das aulas adotando MAs [15,16].
- ✓ Dificuldades de **trabalhar todo o conteúdo** e cumprir o currículo [16, 17].
- ✓ **Rejeição dos estudantes** em relação a utilização de novas metodologias de ensino [16,18].
- ✓ Dificuldade de **compreensão sobre a aplicabilidade** das MAs [15,16].
- ✓ **Falta de informação** sobre como **implementar** as MAs em sala de aula [12,14].

Problema

Diante deste cenário, o problema tratado nesta pesquisa está relacionado com a melhoria do processo de ensino de programação em Computação.

Objetivo do CollabProg

Minimizar as barreiras e/ou os desafios enfrentados pelos docentes durante a adoção de MAs no ensino de programação de computadores.

ivanilse.calderon@icomp.ufam.edu.br [1]; kars@icomp.ufam.edu.br [2] ; saulosilva@super.ufam.edu.br [3] ; hecr@icomp.ufam.edu.br [4] ; roseno.silva@icomp.ufam.edu.br [5] ; miqueias@icomp.ufam.edu.br [6] ; symon@icomp.ufam.edu.br [7] ; williamsonsilva@unipampa.edu.br [8] ; efeitosa@icomp.ufam.edu.br [9]

Inovação

CollabProg é um apoio tecnológico, disponível na internet, que apresenta um conjunto de *guidelines* para apoiar o docente na adoção de MAs no ensino de programação.

- ✓ Fornecerá um conjunto de ***guidelines*** para auxiliar o docente na **adoção** das **MAs**.
- ✓ Apoiará na **identificação** e **escolha** da MA de acordo com o seu **contexto de ensino** .
- ✓ Permitirá a **colaboração** e será **aberto** à comunidade acadêmica.
- ✓ Utilizará a percepção do docente para avaliar a adoção de uma determinada MA.

Publico-alvo

Docentes do ensino superior da área da computação que lecionam disciplinas de programação.

Área de conhecimento

Pode ser utilizado pelos docentes de diversas área, em especial, a área de programação de computadores.

Metodologias Ativas abordadas

(1) Blended Learning, (2) Cooperative Learning, (3) Coding Dojo, (4) Flipped Classroom, (4) Game-Based Learning, (5) Gamification-Based Learning, (6) Gamification-Based Learning, (7) Method 300, (8) Problem-Based Learning , (9) Project-Based Learning, (10) Peer Review, (11) Team-Based Learning, (12) Topdown, (13) Think-Pair-Share e (14) Process Oriented Guided Inquiry Learning

Tecnologia utilizada

Front-end: ReactJS, Lib styled-components e JEST, Vercel e gitHub.

Back-end: Node.js, MongoDB, Mongoose, Swagger, Railway e gitHub.

Abordagem pedagógica

Por se tratar de um ambiente colaborativo e aberto, o CollabProg aborda o modelo 3C de colaboração, que é baseado na concepção de que para colaborar, os membros de um grupo comunicam-se, coordenam-se e cooperam.

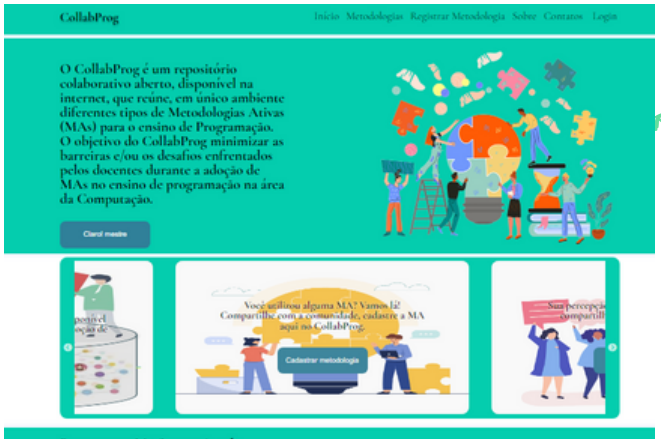
Onde foi desenvolvida

O CollabProg é parte da geração e o uso de conhecimento e a principal contribuição de uma Tese de doutorado desenvolvida nos Programas de Pós-graduação da Univesidade Federal do Amazona (UFAM) e a Universidade Federal do Pampa (Unipampa).

Interface do ColabProg

Acesse agora, clique aqui

Home do Collabprog

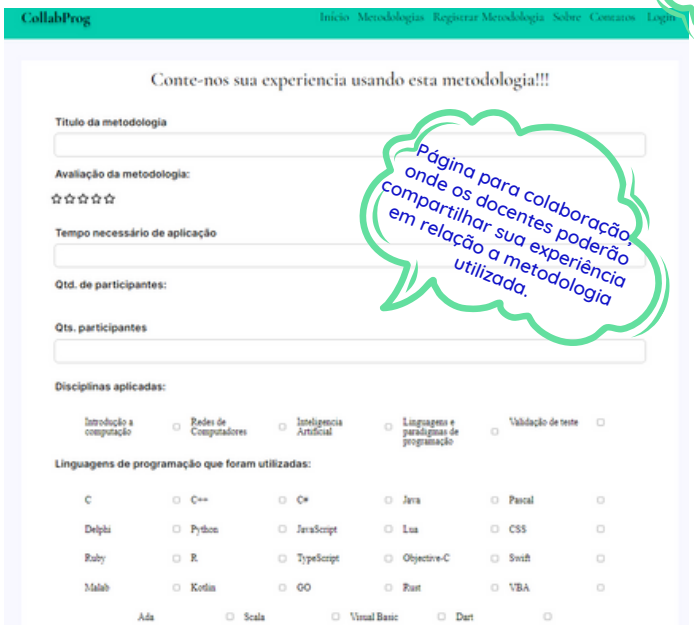


Página que apresenta o detalhamento de como o docente poderá implementar a MA em sua aula.

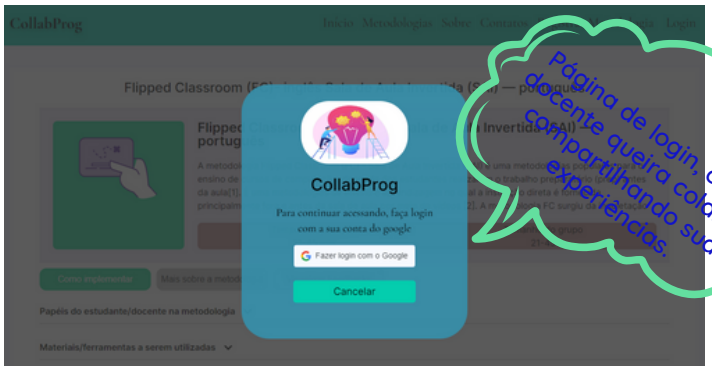
Página que apresenta os feedbacks sobre o adoção das metodologias.

As MAs registradas passarão por um processo de triagem, com critérios de avaliação, que visa garantir a confiabilidade do repositório.

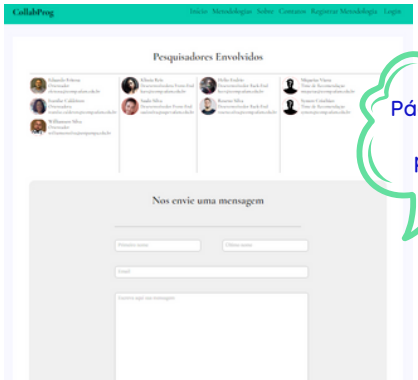
Página para colaboração, onde os docentes poderão cadastrar metodologias que foram utilizadas e suas experiências.



Página para colaboração, onde os docentes poderão compartilhar sua experiência em relação a metodologia utilizada.



Página de login, caso o docente queira colabora compartilhando suas experiências.



Página de contatos com os pesquisadores.

Página com informações sobre a pesquisa.

Agradecimentos - O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Código de Financiamento 001 e parcialmente financiado pela Fundação de Amparo à Pesquisa do Estado do Amazonas – FAPEAM – por meio do projeto POSGRAD. Williamson Silva agradece pelo apoio financeiro da FAPERGS (Projeto ARD/ARC - processo 22/2551-0000606. Ivanilse Calderon agradece ao Instituto Federal de Educação, Ciência e Tecnologia de Rondônia (IFRO)/Campus Porto Velho Zona Norte.

Appendix H

CBIE 2023
CONGRESSO BRASILEIRO DE INFORMÁTICA NA EDUCAÇÃO
Uma escola para o futuro: Tecnologia e conectividade a serviço da educação

Explorando a aceitação do CollabProg como um Facilitador de Metodologias Ativas no Ensino de Programação

Ivanilse Calderon[1,3], Williamson Silva[2], Eduardo Feitosa[1]

[1]Instituto de Computação (IComp) – Universidade Federal do Amazonas (UFAM)
Manaus, AM – Brasil

[2]Programa de Pós-Graduação em Engenharia de Software (PPGES) -
Universidade Federal do Pampa (UNIPAMPA) - Alegrete, RS - Brasil

[3]Instituto Federal de Educação, Ciência e Tecnologia de Rondônia (IFRO)
Campus Porto Velho Zona Norte - Porto Velho, RO - Brasil

{[1,3]ivanilse.calderon,[1]efeitosa}@icomp.ufam.edu.br

[2]williamsonsilva@unipampa.edu.br

Abstract. *There is evidence that Active Methodologies (AM) enable the development of skills and competencies. However, the rate of adoption by teachers is relatively low, especially in teaching programming. To help teachers, we developed CollabProg, an open collaborative repository that brings together, in a single environment, the step-by-step guide on how to lead the adoption of different types of AM for teaching programming. This article describes an exploratory study that aims to evaluate the acceptance of CollabProg, from the perspective of professors. The results show that the professors had a good perception; they could express themselves and easily describe their experiences.*

Resumo. *Existem evidências de que as Metodologias Ativas (MAs) possibilitam o desenvolvimento de habilidades e competências. Contudo, a taxa de adoção pelos docentes é relativamente baixa, especialmente no ensino de programação. Para auxiliar os professores, desenvolvemos o CollabProg, um repositório colaborativo aberto que reúne, em único ambiente, um passo a passo de como conduzir a adoção de diferentes tipos de MAs para o ensino de programação. Este artigo descreve um estudo exploratório que visa avaliar a aceitação do CollabProg, a partir da perspectivas dos docentes.Os resultados mostram que os docentes tiveram boa percepção, pois conseguiram se expressar e descrever suas experiências de forma fácil.*

1. Introdução

O ensino de programação em cursos de Computação é considerado complexo por exigir uma compreensão profunda de conceitos abstratos que ainda não são totalmente compreensíveis aos estudantes [Luxton-Reilly et al. 2018]. Aliadas a isso, continuam sendo ministradas com base na transmissão unidirecional de conhecimento dos docentes para o discentes, o que leva automaticamente ao caminho de uma “aula de doutrinação”. Como consequência, os discentes não se sentem motivados a aprender o conteúdo [Garcia et al. 2021] e, muitas vezes, abandonam as disciplinas e até mesmo o curso

[Sobral 2021]. Logo, os docentes necessitam repensar e adaptar as suas aulas e promover um ambiente educacional mais participativo, centrado no discente e que valorize a construção ativa e o compartilhamento do conhecimento [Calderon et al. 2021].

Tem-se observado que os docentes estão cada vez mais interessados em utilizar e explorar Metodologias Ativas (MAs) em sala de aula [Witt et al. 2018], uma vez que elas envolvem os discentes ativamente no processo de aprendizagem (*learning by doing*) e os direcionam a refletir sobre seu aprendizado como ponto de partida para construção de novos conhecimentos e desenvolvimento de habilidades [Berssanette and de Francisco 2021]. Apesar das evidências positivas que sustentam a eficácia das MAs durante o ensino de programação, a taxa de adoção por parte dos docentes ainda é relativamente baixa [Nguyen et al. 2021]. Diversas barreiras são percebidas durante a adoção das MAs [Tharayil et al. 2018]: falta de tempo para o planejamento de aulas que adotam MAs; dificuldade em cumprir todo o conteúdo do curso; rejeição dos discentes ao uso de novas estratégias pedagógicas em aula; dúvidas quanto à eficácia das MAs para alcançar os objetivos de aprendizagem; falta de informações sobre como implementar MAs nas aulas.

Além destas barreiras, ainda há a resistência dos docentes à mudança devido à familiaridade com abordagens tradicionais de ensino. Isso pode estar relacionado aos fatos que a adoção de MAs nem sempre é direta, requer uma mudança de paradigma e pode exigir um esforço significativo, por parte dos docentes, para se adaptarem a novas práticas. No entanto, poucas pesquisas apresentam soluções ou suporte tecnológico para apoiar o docente a minimizar as barreiras e/ou os desafios enfrentados na adoção das MAs para o ensino de programação, logo, observa-se lacunas para serem pesquisadas.

Diante deste cenário, este artigo descreve a construção de um repositório colaborativo aberto para apoiar na adoção de MAs no ensino de programação denominado CollabProg, guiada pela metodologia de *Design Science Research* (DSR) [Wieringa 2014]. Também apresenta os resultados do primeiro ciclo de *Design*, conduzido com o objetivo de avaliar a aceitação e a viabilidade de uso do CollabProg a partir do ponto de vista de docentes de diferentes instituições de ensino do Brasil.

Os resultados deste ciclo mostram evidências iniciais de que o CollabProg apoia os docentes na adoção de metodologias ativas, além de apresentar as limitações e oportunidades de melhoria. Como contribuição, o CollabProg auxiliará o docente na identificação e escolha de MA(s), de acordo com o seu contexto de ensino e que atenda às suas necessidades reias em sala de aula. Além disso, disponibilizará um conjunto de *guidelines*, que conterão com o passo a passo para guiar os docentes durante a adoção das MAs. Desta forma, os docentes não precisarão buscar, em vários artigos científicos ou livros, formas de como conduzir uma determinada MA em sala de aula, ou seja, o CollabProg reunirá, em um único repositório, um conjunto de estratégias de como conduzir a adoção de diferentes tipos de MAs para o ensino de programação.

2. Trabalhos Relacionados

No contexto educacional, diversos trabalhos foram desenvolvidos com o propósito de criar repositórios digitais voltados para apoiar a prática docente em diversas áreas. A seguir são apresentados os trabalhos que apresentam repositório para apoiar a prática docente.

O portal ALCASYSTEM, desenvolvido por [Castro and Siqueira 2019], é uma

plataforma Web que apoia os docentes na busca, seleção e recomendação de metodologias ativas no contexto da Computação. Para isso, ALCASYSTEM disponibiliza uma variedade de artigos que exploram diferentes abordagens de ensino, além de possuir um fórum para interação entre os docentes. No entanto, os docentes necessitam gerenciar o seu tempo para poder ler e assimilar uma quantidade significativa de artigos recomendados para uso de uma determinada metodologia ativa, o que pode impactar na não efetiva adoção destas pelos docentes.

Silva *et al.* (2020) apresentam o portal OpenSMALS, um repositório aberto para ensino de modelagem de software por meio de metodologias ativas. O OpenSMALS fornece um conjunto de orientações específicas sobre como implementar MAs, bem como artefatos compartilhados por outros docentes, questionários de avaliação, etc. Os autores conduziram um estudo empírico envolvendo cinco docentes e 163 estudantes para avaliar a eficácia do do portal. Os resultados revelaram que o OpenSMALS ajudou os docentes a implementar as metodologias ativas. Contudo, o repositório possui um conjunto limitado de MAs (apenas oito) e com foco em um conteúdo específico, modelagem de software.

Lima et al. (2021) apresentam um guia de seleção que fornece aos docentes uma ferramenta para auxiliá-los na escolha assertiva de uma MA, a partir da identificação do perfil e estilo de aprendizagem do estudante. A primeira versão foi avaliada por meio de sessões de grupo focal. Os resultados mostram indícios de utilidade, clareza, facilidade de uso, organização, flexibilidade, adequação, visualização e seleção de MAs em diferentes contextos de ensino. Observa-se que o guia é específico para ES, traz apenas dez tipos de MAs, é disponibilizado em arquivo digital e não permite interação entre a comunidade.

Ahshan (2021) apresenta um *framework* que implementa atividades / estratégias para garantir o envolvimento ativo dos estudantes durante a pandemia. O *framework* combina o uso equilibrado de pedagogia de ensino ajustada, tecnologias educacionais e um sistema de gerenciamento de *e-learning*. Os resultados da pesquisa indicam que combinar as tecnologias utilizadas, ensino síncrono e atividades de aprendizado ativo na estrutura desenvolvida é eficaz para aprendizagem interativa. Contudo, não apresenta experiências e/ou avaliações de outros docentes em relação às atividades/estratégias apresentadas e foca no contexto do ensino remoto.

Esses trabalhos se concentram em fornecer suporte aos docentes. No entanto, é importante ressaltar que até o momento não foram identificados trabalhos específicos voltados para o ensino de programação. Portanto, o presente artigo traz uma contribuição pioneira nesse domínio, o CollabProg.

3. CollabProg

O CollabProg é um repositório colaborativo que tem por objetivo apoiar os docentes durante a adoção de MAs no ensino de programação. Para tanto, disponibiliza um conjunto de *guidelines* específicos que descrevem os passos para que as MAs sejam adotadas em sala de aula. Em relação às MAs que compõem o CollabProg, buscamos agrupar o conhecimento sobre cada metodologia em um modelo conceitual inspirado na proposta de Sobrinho et al. (2016) e Silva *et al.* (2020). Inicialmente, definimos o domínio e escopo do conhecimento que seria construído a partir dos resultados do estudo. O domínio é a representação e formalização semântica das metodologias de ensino baseadas em princípios de aprendizagem ativa [Sobrinho et al. 2016]. O escopo deste modelo é for-

necer suporte aos docentes no ensino de programação, por meio do conhecimento organizado e representado semanticamente, facilitando sua difusão e uso de MAs.

Na primeira versão do CollabProg, que pode ser acessado *online*[1] , o repositório está dividido em três *menus* rotulados, os quais dispõem de informações para a utilização do CollabProg por parte do usuário. O usuário poderá navegar livremente, selecionar e adotar qualquer MA disponível no repositório. O acesso ao CollabProg não exige nenhum cadastro (aberto), uma vez que apresenta-se como um apoio tecnológico que reúne um detalhamento de como conduzir a adoção de diferentes MAs no ensino de programação. Nele, o docente encontrará informações pertinentes sobre as MAs, incluindo exemplos de adoção, opções de ferramentas adotadas pela comunidade, experiências em cenários diferentes, os resultados alcançados por outros docentes, bem como pontos positivos e negativos sobre a MA adotada. A Figura 1 ilustra a primeira versão do CollabProg com um recorte sobre uma determinada metodologia ativa, a POGIL. A Parte 01 da Figura 1 apresenta uma breve descrição do CollabProg, a Parte 02 apresenta uma breve descrição sobre a metodologia ativa escolhida pelo docente, neste caso a POGIL. Por fim, a Parte 03 apresenta explicações mais detalhadas sobre a metodologia, bem como os papéis existentes na metodologia, os passos para adoção e detalhamento de cada passo. Conforme dito, no site há mais informações que podem auxiliar os professores em sua prática docente.

Figura 1. Primeira Versão do CollabProg.

É importante mencionar que o *menu* **Registrar Metodologia**, utilizado pelo do-

[1]https://acesse.one/OKSqX

cente para cadastrar metodologias, estava em planejamento e não disponível para a avaliação dos participantes do estudo exploratório do CollabProg. As MAs cadastradas passaram pela curadoria do CollabProg, com base em critérios de qualidade pré-estabelecidos, relacionadas a clareza e o detalhamento sobre o uso de uma MA.

4. Estudo Exploratório

No ciclo de *Design* (DSR) é essencial que os *stakeholders* diretamente relacionados ao contexto em que o problema está inserido avaliem o artefato [Wieringa 2014]. Neste sentido, foi realizado um estudo exploratório a fim de verificar a viabilidade de uso e aceitação de docentes sobre o ColabProg.

4.1. Planejamento

O objetivo deste estudo é avaliar a viabilidade de uso e aceitação do CollabProg sob o ponto de vista de docentes. Os pesquisadores recrutaram, por conveniência, docentes de diversos locais do país. Devido ao distanciamento geográfico entre os participantes, os artefatos do estudo tiveram que ser adaptados. Os artefatos usados foram elaborados a partir das ferramentas *on-line* disponíveis via Google *Workspace*, sendo eles: (i) termo de consentimento garantindo a confidencialidade dos dados fornecidos e o anonimato dos docentes (Parecer Comitê de Ética Nº 4.694.031) [2]; (ii) questionário de caracterização para conhecer a experiência dos docentes em sala de aula e no uso de MAs[3]; (iii) documentos contendo o roteiro do estudo, instruções de uso do CollabProg e salas *online* para realização de experimentos[4]; (iv) versão inicial do portal*web* do CollabProg [5]; (iv) modelo de um plano de aula[6]; e (v) formulário de pós-uso baseado nos indicadores do Modelo de Aceitação de Tecnologia (TAM)[7].

4.2. Participantes

Foram recrutados cinco docentes de instituições de ensino superior, que participaram voluntariamente do estudo. A Tabela 1 apresenta uma visão geral dos perfis dos docentes.

Tabela 1. Síntese do perfil dos participantes do estudo

ID	Universidade	Cursos	Experiência	Usa MA	Período	MA
D1	Universidade Federal do Pampa (UNIPAMPA)	Ciência da Computação e Engenharia de Software	1 ano	Não	-	-
D2	Instituto Federal do Amazonas (IFAM)	Ciência da Computação e Informática	3 anos	Sim	3 meses	PP
D3	Universidade Tecnologica Federal do Paraná (UTFPR)	Engenharia de Software e Ciência da Computação	5 anos	Não	-	-
D4	Universidade Estadual de Maringa (UEM)	Ciências da Computação e Engenharia	8 anos	Não	-	-
D5	Instituto Federal de Rondônia (IFRO)	Redes de Computadores e Sistemas para Internet	10 anos	Sim	5 anos	ABP

[8]PP - Programação em Pares; ABP - Aprendizagem Baseada em Problemas

Nota-se que apenas dois participantes usaram MAs em sala de aula, demonstrando a falta de aproveitamento das potencialidades das MAs no ensino de programação.Quanto

[2]https://acesse.one/MxxPA
[3]https://l1nk.dev/Eq9Fo
[4]https://l1nk.dev/gqPRQ
[5]https://acesse.one/jsaCp
[6]https://l1nk.dev/KRqmh
[7]https://l1nk.dev/b7Tem

a motivação para adoção de MA, os docentes relataram que é em virtude de prover mais autonomia aos estudantes e facilitar o processo de aprendizagem, uma vez que programação exige bastante raciocínio e um certo nível de abstração ou mesmo fazer o estudante o protagonista de seu aprendizado. Nenhum dos docentes usa ou fez uso de ferramentas educacionais que os ajudassem no uso de tais MAs em sala de aula.

4.3. Execução

A execução do estudo foi totalmente *on-line* e individual. Antes do *primeiro* ciclo do DSR, decidiu-se realizar um estudo piloto afim de verificar se o estudo alcançaria seu objetivo. Os resultados do piloto foram satisfatórios e não houve necessidade de aprimorar o roteiro do estudo. Cada docente foi convidado via e-mail, que descrevia o objetivo do estudo e algumas orientações norteadoras. Caso aceitassem, um dos pesquisadores combinava uma data para a condução individual do estudo. Após o aceite, o estudo foi conduzido seguindo as etapas detalhadas na Figura 2. Cada etapa é explicada a seguir.

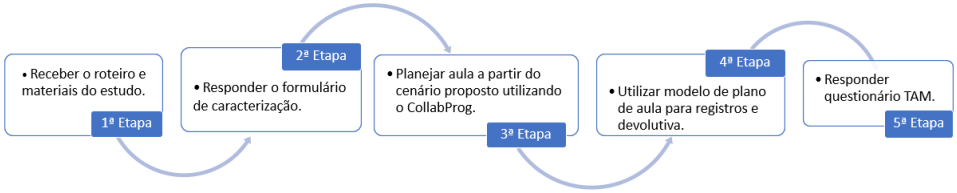


Figura 2. Etapas realizadas no estudo para avaliar o CollabProg.

Na data agendada com o docente, um *link* para um documento com o roteiro de preparação foi enviado via *chat* da sala *online*. Nesse documento estavam disponíveis o formulário *online* do termo de consentimento e o formulário de caracterização para os docentes responderem. Este último possuia perguntas para caracterizar a experiência do docente em relação à adoção das MAs no ensino de programação. É importante ressaltar que a participação de avaliação do CollabProg foi voluntária e todos os participantes assinaram o termo de consentimento, com o qual concordaram participar do estudo e em fornecer os resultados para análise. Após preencherem os questionários, os docentes receberam instruções e explicações sobre o estudo. No roteiro, os docentes deveriam planejar uma aula utilizando uma MA para ensinar o conteúdo de "Variáveis e Constantes" de uma tipica disciplina de Programação I.

Para isso, foi disponibilizado aos docentes: (a) um modelo de plano de aula, ao qual deveria ser preenchido; (b) a versão *on-line* do CollabProg, que deveria ser utilizado como forma de apoio na criação do plano de aula por meio das diretrizes e *guidelines* disponíveis no repositório. Ao final, os docentes disponibilizaram a versão do planejamento. Nota-se que o foco não era avaliar se o plano estava correto ou não, mas saber se o CollabProg ajudou os docentes a planejar a metodologia em todas as etapas da aula. Ressaltamos que os docentes estavam livres para escolher as metodologias que mais se adequavam ao seu conhecimento (teórico e prático), habilidades e, possivelmente, ao seu contexto de ensino. Após realizar o planejamento, o docente era convidado a responder a um questionário de avaliação, no qual apontava sua experiência após o uso do CollabProg.

4.4. Análise dos dados

O questionário de avaliação do CollabProg foi definido com base nos indicadores do Modelo de Aceitação da Tecnologia (TAM) [**?**]. O TAM é um questionário projetado para

obter informações sobre a percepção dos participantes em relação aos principais fatores que influenciam a aceitação ou rejeição de uma determinada tecnologia. Os indicadores definidos foram: (i) **Utilidade Percebida**, que define o grau que o docente acredita que o CollabProg pode melhorar seu desempenho na adoção da MAs; (ii) **Facilidade de Uso Percebida**, que define o grau que o docente acredita que ao usar o CollabProg seria livre de esforço; e (iii) **Intenção de Uso Percebida**, que define o grau que o docente acredita que poderá utilizar o CollabProg no futuro. A razão para focar nestes indicadores é que estes são fortemente correlacionados com a aceitação do CollabProg pelos docentes.

Com base no uso do CollabProg, os docente forneceram suas percepções de acordo com o nível de concordância em relação às afirmativas estabelecidas no TAM. Cada afirmativa deveria ser respondida com base em uma escala Likert de cinco pontos, variando de Discordo Totalmente a Concordo Totalmente. A Tabela 2 apresenta as afirmativas respondidas pelos docentes e baseadas nos indicadores do TAM. Além disso, foi acrescentada duas questões abertas para permitir um melhor entendimento das respostas dos docentes. A partir das respostas recebidas, foi conduzida uma análise qualitativa empregando técnicas de codificação.

Tabela 2. Perguntas a serem respondidas pelos docentes

Utilidade Percebida	
UP1	Usar o repositório ColabProg melhorou o meu desempenho em relação ao planejamento das aulas adotando MAs
UP2	Usar o repositório ColabProg melhorou a minha produtividade em relação adoção de MAs
UP3	Usar o repositório ColabProg me permitiu relatar completamente os aspectos da minha experiência na adoção das MAs
UP4	Eu acho o repositório ColabProg útil para relatar minha experiência da adoção de MAs
Facilidade de Uso Percebida	
FUP1	O repositório ColabProg foi claro e fácil de entender
FUP2	Usar o repositório ColabProg não demandou muito esforço mental
FUP3	Eu acho que o repositório ColabProg é fácil de usar
FUP4	Eu acho fácil relatar a minha experiência de adoção das MAs usando o repositório ColabProg
Intenção de Uso Percebida	
IUP1	Assumindo que eu tenha acesso ao repositório ColabProg, eu pretendo usá-lo para aplicar MAs no ensino de programação
IUP2	Dado que eu tenha acesso ao repositório ColabProg, eu prevejo que eu o usaria para me apoiar na adoção de MAs no ensino de programação
IUP3	Eu pretendo usar o repositório ColabProg para avaliar a minha experiência com a adoção de uma MAs no próximo mês
Questões Abertas	
QA1	Quais foram os principais desafios / pontos negativos percebidos por você ao utilizar o ColabProg?
QA2	Quais foram os principais pontos positivos que você percebeu ao utilizar o ColabProg?

5. Resultados e Discussões

Esta seção apresenta os resultados qualitativos e quantitativos do estudo, bem como os discussão dos resultados.

5.1. Resultados Quantitativos do Questionário TAM

A Figura 3 apresenta os resultados gerais das percepções dos participantes sobre o CollabProg, conforme as afirmativas do TAM apresentadas na Tabela 2, para conhecer a experiência dos professores quanto à utilidade, facilidade e intenção de uso do respositório.

Em relação às percepções dos docentes sobre a **Utilidade Percebida** do CollabProg, observa-se que em todas as afirmativas (UP1, UP2, UP3, UP4), todos os docentes concordam totalmente com a utilidade do CollabProg para o planejamento das aulas de ensino de programação com a adoção de MAs. Além disso, ficou evidente que o CollabProg se mostra como uma ferramenta capaz de potencializar ou apoiar a produtividade do docentes em sua prática. Também é possível perceber que o CollabProg se configura

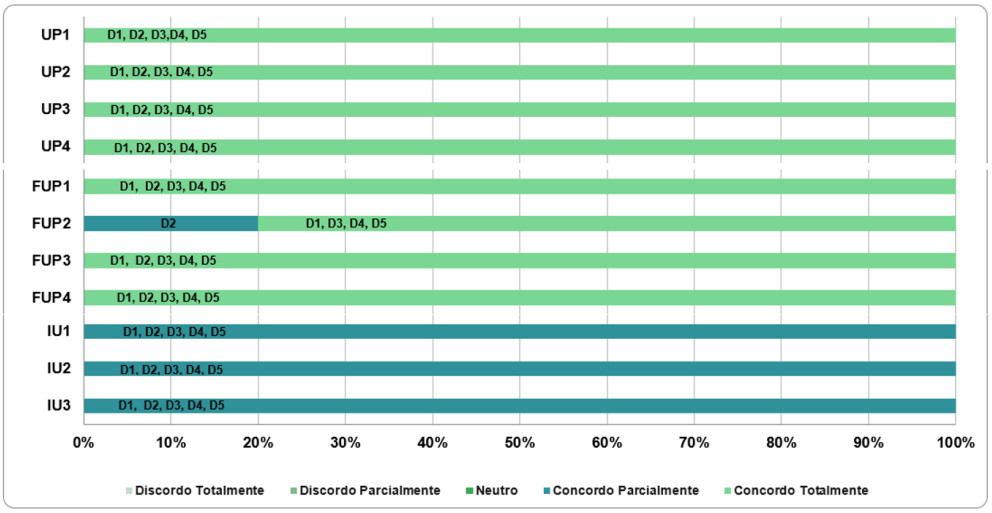


Figura 3. Resultados gerais das percepções sobre o CollabProg.

como um suporte ferramental capaz de permitir que o docente utilize suas experiências ao selecionar uma MA para adotar em suas aulas. Por fim, os resultados das percepções dos participantes refletem uma boa aceitação por parte dos docentes do CollabProg como apoio na adoção de MAs no ensino de programação.

Sobre a **Facilidade de Uso Percebida** do CollabProg, nota-se que houve concordância total nas três afirmativas (FUP1, FUP2 e FUP3) Os docentes disseram que é fácil relatar suas experiências de adoção de MAs usando o CollabProg. Também apontaram que seu uso não exigiu muito esforço mental, sendo de fácil entendimento e uso, especialmente no que diz respeito às necessidades do dia a dia da prática docente no ensino de programação. De modo geral, todos os docentes consideram o CollabProg claro e fácil de entender, bem como fácil de usar. A única exceção foi na afirmativa FUP2, onde D2 concordou parcialmente em relação à facilidade de uso do CollabProg.

Quanto a **Intenção de Uso Futuro Percebida** do CollabProg, todos os docentes concordaram parcialmente com as três afirmativas (IU1, IU2, IU3). A intenção de uso do CollabProg é importante para verificar a disponibilidade e o interesse da comunidade em relação à ferramenta, bem como sua aceitação como apoio ao ensino de programação por parte dos docentes. Sendo assim, observamos que os docentes avaliaram positivamente e manifestaram a intenção de utilizar o repositório CollabProg.

5.2. Percepções dos participantes sobre o uso do CollabProg

Para avaliar a experiência dos docentes em relação à utilização do CollabProg, analisamos a seguinte frase: *"utilizar o CollabProg contribuiu para a adoção de metodologia ativa em minhas aulas de ensino de programação?"*. De modo geral, a percepção foi positiva, sendo apontados aspectos relevantes em relação à experiência da utilização do CollabProg. No que diz respeito aos pontos positivos, foram identificadas quatro subcategorias que abordam os benefícios do repositório.

Na primeira subcategoria, **explicação detalhada das etapas**, D1 comentou que: *"não tinha conhecimento sobre as metodologias e o CollabProg me permitiu aplicá-las de forma fácil"*. D3 afirmou que o: *"CollabProg facilitou bastante a compreensão sobre*

as metodologias ativas disponíveis no repositório". D3 também comentou que, em outros momentos, pretendia usar o POGIL em suas aulas, mas a sua documentação é muito extensa e que *"a forma como foi apresentada no CollabProg foi bem mais intuitiva para compreender o funcionamento dessa MA e planejar as aulas"*. Na segunda subcategoria, **aumento da produtividade na implementação de MA**, D2 evidenciou que *"sem o CollabProg dificilmente eu iria atrás dos detalhes de uma MA para ensinar programação"*.

Em relação à terceira subcategoria, **utilidade dos exemplos práticos**, D3 expos que *"os exemplos apresentados foram muito úteis para compreender melhor como podemos adotar a metodologia"*. O docente complementou dizendo que, muitas vezes, lemos sobre as metodologias, mas fica tudo muito abstrato e que *"ter os passos envolvidos e os exemplos torna muito mais fácil compreender como aplicar a metodologia"*. Por fim, em relação ao **estímulo ao trabalho colaborativo e à participação ativa dos alunos**, quarta subcategoria, D4 afirmou que *"trabalhos colaborativos enriquecem o aprendizado"*, e D5 compartilhou que *"a principal vantagem, na minha opinião, é fazer com que o aluno participe mais ativamente das aulas e, consequentemente, tenha um melhor aprendizado"*.

As percepções apresentadas pelos docentes, além de contribuírem com a evolução do repositório, confirmaram o interesse pela utilização desta ferramenta. Em relação à utilidade do CollabProg, todos os docentes consideraram que o repositório CollabProg pode contribuir para o planejamento das aulas no que tange à adoção de MAs. Enquanto isso, a maioria consideraram que os conteúdos apresentados no repositório são úteis.

Também foram identificados alguns pontos negativos pelos docentes. O primeiro está relacionado a **dificuldade na compreensão das etapas e conceitos**. D1 comentou sentir dificuldade em *"compreender as metodologias ativas (algumas etapas tive que ler diversas vezes)"*. D2 complementou dizendo que *"apesar de estar muito bem organizado, ainda senti dificuldades em montar o passo-a-passo."* O segundo ponto negativo é referente aos **desafios na montagem do passo-a-passo e confusão em pontos específicos**. Neste sentido, D3 enfatizou que *"alguns pontos ficaram confusos durante a leitura da metodologia ativa, em especial o POGIL, que adotei"*. D5 disse que a sua principal dificuldade *"foi construir um plano de aula que refletisse a metodologia ativa em questão"*.

Com base em suas experiências, os docentes destacaram algumas sugestões para melhorar a utilização do ColabProg. A primeira foi buscar maior clareza e simplicidade nas explicações das etapas e conceitos das metodologias, tornando-as mais acessíveis e fáceis de compreender. Mencionaram a necessidade de orientações práticas para a contribuição com os planos de aula e o uso do ColabProg pelos docentes. Destacaram a importância de aprimorar a documentação e os exemplos das metodologias, tornando-os mais claros e abrangentes. Também sugeriram ter uma explicação mais detalhada sobre a atribuição de papéis nas MAs, a fim de evitar confusões e facilitar a implementação.

6. Considerações Finais

Este artigo apresentou os resultados do primeiro ciclo de *Design* conduzido com o objetivo de avaliar a aceitação e a viabilidade de uso do CollabProg, um repositório colaborativo e aberto para apoiar a adoção de MAs no ensino de programação, a partir do ponto de vista de docentes de diferentes instituições de ensino do Brasil. A partir dos resultados alcançados buscamos evoluir e realizar melhorias no CollabProg, especialmente nos pontos negativos e necessidades apontadas pelos docentes. Realizamos novo ciclo de *design*

e elaboramos a segunda versão do CollabProg, acessada *online*[9] e que parte dela pode ser vista na Figura 4. Nesta versão, o CollabProg está dividido em cinco *menus* rotulados, os quais dispõem de informações para direcionar o usuário. Nesta versão, forma mantidas as funcionalidade da primeira versão, contudo, já passou-se a apresentar o *menu* **Registrar Metodologia**, para o cadastro de metodologias.

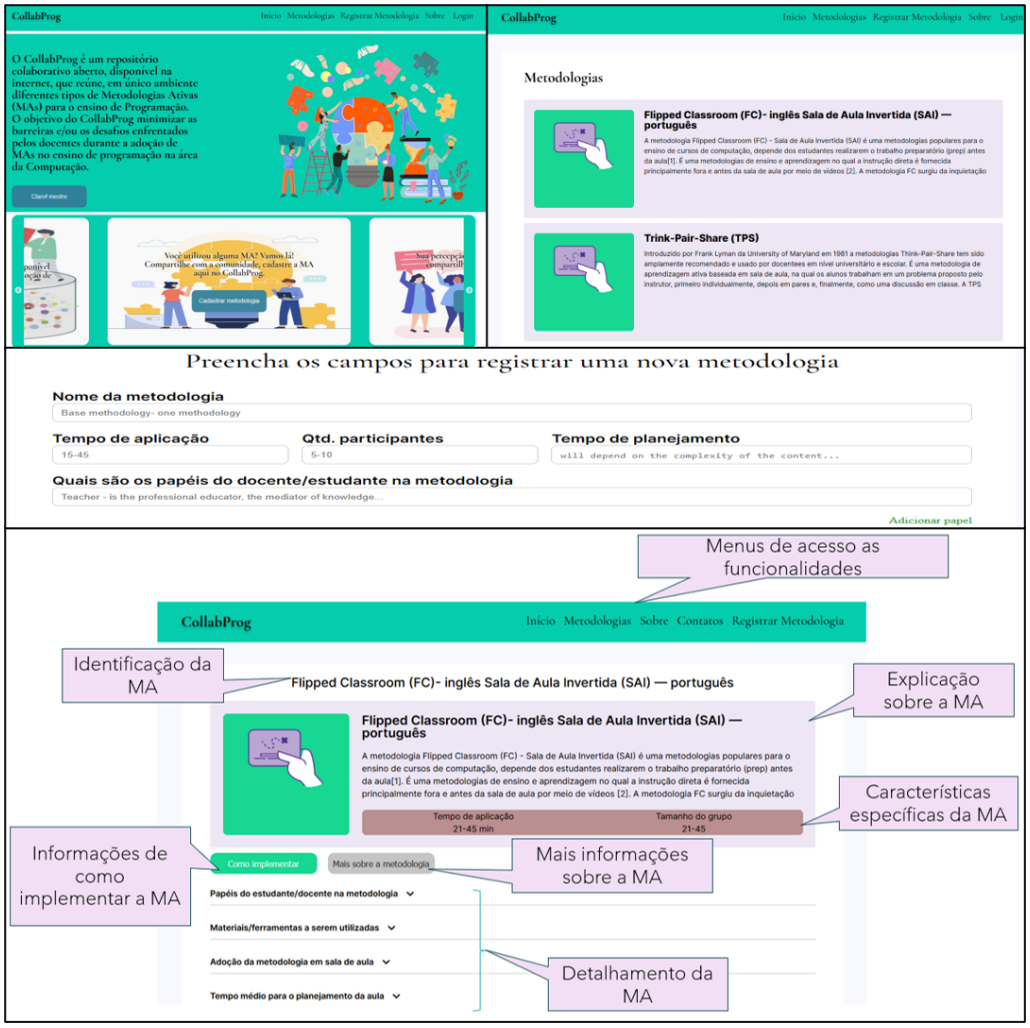


Figura 4. Segunda versão do CollabProg.

Como trabalhos futuros, pretende-se disponibilizar uma função de **Recomendação** das MAs, permitindo que o docente informe algumas características sobre a turma, o conteúdo a ser ensinado, disciplina, entre outras informações, para que o CollabProg possa recomendar a MA que melhor se adequará ao cenário informado.

Agradecimentos

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Código de Financiamento 001 e parcialmente financiado pela Fundação de Amparo à Pesquisa do Estado do Amazonas – FAPEAM – por meio do projeto POSGRAD. Williamson Silva agradece pelo apoio financeiro da FAPERGS (Projeto ARD/ARC - processo 22/2551-0000606. Ivanilse Calderon agradece ao

[9]https://l1nq.com/B4rZg

Instituto Federal de Educação, Ciência e Tecnologia de Rondônia (IFRO)/Campus Porto Velho Zona Norte.

Referências

Ahshan, R. (2021). A framework of implementing strategies for active student engagement in remote/online teaching and learning during the covid-19 pandemic. *Education Sciences*, 11(9):483.

Berssanette, J. H. and de Francisco, A. C. (2021). Active learning in the context of the teaching/learning of computer programming: A systematic review. *Journal of Information Technology Education. Research*, 20:201.

Calderon, I., Silva, W., and Feitosa, E. (2021). Um mapeamento sistemático da literatura sobre o uso de metodologias ativas durante o ensino de programação no brasil. In *Anais do XXXII Simpósio Brasileiro de Informática na Educação*, pages 1152–1161. SBC.

Castro, R. M. d. and Siqueira, S. (2019). Alcasystem-um portal com técnicas de aprendizagem ativa para disciplinas da área da computaçao. In *Anais dos Workshops do Congresso Brasileiro de Informática na Educação*, volume 8, page 1243.

Garcia, F. W. D. S., Carvalho, E. D. C., and Oliveira, S. R. B. (2021). Use of active methodologies for the development of a teaching plan for the algorithms subject. In *2021 IEEE Frontiers in Education Conference (FIE)*, pages 1–9. IEEE.

Lima, J., Alencar, F., and Santos, W. (2021). A preliminary guide for assertive selection of active methodologies in software engineering education. In *Brazilian Symposium on Software Engineering*, pages 170–179.

Luxton-Reilly, A., Albluwi, I., Becker, B. A., Giannakos, M., Kumar, A. N., Ott, L., Paterson, J., Scott, M. J., Sheard, J., and Szabo, C. (2018). Introductory programming: a systematic literature review. In *Proceedings Companion of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education*, pages 55–106.

Nguyen, K. A., Borrego, M., Finelli, C. J., DeMonbrun, M., Crockett, C., Tharayil, S., Shekhar, P., Waters, C., and Rosenberg, R. (2021). Instructor strategies to aid implementation of active learning: a systematic literature review. *International Journal of STEM Education*, 8:1–18.

Silva, W., Gadelha, B., Steinmacher, I., and Conte, T. (2020). Towards an open repository for teaching software modeling applying active learning strategies. In *2020 IEEE/ACM 42nd International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET)*, pages 162–172. IEEE.

Sobral, S. R. (2021). Strategies on teaching introducing to programming in higher education. In *World Conference on Information Systems and Technologies*, pages 133–150. Springer.

Sobrinho, H., Castro, L., Nogueira, A., Harada, E., and Gadelha, B. (2016). Organizando o conhecimento sobre técnicas de aprendizagem colaborativas. *Nuevas Ideas em Informatica Educativa*, 12:152–156.

Tharayil, S., Borrego, M., Prince, M., Nguyen, K. A., Shekhar, P., Finelli, C. J., and Waters, C. (2018). Strategies to mitigate student resistance to active learning. *International Journal of STEM Education*, 5(1):1–16.

Wieringa, R. J. (2014). *Design science methodology for information systems and software engineering*. Springer.

Witt, D. T., Kemczinski, A., and dos Santos, L. M. (2018). Resolução de problemas: Abordagens aplicadas no ensino de computação. *Anais do Computer on the Beach*, pages 731–740.

Uma Plataforma Web para apoiar Docentes no Ensino de Programação em Cursos de Sistemas de Informação

Ivanilse Calderon[1,3], Williamson Silva[2], Eduardo Feitosa[1]

[1]Instituto de Computação (IComp) – Universidade Federal do Amazonas (UFAM)
Manaus, AM – Brasil

[2]Laboratory of Empirical Studies in Software Engineering (LESSE) - Departamento de Engenharia de Software - Universidade Federal do Pampa (UNIPAMPA)
- Alegrete, RS - Brasil

[3]Instituto Federal de Educação, Ciência e Tecnologia de Rondônia (IFRO)
Campus Porto Velho Zona Norte - Porto Velho, RO - Brasil

{[1,3]ivanilse.calderon,[1]efeitosa}@icomp.ufam.edu.br

[2]williamsonsilva@unipampa.edu.br

Abstract. *Teaching programming in Information Systems courses is not just about fulfilling a step in the academic curriculum or merely about writing code with students, but rather a journey to prepare them for job market opportunities and to empower them to be agents of change and innovation. Active methodologies are strategies that can assist in this process. Educators face barriers and challenges in adopting active methodologies to enhance their teaching practices in programming. This work presents CollabProg, an open web platform available online, which brings together different types of active methodologies in a single environment to support educators in programming teaching.*

Resumo. *Ensinar programação nos cursos de Sistemas de Informação não é apenas cumprir uma etapa no currículo acadêmico ou sobre escrever linhas de código com os estudantes, mas uma jornada para preparar os estudantes para as oportunidades do mercado de trabalho e para capacitá-los a serem agentes de mudança e inovação. As metodologias ativas são estratégias que podem ajudar neste processo. Os docentes enfrentam barreiras e desafios para a adoção de metodologias ativas para melhorar suas praticas docentes em relação ao ensino de programação. Este trabalho apresenta o CollabProg, uma plataforma web aberta, disponível na internet, que reúne, em único ambiente diferentes tipos de metodologias ativas para apoiar os docentes no ensino de programação.*

1. Introdução

O ensino de programação é fundamental no currículo de cursos de Sistemas de Informação (SI) e representa um desafio para os docentes [Imbulpitiya et al. 2019]. Ao ensinar programação, espera-se capacitar os estudantes com habilidades essenciais para o mundo digital, que está em constante evolução. No entanto, a natureza dinâmica da tecnologia exige que os educadores busquem recursos atualizados e eficazes para o ensino. Isso ocorre porque tais disciplinas demandam conhecimentos prévios em lógica, matemática, leitura, interpretação de texto, abstração de ideias, entre outras habilidades

[Bigolin et al. 2020]. As Metodologias Ativas (MAs) têm despertado o interesse dos docentes para a prática pedagógica em sala de aula. Segundo Koenig (2020), as MAs têm suas bases na teoria Construtivista, que busca tornar o estudante protagonista de sua própria aprendizagem. Isso auxilia os estudantes a desenvolver habilidades críticas, refletir sobre suas práticas, além de explorar atitudes, valores pessoais e o aprendizado por meio da prática (*learning by doing*) [Ribeiro et al. 2021].

Este trabalho descreve o desenvolvimento de uma plataforma web colaborativa e aberta, denominada CollabProg, destinada a apoiar os docentes na adoção de MAs no ensino de programação. Na seção 2, são abordados os trabalhos relacionados, enquanto a seção 3 apresenta o referencial teórico. O CollabProg é detalhado na seção 4. Por fim, as considerações finais e os trabalhos futuros são abordados na seção 5.

2. Trabalhos relacionados

Pesquisadores têm se empenhado em aprimorar a adoção de MAs por meio de novas tecnologias de comunicação e instrução. No contexto educacional, observam-se pesquisas para a criação de repositórios digitais que apoiam a prática docente em diversas áreas. O portal ALCASYSTEM [de Castro and Siqueira 2019] é um exemplo disso, sendo uma plataforma web que auxilia os docentes na busca, seleção e recomendação de MAs no contexto da Computação. Outro exemplo é o portal OpenSMALS [Silva et al. 2020], um repositório aberto voltado para o ensino de modelagem de software por meio de MAs. Nesse contexto, Lima et al., (2021) criaram um guia de seleção destinado a auxiliar os docentes na escolha de MAs com base no perfil e estilo de aprendizagem dos estudantes, é específico para a engenharia de software. Ahshan (2021) apresenta um *framework* para implementar atividades e estratégias que promovam o engajamento dos estudantes, combinando tecnologias educacionais e um sistema de gerenciamento de *e-learning*.

3. Referêncial teórico

Dentre os grandes desafios na área de Sistemas de Informação (SI), observa-se a necessidade de aprimorar a formação para desenvolver habilidades, conhecimento e atitudes profissionais, além de analisar os conteúdos das disciplinas [Cafezeiro et al. 2017]. O ensino de programação desenvolve habilidades que abrem portas para uma ampla gama de oportunidades profissionais e acadêmicas na área de Tecnologia da Informação (TI). Contudo, o ensino de programação nos cursos de SI é considerado complexo, pois requer uma compreensão profunda de conceitos abstratos que ainda não são totalmente compreensíveis aos estudantes [Luxton-Reilly et al. 2018]. Neste contexto, nota-se que os docentes dos cursos de SI podem enfrentar diversas barreiras, como a falta de tempo para o planejamento das aulas [Michael 2007], dificuldades em cumprir todo o conteúdo da disciplina [Eickholt 2018], entre outras na adoção das MAs para o ensino de programação. Para mitigar esse problema, os docentes precisam de um ambiente onde possam entender e, consequentemente, adaptar ou empregar novas estratégias de ensino, oferecendo, assim, um ambiente de aprendizagem desafiador e engajador para os estudantes.

4. O desenvolvimento do CollabProg

4.1. Método

Para o desenvolvimento do CollabProg, utilizou-se o *Design Science Research* (DSR), escolha justificada pela conexão entre conhecimento e prática [Wieringa 2009]. Segundo

Hevner et al., (2004) o DSR procura identificar e compreender os problemas do mundo real e propor soluções apropriadas e úteis, fazendo avançar o conhecimento teórico da área. O DRS é dividida em três ciclos. No Ciclo de Relevância foi definido o problema investigado e a motivação para a pesquisa. No Ciclo de Design desenvolveu-se, avaliou-se e evoluiu-se o CollabProg. Por fim, no Ciclo de Rigor refere-se aos principais fundamentos relacionados ao conhecimento sobre a adoção de MAs para o ensino de programação, às estratégias para adoção das MAs e aos estudos experimentais.

4.2. Processo de curadoria e seleção das metodologias

O processo de curadoria das MAs que compõem o CollabProg priorizou o compartilhamento apenas dos conteúdos e das opções de suporte ferramental disponíveis na literatura, para a utilização dos docentes. Deste modo, busca-se evitar possíveis frustrações para os usuários desta plataforma web, visto que não serão apresentados conhecimentos e/ou conteúdos sem base científica, sem experimentos ou irrelevantes. A triagem realizada para a seleção das MAs teve como critério principal a escolha de metodologias respaldadas por evidências científicas sólidas, eliminando a adoção de metodologias desprovidas de experimentos ou cujo embasamento teórico se mostrasse irrelevante para a comunidade, diante do contexto desta pesquisa. O detalhamento do protocolo definido e utilizado para a Avaliação de Qualidade (AQ) dos estudos primários pode ser visto *on-line*[1].

4.3. Primeira versão do CollabProg

A Figura 1 ilustra a primeira versão do CollabProg com um recorte sobre uma determinada metodologia ativa, a POGIL. A Parte 01 da Figura 1 apresenta uma breve descrição do CollabProg, a Parte 02 apresenta uma breve descrição sobre a MA escolhida pelo docente, neste caso a POGIL. Por fim, a Parte 03 apresenta explicações mais detalhadas sobre a metodologia, bem como os papéis existentes na metodologia, os passos para adoção e detalhamento de cada passo.

4.4. Avaliação do CollabProg

Para avaliar a viabilidade de uso e aceitação da segunda versão do CollabProg, foi realizado um estudo exploratório, com a participação de docentes de todo o Brasil, utilizando o Modelo de Aceitação da Tecnologia (TAM) [Silva 2015]. Os resultados deste estudo podem ser consultados em Calderon et al., (2023). Com base na avaliação do CollabProg e nas sugestões dos docentes, concluímos que o repositório representa uma ferramenta valiosa para apoiar o ensino de programação com MAs. Embora tenhamos recebido elogios pela sua capacidade de motivar a adoção das MAs e facilitar a compreensão de seu funcionamento, também foram identificadas oportunidades de melhoria em relação à clareza e simplicidade das explicações, à documentação das metodologias e à explicação dos papéis atribuídos em cada uma delas.

4.5. Segunda versão do CollabProg

A Figura 2 apresenta um recorte da segunda versão. Dividida em cinco menus rotulados, os quais dispõem de informações para direcionar o usuário. A Parte 1 apresenta a página inicial e o detalhamento do menu Metodologias (identificação da MA e sua descrição).

[1]https://figshare.com/s/794c9f7e5adfdff915d1

Figura 1. Primeira versão CollabProg

Na Parte 2 são apresentadas informações específicas da MA selecionada. Nesta parte, o usuário, de modo objetivo, consegue verificar se a MA atenderá ou não ao seu contexto de ensino, considerando o tempo de aula, a quantidade de estudantes, quais serão os papéis a serem desempenhados na aula, dentre outras informações que contribuirão para o docente decidir por qual MA adotar. Por fim, na Parte 3 são apresentados dois *submenus*: "Como implementar"e "Mais sobre a metodologia". Nestes, o usuário encontrará um direcionamento detalhado e objetivo sobre a MA selecionada, facilitando o entendimento de como, em quais condições e onde a MA poderá ser adotada.

5. Considerações finais e trabalhos futuros

Desde sua proposta inicial, o CollabProg tem evoluído constantemente e sua segunda versão representa um avanço significativo em termos de estrutura e usabilidade, com uma organização mais clara e intuitiva, facilitando a navegação e compreensão das MAs disponíveis. A plataforma foi submetida a um estudo exploratório com docentes, utilizando o TAM, confirmando sua utilidade no apoio ao ensino de programação com MAs. Atualmente, a terceira versão do CollabProg está em fase de implementação, com a introdução de novos recursos, como o menu "Registrar Metodologia"para o cadastro de novas metodologias e a função de "Recomendação"de MAs, permitindo que os docentes informem características da turma para receber sugestões personalizadas. Esta versão do CollabProg está disponível *online*[2] para acesso.

[2]https://l1nq.com/drfa3

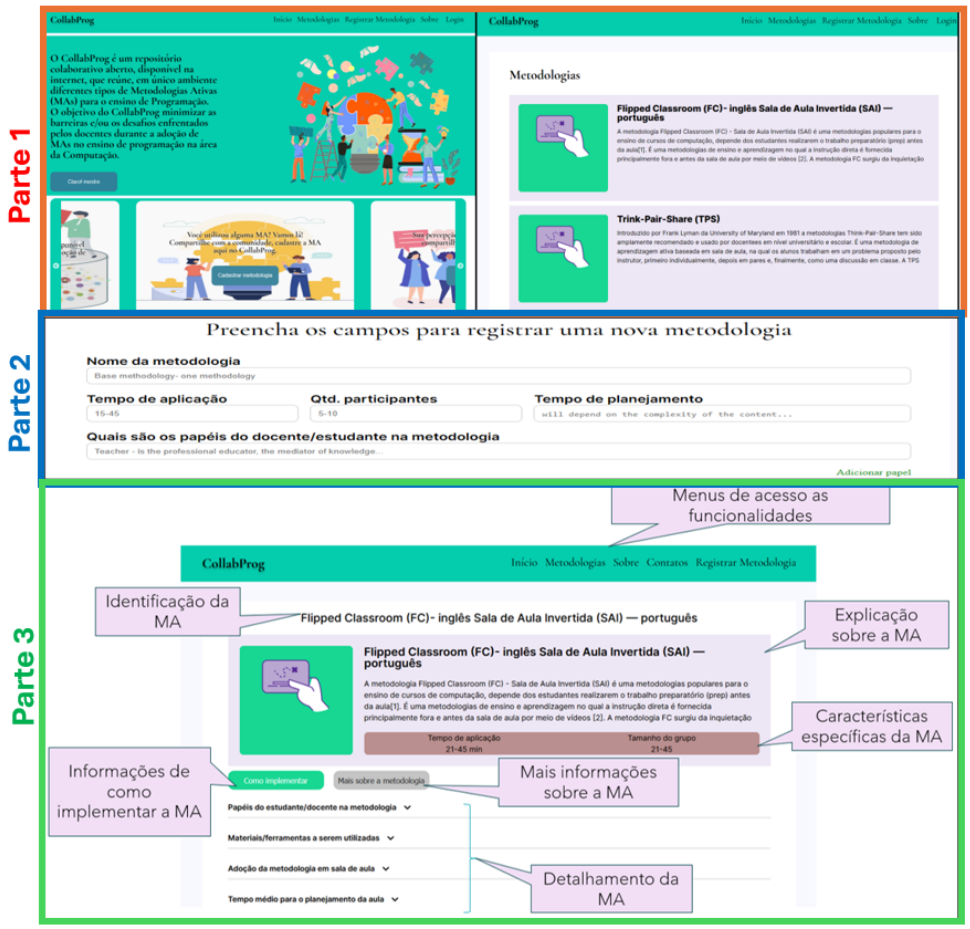


Figura 2. Segunda versão do CollabProg.

Agradecimentos

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Código de Financiamento 001 e parcialmente financiado pela Fundação de Amparo à Pesquisa do Estado do Amazonas – FAPEAM – por meio do projeto POSGRAD. Williamson Silva agradece pelo apoio financeiro da FAPERGS (Projeto ARD/ARC - processo 22/2551-0000606. Ivanilse Calderon agradece ao Instituto Federal de Educação, Ciência e Tecnologia de Rondônia (IFRO)/Campus Porto Velho Zona Norte.

Referências

Ahshan, R. (2021). A framework of implementing strategies for active student engagement in remote/online teaching and learning during the covid-19 pandemic. *Education Sciences*, 11(9):483.

Bigolin, N. M., Silveira, S. R., Bertolini, C., de Almeida, I. C., Geller, M., Parreira, F. J., da Cunha, G. B., and Macedo, R. T. (2020). Metodologias ativas de aprendizagem: um relato de experiência nas disciplinas de programação e estrutura de dados. *Research, Society and Development*, 9(1):e74911648–e74911648.

Cafezeiro, I. L., Viterbo Filho, J., Da Costa, L. C., Salgado, L. C. d. C., Meira, M. R., and Monteiro, R. S. (2017). Grand research challenges in information systems in brazil

2016-2026. *GRAND RESEARCH CHALLENGES IN INFORMATION SYSTEMS IN BRAZIL 2016-2026*.

Calderon, I., Silva, W., and Feitosa, E. (2023). Explorando a aceitação do collabprog como um facilitador de metodologias ativas no ensino de programação. In *Anais do XXXIV Simpósio Brasileiro de Informática na Educação*, pages 93–104. SBC.

de Castro, R. M. and Siqueira, S. (2019). Alcasystem-um portal com técnicas de aprendizagem ativa para disciplinas da área da computaçao. In *Anais dos Workshops do Congresso Brasileiro de Informática na Educação*, volume 8, page 1243.

Eickholt, J. (2018). Barriers to active learning for computer science faculty. *arXiv preprint arXiv:1808.02426*.

Hevner, A. R., March, S. T., Park, J., and Ram, S. (2004). Design science in information systems research. *MIS quarterly*, pages 75–105.

Imbulpitiya, A., Kodagoda, N., Gamage, A., and Suriyawansa, K. (2019). Using active learning integrated with pedagogical aspects to enhance student's learning experience in programming and related concepts. In *International Conference on Interactive Collaborative Learning*, pages 218–228. Springer.

Koenig, K. M. (2020). Personal response systems: Making an informed choice. *Active Learning in College Science: The Case for Evidence-Based Practice*, pages 123–139.

Lima, J., Alencar, F., and Santos, W. (2021). A preliminary guide for assertive selection of active methodologies in software engineering education. In *Brazilian Symposium on Software Engineering*, pages 170–179.

Luxton-Reilly, A., Albluwi, I., Becker, B. A., Giannakos, M., Kumar, A. N., Ott, L., Paterson, J., Scott, M. J., Sheard, J., and Szabo, C. (2018). Introductory programming: a systematic literature review. In *Proceedings Companion of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education*, pages 55–106.

Michael, J. (2007). Faculty perceptions about barriers to active learning. *College teaching*, 55(2):42–47.

Ribeiro, I. C., Silva, W., and Feitosa, E. L. (2021). Repositório colaborativo para apoiar a adoção de metodologias ativas no ensino de programação. In *Anais Estendidos do I Simpósio Brasileiro de Educação em Computação*, pages 56–57. SBC.

Silva, P. (2015). Davis' technology acceptance model (tam)(1989). *Information seeking behavior and technology adoption: Theories and trends*, pages 205–219.

Silva, W., Gadelha, B., Steinmacher, I., and Conte, T. (2020). Towards an open repository for teaching software modeling applying active learning strategies. In *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering: Software Engineering Education and Training*, pages 162–172.

Wieringa, R. (2009). Design science as nested problem solving. In *Proceedings of the 4th international conference on design science research in information systems and technology*, pages 1–12.

IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 6 February 2025, accepted 8 March 2025, date of publication 24 March 2025, date of current version 23 April 2025.

Digital Object Identifier 10.1109/ACCESS.2025.3554156

RESEARCH ARTICLE

Building Bridges Instead of Putting Up Walls: An Educational Tool to Facilitate Instructors in Adopting Active Learning Methodologies for Teaching Programming

IVANILSE CALDERON[1], WILLIAMSON SILVA[2], AND EDUARDO FEITOSA[1]
[1]Institute of Computing, Federal University of Amazonas - (UFAM), Manaus 69067-005, Brazil
[2]Software Engineering Department/PPGES, Federal University of Pampa (UNIPAMPA), Alegrete 96460-000, Brazil

Corresponding author: Ivanilse Calderon (ivanilse.calderon@icomp.ufam.edu.br)

This work was supported in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior–Brazil (CAPES) under Grant 001; in part by the Amazonas State Research Support Foundation (FAPEAM) through the Programa Institucional de Apoio à Pós-Graduação Stricto Sensu (POSGRAD Project), from Fundação de Amparo à Pesquisa do Estado do Rio Grande do Sul (FAPERGS) (Auxílio Recém-Doutor / Auxílio Recém- Contratado - ARD/ARC - Project) under Grant 22/2551-0000606-0 and Grant 23243.004494/2024-20; and in part by the Institutional Qualification Incentive Program (PIQ) under Grant 34/2024/REIT-DGP/IFRO. The Research Group in Technologies and Education in Computing (GPComp) and the Federal Institute of Education, Science, and Technology of Rondônia (IFRO) supported Ivanilse Calderon's work under Grant 23243.004494/2024-20.

ABSTRACT Teaching programming is a challenging task, as it requires instructors to guide students in developing complex skills such as real-world abstraction, problem-solving, and logical reasoning. However, the traditional teaching approach is often ineffective in achieving these objectives. Evidence suggests that Active Learning Methodologies (ALMs) can provide a more conducive environment for skill and competency development. Nonetheless, instructors' adoption rate of ALMs remains relatively low due to various barriers and factors, particularly in programming education. To assist instructors in facing this challenge, we present in this article CollabProg, an open collaborative repository designed to support instructors in identifying and selecting the appropriate ALMs for their teaching context and specific classroom needs. Additionally, CollabProg provides a set of practical guidelines, offering a step-by-step guide to assist instructors in adopting ALMs. We adopted the Design Science Research Methodology (DSRM) to systematically address the research problem and guide the development, evaluation, and evolution of CollabProg. Furthermore, we present two case studies to evaluate the acceptance and feasibility of using CollabProg from the perspective of instructors at different educational institutions in Brazil. The evidence demonstrates that CollabProg effectively supports instructors in adopting active learning methodologies while identifying limitations and opportunities for improvement. We also found that CollabProg helped instructors identify and choose suitable ALMs for their teaching context to meet their specific classroom needs. The guidelines provided by the repository were useful and highly practical for lesson planning in adopting ALMs. The adoption of CollabProg underscores the need for effective strategies to support instructors in teaching programming and motivating students to learn. These strategies are particularly important in collaborative learning contexts, where social interaction is key. CollabProg's versatility in supporting such contexts is important for successful instruction.

INDEX TERMS Teaching programming, active learning methodologies, computer programming, educational tools, CollabProg.

The associate editor coordinating the review of this manuscript and approving it for publication was S. Chandrasekaran.

I. INTRODUCTION

Teaching programming requires a set of skills that students must develop, encompassing not only the understanding of

the syntax and semantics of a specific programming language but also the ability to apply creativity to solve complex problems. This process integrates the precision of logical thinking with the flexibility of creative problem-solving [5] Consequently, teaching and learning programming can be particularly challenging, especially in foundational courses such as CS1, CS2, and Data Structures. Students often perceive these courses as highly complex, demanding a solid grasp of abstract concepts and rigorous analytical thinking. From the instructor's perspective, these courses demand that students develop critical skills, including real-world abstraction, problem-solving, and logical reasoning [2].

Using a traditional teaching approach often proves ineffective for developing essential programming skills. The teacher-centered model, typical of conventional classrooms, frequently results in students passively absorbing information [8]. Consequently, many students either drop out of these courses or leave the program altogether, perceiving the methodology as inadequate for their learning needs [9], [10]. However, this scenario has shifted significantly in recent decades, fueled by continuous technological advancements and the emergence of innovative pedagogical approaches. A notable example of this evolution is the adoption of Active Learning Methodologies (ALMs), which have been widely discussed and implemented in programming education [11]. These methodologies emphasize active student engagement in the learning process, fostering a more dynamic environment that significantly enhances the development of practical skills.

ALMs represent a pedagogical approach integrating active student participation, experiential learning, and learning by doing [3], [4], [31]. By adopting ALMs, students assume a central role in their learning process. This active engagement enhances their interest and deepens their involvement with the content covered in courses such as algorithms [10]. ALMs offer numerous advantages over traditional teaching methods, including personalized learning, access to a wide range of educational resources, immediate feedback, and adaptive learning [32]. However, it is important to emphasize that, despite the benefits of ALMs, traditional teaching methods have their merits and can serve as complementary approaches.

Despite the positive evidence supporting the effectiveness of ALMs in programming education, their adoption rate among instructors remains relatively low [17]. Several barriers have been identified that hinder their widespread implementation [2]. These include the lack of time to design lessons incorporating ALMs, the challenge of covering the entire course content within an ALM framework, student resistance to new pedagogical strategies, doubts about the effectiveness of ALMs in achieving learning objectives, and insufficient guidance on how to implement ALMs effectively in the classroom. Additionally, instructor resistance to change poses a significant barrier, particularly for those already accustomed to traditional teaching methods [2], [17], [28], [81]. These challenges highlight the need for educators to have adequate professional development and resources to facilitate the transition to ALMs. Proactively addressing these barriers can promote the broader adoption of ALMs and enhance the quality of programming education.

This article describes the development, evaluation, and evolution of an open and collaborative repository designed to support the adoption of Active Learning Methodologies (ALMs) in programming education, called **CollabProg**. CollabProg provides educators with a valuable educational tool to implement active methodologies, fostering a more engaging and practical approach to programming education. Additionally, we presented the results of the first Design Cycle, which we conducted to assess the acceptance and feasibility of using CollabProg from the perspective of instructors across various educational institutions in Brazil. The findings from this cycle demonstrate that CollabProg effectively supports instructors in adopting active learning methodologies, a particularly relevant contribution given the growing demand for dynamic and participatory teaching methods. Furthermore, the article highlights the limitations and opportunities for improvement identified in the results, reflecting a commitment to the continuous evolution and enhancement of the tool. This ensures its effectiveness and ongoing relevance in the rapidly evolving landscape of computer education.

The main contribution of our study is (i) **CollabProg**, an educational tool designed to assist instructors in identifying and selecting Active Learning Methodologies (ALMs) that align with their teaching context and classroom needs; (ii) **a structured repository of step-by-step guidelines**, enabling instructors to implement ALMs effectively without the need to search through multiple scientific articles or books; and (iii) **a consolidated collection of strategies**, facilitating the adoption of various ALMs specifically tailored to programming education.

The article is structured as follows. Section II provides foundational information and context for the study. Section III presents literature and prior research relevant to the topic. Section IV introduces the proposed framework, the CollabProg. Section V discusses the significance and practical implications of the study. Section VI outlines the initial design and implementation of the framework. Section VII describes the enhancements and updates based on feedback from the first cycle. Section VIII emphasizes the methodological rigor and validation processes employed. Section IX discusses the limitations of this work. Section X explores the broader impacts of the findings. Finally, Section X concludes our work and discusses some future work.

II. BACKGROUND

This section presents the theoretical foundations of Programming Education in Computer Science and Active Learning Methodologies.

A. TEACHING PROGRAMMING IN COMPUTING

Programming is often regarded as the cornerstone of all computing technologies. Teaching programming has become essential due to the growing significance of computing in everyday life [16]. However, instructors face numerous challenges in this process [2]. Students are required to understand the syntax and semantics of programming languages and apply creativity to solve complex problems, blending the precision of logical thinking with the adaptability of creative problem-solving [18].

At the beginning of their courses, many students struggle with designing and writing simple programs, and some even hesitate to learn programming, perceiving it as a complex subject [6]. A lack of understanding of fundamental concepts further compounds these challenges [7]. Courses on programming logic and computer programming are often perceived as difficult because they require foundational skills in logic, mathematics, and text interpretation [19]. To succeed in these courses, students need prior knowledge in logic, mathematics, reading comprehension, idea abstraction, and other critical skills [20].

These challenges are reflected in the high dropout rates observed in undergraduate Computing programs. Computer Science majors and the increasing number of non-majors who take programming courses often struggle, displaying clear signs of poor performance, frustration, and disengagement. Some institutions have reported dropout rates as high as 50%, while the global average pass rate for introductory Computer Science courses is approximately 68% [24]. Significant efforts have been made to understand why learning to program remains a persistent challenge [24]. This is believed to stem, in part, from current instructional techniques [25], high expectations from instructors [26], and the perceived lack of support for beginner students [21]. In summary, teaching programming courses in higher education is inherently complex due to the diverse skills required for student success [27].

B. ACTIVE LEARNING IN PROGRAMMING EDUCATION

Traditional lecture-based classes, typically instructor-centered, involve students passively listening and absorbing information, often supported by slide presentations. While this approach remains essential in certain contexts, it often represents only a superficial shift in the teaching paradigm, replacing the blackboard with a projector [8]. Technological advancements have primarily altered the means of delivering information to students rather than fostering significant changes in how they learn. However, this instructional method has limitations, as it fails to promote higher-order thinking and advanced reasoning skills [30].

On the other hand, Active Learning (AL), deeply rooted in constructivism [12] offers an approach where students actively construct their knowledge, taking greater responsibility and control over their learning process [13], [14]. Constructivism is a learning theory that posits individuals actively build their knowledge based on their experiences [15]. In this context, AL emphasizes active knowledge construction by the student, empowering them to take ownership of their learning. Rather than passively absorbing information, students learn through practice and experience. This fundamental shift encourages students to take responsibility for their learning journey [37]. In AL, instructors act as facilitators, guiding students to think critically, reflect deeply, and nurture their curiosity [22], [23], [35], [38], [40].

According to [28] AL enables instructors to create learning environments where students construct knowledge, develop critical and reflective skills, and explore personal attitudes and values. AL is a student-centered approach, well-suited for fostering skills in independent study, self-determination, and collaborative work [39]. As noted by [33], [37], [41], [42] there is a growing consensus that humans learn most effectively when they are active (rather than passive) and engaged (rather than distracted), when the material is meaningful (rather than disjointed), and when learning occurs in a socially interactive, iterative (rather than merely repetitive), and enjoyable context.

The literature highlights the advantages of AL in the curricular structure of undergraduate courses [2], [35], [43], [44], [45], [46], demonstrating that active learning strategies are more effective than traditional lectures in promoting a wide range of desirable educational outcomes, such as increased learning [36], [47], [48], [49], [50]. However, despite the favorable evidence, traditional teaching still prevails as the dominant mode of instruction in university courses [36], [51], [52], [53], [54].

C. ACTIVE LEARNING METHODOLOGIES IN PROGRAMMING EDUCATION

Teaching in computing has encountered significant challenges, particularly in balancing extensive theoretical knowledge with learning dynamics that are inherently practical and applied [1]. Adopting innovative pedagogical approaches for teaching programming is essential to develop the critical skills required for students pursuing careers in the field [30]. These skills include abstraction and logical reasoning. Consequently, instructors have increasingly turned to ALM as an effective alternative for programming education [28].

Adopting ALMs for teaching programming has practical and successful implications for instructors aiming to implement AL, as they prepare students for real-world challenges they may encounter in the job market [10]. Consequently, various ALMs are available to facilitate their adoption, helping to mitigate the challenges instructors face when implementing ALMs in programming education.

It is essential to recognize that successfully implementing ALM in programming education is not a haphazard process. It requires a certain degree of knowledge and meticulous planning. Understanding the various ways, whether successful or not, of implementing different strategies for adopting ALM is a key step in this process. This knowledge can be

a solid foundation for instructors seeking to incorporate new ALMs into their programming courses.

As a result, several ALMs have been implemented in programming education in undergraduate computing courses. These methodologies aim to equip students with the skills needed to tackle challenges in the job market, foster greater autonomy in problem-solving, and enhance communication abilities [10].

III. RELATED WORKS

Researchers have explored ways to enhance the adoption of ALM by leveraging new communication and instructional technologies. In the educational context, numerous initiatives have focused on developing digital repositories to support teaching practices across various disciplines.

The ALCASYSTEM portal, developed by [59] is a web-based platform designed to assist instructors in searching for, selecting, and recommending ALM within the context of Computing. It offers a variety of articles exploring different teaching approaches and includes a forum to facilitate interaction among instructors. However, a significant challenge is that instructors must dedicate considerable time to reading and assimilating the recommended articles for each ALM, which can hinder their effective adoption.

In [56], [65], the authors present the OpenSMALS portal, an open repository designed for teaching software modeling through ALMs. OpenSMALS offers specific guidelines on implementing ALM, along with artifacts shared by other instructors, evaluation questionnaires, and additional resources. However, the repository is limited in scope, featuring only eight ALMs, and is focused exclusively on the content area of software modeling.

In [57], the authors developed a selection guide to assist instructors in choosing an ALMs based on identifying students' profiles and learning styles. The authors designed the guide to be practical, easy to use, and adaptable for visualizing and selecting ALMs across different teaching contexts. However, it is limited to Software Engineering and includes only ten ALMs, which are presented in a static digital format that does not facilitate user interaction.

In [58], the authors present a framework to implement activities and strategies that ensure active student engagement during the pandemic. The framework integrates a balanced approach, combining adjusted teaching pedagogy, educational technologies, and an e-learning management system. However, it lacks insights from other instructors' experiences or evaluations regarding the proposed activities and strategies, and its focus is limited to the context of remote teaching.

We recognize the growing demand for specialized support in programming education. However, while existing resources—such as relevant educational materials — offer valuable assistance to instructors, many of these efforts are narrowly focused on specific areas, such as software modeling. Others provide a broad collection of articles for reference, but there remains a notable gap in the availability of tools tailored explicitly for programming education.

Therefore, this article introduces a pioneering contribution: CollabProg. With CollabProg, instructors will no longer need to search through numerous scientific articles or books to find ways to implement specific ALMs in the classroom. Instead, CollabProg consolidates a comprehensive set of strategies for adopting various ALMs tailored to programming education into a single repository. In this context, this article presents an innovative and groundbreaking contribution — CollabProg — which we will explain in detail in the following sections.

IV. CollabProg PROPOSITION

The Design Science Research Methodology (DSRM) [59], [60] guides the development of CollabProg, which helps define the research problem and supports the creation, evaluation, and evolution of the tool. DSRM bridges the gap between knowledge and practice [60] and is widely adopted by researchers for developing educational artifacts [61].

The DSRM also aims to identify and understand real-world problems, propose effective and valuable solutions, and advance the theoretical knowledge of the field [59]. The research for CollabProg began in March 2020 and received approval from the Research Ethics Committee of the Federal University of Amazonas (UFAM) under protocol number 4.694.031. The application of DSRM is structured into three cycles: the Relevance Cycle, the Design Cycle, and the Rigor Cycle.

In the **Relevance Cycle**, we define the problem to be investigated, understand the study context, establish the motivation for addressing the problem, and set the acceptance criteria for the final evaluation of the research outcomes. To achieve this, we first conducted a Systematic Mapping Study (SMS) to summarize the types of ALMs and the experimental evidence related to their adoption in programming education. Based on the findings from this SMS, we defined two types of acceptance criteria for CollabProg: Design Criteria and Behavior Criteria.

The **Design Criteria** specify what CollabProg should offer its users, as detailed below:

- **DC1 –** The artifact should offer a diverse range of ALMs, including detailed descriptions, practical application examples, and relevant usage contexts.
- **DC2 –** The artifact should provide clear and structured guidelines for implementing each ALM, covering key aspects such as planning, execution, and evaluation.
- **DC3 –** The artifact should include curated ALMs (a process of carefully selecting and organizing ALM content) featuring critical analyses, evidence-based recommendations, and feedback from instructors who have previously implemented these methodologies.

The **Behavior Criteria** are related to contributing to the teaching practices of programming and are as follows:

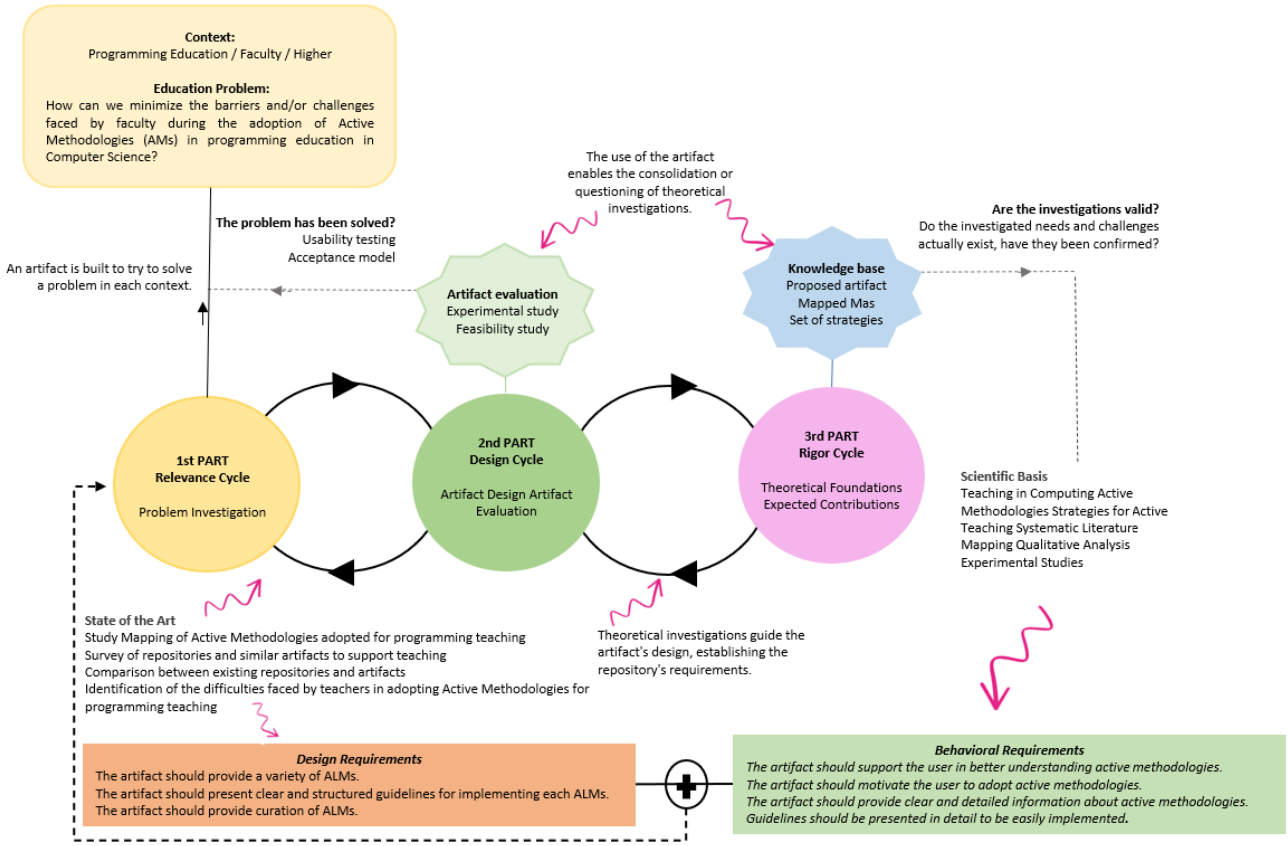


FIGURE 1. Research methodology. Adapted from [63].

- **BC1 –** The artifact should enhance the user's understanding of ALMs by providing educational resources such as tutorials, case studies, and explanatory videos.
- **BC2 –** The artifact should encourage instructors to adopt ALMs by presenting evidence of their effectiveness, observed benefits in other institutions, and success stories.
- **BC3 –** The artifact should offer clear and detailed information about ALMs, including pedagogical objectives, step-by-step implementation instructions, and potential challenges with suggested solutions.
- **BC4 –** The artifact should provide practical and detailed guidelines, using accessible language and concrete examples to facilitate implementation across various disciplines and educational levels.

In the **Design Cycle**, we develop, evaluate, and refine the artifact, CollabProg. This development is primarily grounded in ALMs adopted for teaching programming. To assist instructors in selecting appropriate ALMs, we identified and curated ALMs from the SMS, as detailed in Section V. CollabProg was evaluated by applying it to specific problems and contexts, enabling us to assess whether we achieved the expected outcomes based on requirements and whether further iterations of the Design Cycle were necessary. Consequently, we evaluated CollabProg in practice—specifically, with instructors in real learning contexts—as discussed in Section VI-E7.

Finally, the **Rigor Cycle** focuses on the generation and application of knowledge [59]. In this cycle, the primary foundations include knowledge about adopting ALMs for teaching programming in Computing, strategies for implementing ALMs, the SMS, experimental studies, qualitative and quantitative analyses, focus groups, interviews, and other relevant methods. Regarding knowledge generation, the main contribution is CollabProg, a comprehensive set of strategies for adopting AMs from the perspective of programming instructors.

The following subsections will provide a detailed explanation of the DSRM cycles.

V. RELEVANCE CYCLE

In DSRM, a practical problem drives the investigation, generating new research questions and challenges that expand existing knowledge [59]. According to [60] the initial phase of the research focuses on understanding the problem without proposing immediate solutions.

To gain a deeper understanding of the research problem, we first conducted a SMS. The goal was to identify how instructors utilize ALMs in programming education. According to [62] an SMS systematically categorizes and

summarizes existing information on a research question unbiasedly. As suggested by [62] this process involves planning, conducting, and analyzing results. For further details on the SMS, refer to [29].

The Research Question guiding the SMS was: **How have instructors used active learning methodologies while teaching programming in undergraduate courses?** From this question, we identified 3,850 publications through a meticulous search process. After rigorously applying selection criteria, we accepted 81 publications.

From the 81 publications analyzed in the SMS, we identified 37 types of ALMs. Among these, 17 publications mentioned using Mixed Methodologies, such as Flipped Classroom and Problem-based Learning. The Flipped Classroom (FC) was cited in 14 publications, while Gamification-based Learning (GM) appeared in 11. Eight publications employed Problem-Based Learning (PBL), while five used Game-Based Learning (GBL). Authors in four publications also developed their own ALMs, and four others applied the Project-Based Learning (PjBL) methodology. Other identified methodologies include:

- Cooperative Learning (CL) and Pair Programming (PP), each cited in three publications;
- Team-based Learning (TBL), Think-Pair-Share (TPS), and Coding Dojo (Dojo) in two publications;
- Blended Learning (BL), Peer Review (PR), Project-based Service-learning (PBSL), Method 300 (M300), Process Oriented Guided Inquiry Learning (POGIL), and Topdown Approach (TopD), each in one publication.

We observed that researchers applied ALMs in over 30 different subjects. Introductory Programming was the most frequently mentioned subject associated with ALMs such as PBSL, PJBL, PP, TBL, and TPS. Algorithms and Data Structures were addressed using GM, GBL, FC, M300, and DOJO methodologies. Computer Programming was frequently linked to ALMs such as BL, FC, and mixed methodologies. Other subjects, including Parallel Programming, Object-Oriented Programming, System Programming, Software Design, Teaching Programming, and Programming Paradigms, were associated with methodologies like CL, PBL, and PR.

Additionally, we found that Java is the most frequently used programming language, mentioned in 27 publications, while C++, Python, and C appeared in 12, 11, and 10 publications, respectively. Notably, several publications did not specify the programming language adopted.

The results of the SMS underscored the significance of using ALMs in programming education, demonstrating that these methodologies engage students in ways that foster active, participatory, and contextualized learning. The studies revealed that ALMs encourage students to apply their knowledge to solve real-world problems, developing practical and critical skills essential for success in programming. The diversity of subjects addressed through various ALMs reflects the interdisciplinary nature of programming, preparing students to tackle complex and diverse challenges. Consequently, adopting ALMs enhances the student learning experience and empowers instructors to deliver effective and relevant education that aligns with market demands and the needs of contemporary society.

VI. FIRST DESIGN CYCLE: CollabProg 1.0

This section discusses CollabProg's conception, evaluation, and iterative improvement process. We developed CollabProg to address instructors' practical challenges when adopting ALMs in programming education by providing specific guidelines for implementing various ALMs. It is a collaborative and open repository to support instructors in integrating ALMs into their programming teaching practices.

A. ORGANIZATION OF KNOWLEDGE ABOUT THE METHODOLOGIES

The results obtained through the SMS enabled us to identify and categorize the ALMs being adopted by instructors. Additionally, they revealed significant positive evidence regarding their effectiveness in programming education. After identifying these ALMs, we drew inspiration from those proposed by [64] and [65] and organized the knowledge about each methodology into a conceptual model represented as a class diagram. To achieve this, we first defined the domain and scope of the knowledge to be constructed, based on the results obtained in [29].

According to [64], the domain refers to the semantic representation and formalization of teaching methodologies grounded in active learning principles. The scope of this model is to assist instructors in teaching programming within higher education by organizing and semantically representing knowledge, thereby facilitating the dissemination and adoption of ALM. Based on this, we structured the information gathered from the ALMs into a conceptual model, represented using the class diagram shown in Figure 2.

In the model, the **Category** class represents the classification of ALM based on their approach. This class is associated with the **Methodology** class, representing the ALM included in CollabProg. As observed in the SMS, the authors combine methodologies to enhance or complement the positive outcomes of programming education. A self-relationship within the Methodology class represents this possibility. The **Step** class outlines the necessary steps for adopting these methodologies. The **Activity** class describes the activities to be performed while implementing methodologies in the classroom, such as content planning, methodology explanation, and role assignment, among others. The **Technology** class represents the educational technologies that can be utilized for each activity, whether virtual environments, games, or other tools. Finally, to define the roles involved in the methodologies, the **Participant** and **Role** classes are associated with each other and linked to the Methodology class.

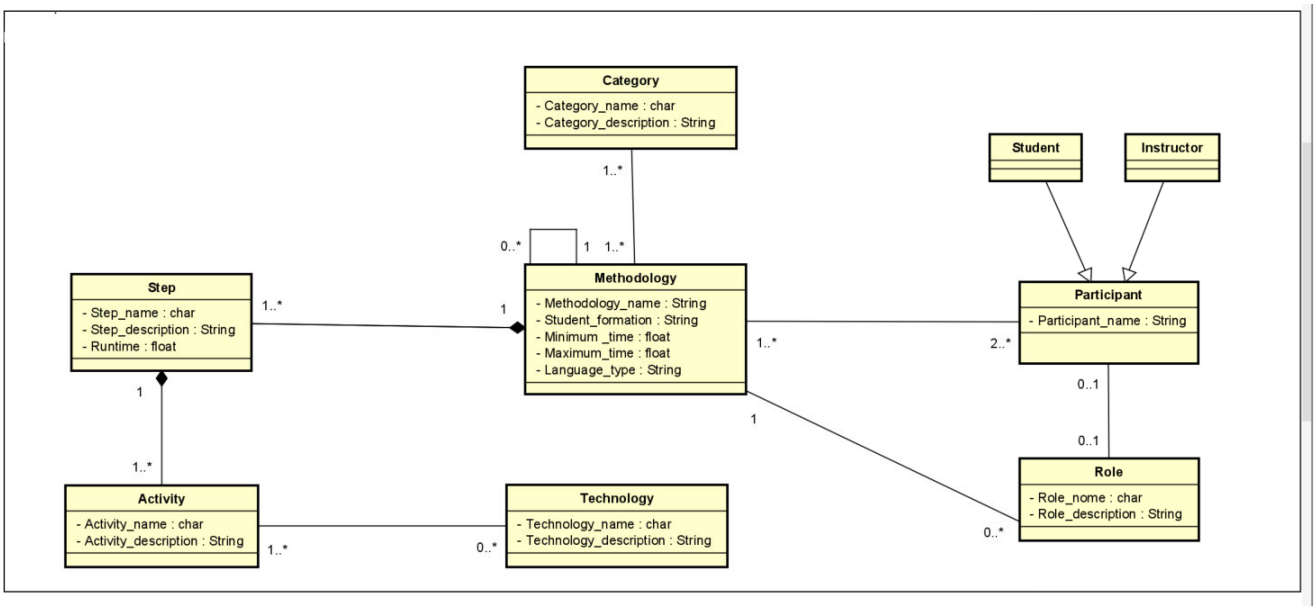


FIGURE 2. Conceptual model of CollabProg.

B. SELECTION AND CURATION PROCESS OF ACTIVE LEARNING METHODOLOGIES

After organizing the knowledge about ALMs, we thoroughly curated the information used to compose CollabProg. This involved examining scientific evidence and experimental studies demonstrating the practical application of ALMs in the classroom. It is important to note that the studies analyzed were those selected in the SMS. Content curation plays a critical role, as instructors often struggle to determine the origin of information, which can compromise the assessment of its accuracy and authenticity [66]. Therefore, we aimed to prevent potential frustrations for the end users of the repository—primarily instructors—by ensuring that the available content is scientifically grounded, based on empirical evidence, and relevant to its intended purpose [66].

The curation process primarily focused on studies that provided detailed analyses to assist instructors in effectively implementing these methodologies in the classroom, particularly in programming education. We selected ALMs supported by solid scientific evidence, excluding those lacking experimental validation or irrelevant theoretical foundations. As a result, CollabProg aims to share strategies for adopting ALMs and commits to ensuring the quality and relevance of the available knowledge. This ensures that the teaching community can access valuable, well-founded resources to enhance their pedagogical practices in computer programming education.

We established a set of Quality Assessment Criteria (QAC) to evaluate the quality of the primary studies selected in the SMS for developing the CollabProg repository. The QACs are designed to analyze studies on adopting ALMs for programming education in higher education, particularly in Computing. These criteria quantify the relevance of each primary study, guiding the selection of content to be included in CollabProg.

Table 1 presents the specification of the QAC and the corresponding scores each ALM can receive. We established six QACs to extract detailed information from primary studies that support instructors in implementing the methodologies in the classroom. These criteria guide the search for detailed information in the studies and classify them as **Strong**, **Medium**, or **Weak**. The scoring scheme for publications is as follows: **Strong Description = 2, Medium Description = 1, and Weak Description = 0**.

Criteria that require a strong rating directly impact the implementation and understanding of ALMs in the classroom (QAC1 and QAC2). Studies that do not achieve the maximum score in these criteria are excluded. For criteria that can receive a weak rating, the absence of information in primary studies will not hinder CollabProg users' adoption of the methodology. The detailed protocol used for primary studies' Quality Assessment (QA) is available online.[1] These criteria are as follows:

- **QAC1 – Description of ALMs**. This criterion must be strong. Studies should provide detailed information about the ALM and its benefits, enabling CollabProg users to understand better the methodology they intend to adopt.
- **QAC2 – Adoption Support**. This criterion must be strong, as studies should offer clear and practical guidance on the steps required for CollabProg users to implement and adopt the methodology in their

[1] https://figshare.com/s/794c9f7e5adfdff915d1

TABLE 1. Quality assessment criteria X publication score.

Criteria	Description of the criterion	Score
QAC1	**Strong Description:** If the methodology is clearly identified and described comprehensively, with information related to its concept, origin, objective, characterization, application, and the benefits of implementation in the classroom.	2
	Medium Description: If the methodology description is partially or incompletely described, with not all information related to its concept, origin, objective, characterization, application, and the benefits of implementation in the classroom being provided.	1
	Weak Description: If no descriptions related to the concept, origin, objective, characterization, application, and the benefits of implementation in the classroom are mentioned.	0
QAC2	**Strong Description:** If it describes in detail the steps for implementing the methodology in the classroom, clearly presents the step-by-step process to be followed for adopting the methodology, and provides relevant information about the tools and/or technologies used during the adoption of the methodology.	2
	Medium Description: If it describes incompletely and with few details the steps for implementing the methodology in the classroom, presents the step-by-step process for adopting the methodology incompletely, and provides incomplete information about the tools and/or technologies used during the adoption of the methodology.	1
	Weak Description: If it does not describe the steps for implementing the methodology in the classroom, the step-by-step process for adopting the methodology, and information about the tools and/or technologies used during the adoption of the methodology.	0
QAC3	**Strong Description:** If it clearly describes and specifies which metrics were used to evaluate the improvement in programming teaching.	2
	Medium Description: If it describes incompletely which metrics were used by instructors to evaluate the improvement in programming teaching.	1
	Weak Description: If it does not describe or specify which metrics were used to evaluate the improvement in programming teaching.	0
QAC4	**Strong Description:** If it describes information that allows identifying and characterizing the type of programming language used.	2
	Weak Description: If no relevant information allowing the identification of the programming language used is mentioned.	0
QAC5	**Strong Description:** If it presents complete, clear, and relevant information about the teaching modality where the methodology was implemented.	2
	Weak Description: If it does not present the teaching modality where the methodology was implemented.	0
QAC6	**Strong Description:** If it presents a clear description of the results obtained with the adoption of the methodology during the teaching of the content, lessons learned, positive or negative points, from the instructor's perspective.	2
	Medium Description: If it presents incomplete information about the results and lessons learned by adopting the methodology during teaching programming.	1
	Weak Description: If it does not present the results achieved or the lessons learned by adopting the methodology used for teaching programming.	0

classrooms. This approach will instill confidence and capability in the users.

- **QAC3 – Metrics**. This criterion can be weak, as it seeks studies that present the metrics used to evaluate the methodology's effectiveness in improving teaching and learning.
- **QAC4 – Programming Language**. This criterion can be weak, as it aims to identify the programming language used during the methodology's implementation.
- **QAC5 – Teaching Modality**. This criterion can be weak, as it identifies the teaching modality (face-to-face, blended learning, or distance education) in which the methodology was implemented.
- **QAC6 – Results Description**. This criterion must be strong, as it seeks solid empirical evidence on the outcomes of implementing the ALMs. This emphasis on empirical evidence ensures the validity and reliability of the studies.

We evaluated each study on a scale from 0 to 2, following the description provided in Table 2. Based on the scores, we classified the studies and excluded those not meeting the required criteria. We removed publications with a score of 0, even if they fell within the research domain. We present in Table 2 all selected ALMs included in CollabProg.

It is important to note that the entire evaluation and curation process was conducted by three researchers, ensuring a rigorous and thorough analysis. All three researchers consistently reviewed and evaluated each stage of the process, making decisions collaboratively and based on solid evidence. This collaborative approach enhanced the accuracy and reliability of the study selection, ensuring that only the most relevant and detailed publications were included in the CollabProg repository.

TABLE 2. Selected active learning methodologies for CollabProg compositiong.

Methodology Name	Authors
Blended Learning (BL)	[69]
Cooperative Learning (CL)	[70]
Flipped Classroom (FC)	[71]
Game-Based Learning (GBL)	[72]
Gamification-Based Learning	[73]
Method 300 (M300)	[74]
Problem-Based Learning (PBL)	[75]
Project-Based Learning (PjBL)	[76]
Peer Review (PR)	[77]
Team Based Learning (TBL)	[78]
Topdown (TopD)	[79]
Think-Pair-Share (TPS)	[80]
Coding Dojo (DOJO)	[81]

C. CollabProg - VERSION 1.0

After completing the selection and curation process of ALMs, we structured the collected information and developed the first version of CollabProg. Figure 3 illustrates the initial version of CollabProg, focusing on a specific active methodology, POGIL. Part 01 of Figure 3 provides a brief overview of CollabProg, while Part 02 offers a concise description of the selected ALM, in this case, POGIL. Finally, Part 03 provides a detailed explanation of the methodology,

FIGURE 3. First version of CollabProg.

including its roles, implementation steps, and a breakdown of each step.

In CollabProg version 1.0, we organized the repository into three labeled menus, each designed to help users navigate, select, and adopt any available ALMs. Instructors can access a wide range of information on ALMs within CollabProg, including adoption examples, community-recommended tools, real-world experiences, and feedback from other instructors. This platform provides valuable insights into the strengths and limitations of different ALMs. A key feature is that users do not need to register to access CollabProg—it is open to everyone. Within the main interface (Home), instructors can access the following menus:

- **About**: Provides an overview of the CollabProg repository.
- **Methodology**: Lists the ALMs mapped from the SMS results.
- **Recommendation**: This feature allows instructors to input characteristics about their class, such as the content to be taught, the discipline, and other relevant details, enabling CollabProg to recommend the most suitable ALM for the scenario. The recommendation includes step-by-step instructions for implementing the ALM, details on roles during its execution, activity suggestions, and options for community-recommended tools to support the process.
- **Register methodology**: This feature invites instructors to actively contribute to the CollabProg repository by sharing a new ALM or an adaptation of one they have implemented or tested for teaching programming. This collaborative space fosters community and shared learning among instructors and researchers.
- **Contact**: This feature is a communication channel between the researchers developing the platform and the

academic community. Users can contact the authors via email to report errors and issues or provide suggestions for improving the repository.

We implemented CollabProg across three layers (back-end, front-end, and recommendation system), with the participation of six students dedicated to the development process. The back end, responsible for managing business rules and data, provides an API for interaction with the front end, enabling the registration and retrieval of information necessary to populate CollabProg's screens and methodology recommendations. It is important to note that the recommendation system has not yet been fully implemented and remains a pending aspect of the project. When selecting technologies, tools, and programming languages, we prioritized those that facilitate development, task organization, and the completion of development stages, such as:

- **Node.js**: A JavaScript runtime environment for building scalable and asynchronous server-side applications.
- **Mongoose**: An Object Data Modeling (ODM) library for MongoDB and Node.js, facilitating integration and management of data relationships.
- **NestJS**: A back-end framework for Node.js offering efficiency and scalability, using TypeScript and a syntax similar to Angular.

We chose a non-relational database for the database due to the self-contained nature of the methodology and user experience documents. We selected MongoDB for its efficient storage capacity, schema flexibility, and ease of implementation and maintenance. We also chose development tools based on their ability to support the development process and facilitate the visualization of changes made by developers, including:

- **Docker**: Used throughout the development lifecycle to enable quick, easy, and portable application development, both in desktop and cloud environments.
- **GitHub**: A platform that hosts the application code, offering Git version control.
- **Visual Studio Code**: A lightweight yet powerful code editor available for Windows, macOS, and Linux, with integrated support for JavaScript, TypeScript, and Node.js.
- **Swagger UI**: Allows visualization and interaction with API resources without logic implementation, with automatic generation from the OpenAPI specification, simplifying back-end development and client-side consumption.

CollabProg's back end was hosted on a free platform to enable quick access to the front end, which was integrated using Railway. We chose Railway, a cloud platform that simplifies software deployment complexity, for its ease of application deployment. The tools used for front-end development were as follows:

- **React**: A JavaScript library used to create component-based user interfaces, known for its ease of use, flexibility, and scalability, selected for its declarative nature, component-based approach, and ability to be server-rendered using Node.
- **TypeScript**: A superset of JavaScript developed by Microsoft in 2012, incorporating features and tools lacking natively in the JavaScript language, chosen for the ease of performing activities through ''transcompilation'' of code into ''pure'' JavaScript before execution.

D. PRACTICAL USE OF CollabProg

The scenario described below illustrates how CollabProg can assist instructors in implementing ALMs —in this case, POGIL—in programming education, specifically for teaching conditional structures (if, if-else, and else). The goal is to provide a practical example demonstrating how the CollabProg repository facilitates lesson planning and execution.

Scenario Context: The instructor is teaching a *Programming I* course to a first-semester class in a Computer Science program. The lesson focuses on conditional structures, a foundational concept in programming. To address the student's learning needs, the instructor decided to adopt the POGIL methodology, supported by CollabProg.

1) OVERVIEW OF THE METHODOLOGY IN CollabProg

CollabProg provides a centralized repository that equips instructors with tools and resources to plan lessons effectively:

- **Detailed descriptions of POGIL:** The repository provides structured templates to guide instructors in implementing collaborative and inquiry-driven activities tailored to programming education.
- **Planning guides:** The repository provides structured templates to guide instructors in implementing collaborative and inquiry-driven activities tailored to programming education.
- **Interactive tools:** CollabProg allows instructors to design problems divided into progressive stages, using prompts that stimulate inquiry and promote the practical application of concepts.

In the scenario, the instructor utilized a POGIL template from CollabProg to design a practical lesson where students:

- **Explore** the fundamentals of conditional structures through guided questions and example analysis.
- **Conceptualize** the material by writing small code snippets.
- **Apply** their knowledge by solving a broader programming problem.

2) PRACTICAL APPLICATION OF POGIL IN THE CLASSROOM

Using CollabProg's step-by-step guide, the instructor structures the lesson as follows:

Step 1: In the classroom, the lesson is conducted in a structured sequence designed to maximize student engagement and collaboration. The instructor organizes students into

small groups, assigning specific roles to each member (e.g., facilitator, recorder, questioner, quality monitor) to ensure active participation from everyone.

Step 2: The instructor introduces a practical problem, breaking it into manageable steps. Each group collaborates to explore concepts, answer questions, and write code snippets guided by prompts provided in the template.

Step 3: As students work, the instructor observes group interactions, monitors progress, and provides clarification or redirection when necessary, fostering a collaborative learning environment.

Step 4: At the end of the session, each group presents its solutions, allowing the class to compare approaches and discuss alternative strategies. CollabProg assists in tailoring assessments to the classroom context, helping instructors choose methods that align with students' needs and characteristics.

In this case, using CollabProg, instructors can access a step-by-step guide for implementing the POGIL methodology in the classroom. The platform also assists in determining the most suitable type of assessment for the classroom context, considering the student's needs and characteristics. CollabProg provides clear guidelines, enabling instructors to follow structured recommendations and achieve optimal results in their teaching practice. This approach fosters a deeper understanding of the content, ensuring a dynamic, interactive, and reflective learning experience for all students.

3) OVERVIEW OF APPLYING THE POGIL METHODOLOGY WITH THE USE OF CollabProg

This example demonstrates how CollabProg serves as both a repository of ALMs and a practical tool for:

- **Streamlining Lesson Planning**: Ready-to-use, customizable templates reduce planning time for instructors.
- **Enhancing Student Engagement**: Collaborative activities increase interaction and participation.
- **Fostering Skill Development**: Students develop technical and process skills such as communication, leadership, and problem-solving.

The POGIL methodology emphasizes collaboration and inquiry. Instructors can implement this methodology effectively in their classrooms with a structured approach. CollabProg provides:

- **Step-by-Step Guide:** Instructions on organizing groups, assigning roles, and structuring activities around inquiry-based problems.
- **Customization:** Tools to adapt POGIL activities to meet the specific needs of beginner or advanced learners.
- **Assessment Tools:** Methods for formative and summative evaluations and timely, personalized feedback to enhance the learning experience.

This scenario highlights how CollabProg supports the adoption of ALMs, from planning to execution, by offering a concrete example of its application in programming education. It emphasizes the platform's positive impact on lesson organization, student engagement, and the practical implementation of methodologies such as POGIL.

E. EVALUATING CollabProg 1.0

In this sense, we conducted an exploratory study to verify the feasibility of using and accepting CollabProg 1.0. In the DSRM cycle, it is essential that stakeholders directly involved in the problem's context evaluate the artifact [60]. In this regard, we conducted an exploratory study to assess the feasibility and acceptance of CollabProg 1.0.

1) PLANNING

This study aims to evaluate the feasibility and acceptance of CollabProg from the instructors' perspective. We recruited instructors from various locations across Brazil using a convenience sampling approach. Due to the geographical distance between participants, we adapted the study artifacts using online tools available via Google Workspace, including:

- (i) a consent form ensuring the confidentiality of provided data and the anonymity of instructors (Ethics Committee No. 4,694,031)[2];
- (ii) a characterization questionnaire to understand instructors' experience in the classroom and with the use of ALMs[3];
- (iii) documents containing the study script, CollabProg usage instructions, and online rooms for conducting experiments[4];
- (iv) the initial version of the CollabProg web portal[5];
- (v) a lesson plan template[6]; and
- (vi) a post-use form based on Technology Acceptance Model (TAM) indicators.[7]

2) PARTICIPANTS

We recruited five instructors from higher education institutions voluntarily participating in the study. Table 3 provides an overview of the instructors' profiles.

Only two participants had used ALMs in the classroom (PP—Pair Programming and PBL—Problem-Based Learning), highlighting the underutilization of ALMs' potential in programming education. Regarding their motivation for adopting ALMs, the instructors reported that it provided students with greater autonomy, facilitated the learning process—since programming requires significant reasoning and abstraction—and empowered students to take ownership of their learning. None of the participants had used educational tools to assist them in implementing ALMs in the classroom.

[2]https://bit.ly/3zelXpx
[3]https://bit.ly/4eDjWDl
[4]https://bit.ly/3VB4VcB
[5]https://bit.ly/3VX4yun
[6]https://bit.ly/4bnH3yW
[7]https://bit.ly/3VXtmT4

TABLE 3. Summary of participants' profiles in the study.

ID	University	Courses	Experience	Uses ALM	Period	ALM
I1	Federal University of Pampa (UNIPAMPA)	Computer Science and Software Engineering	1 year	No	-	-
I2	Federal Institute of Amazonas (IFAM)	Computer Science and Informatics	3 years	Yes	3 months	PP
I3	Federal Technological University of Paraná (UTFPR)	Software Engineering and Computer Science	5 years	No	-	-
I4	State University of Maringa (UEM)	Computer Science and Engineering	8 years	No	-	-
I5	Institute of Federal of Rondônia (IFRO)	Computer Networks and Internet Systems	10 years	Yes	5 years	PBL

[8]PP - Pair Programming; PBL - Problem-Based Learning

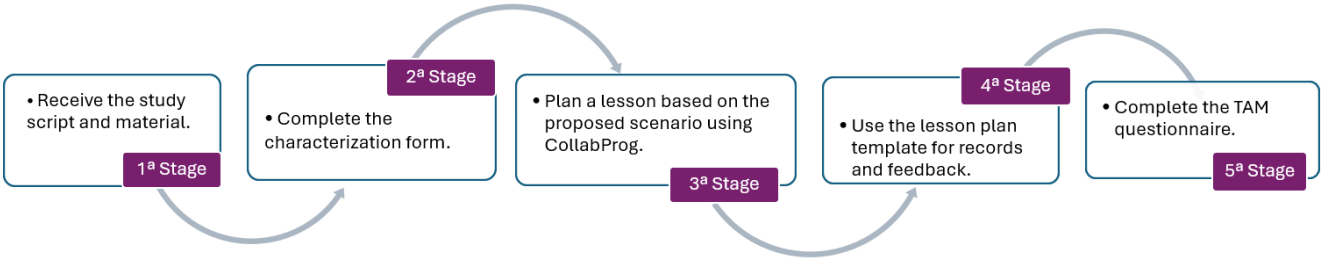


FIGURE 4. Stage of the study to evaluate CollabProg.

3) EXECUTION

We conducted the study entirely online and on an individual basis. Before the first DSRM cycle, we performed a pilot study to verify whether the study would achieve its objectives. The pilot results were satisfactory, and we concluded that no improvements to the study script were necessary. We invited each instructor via email, describing the study's goals and some guiding instructions. If they agreed to participate, one of the researchers scheduled a date for the individual session. Upon acceptance, we conducted the study following the detailed steps outlined in Figure 4. We explain each stage below.

On the scheduled date, we sent the instructor a link to a document containing the preparation script via online chat. This document included the online consent form and a characterization form with questions about the instructor's experience adopting ALMs for programming education. Participation in the CollabProg evaluation was voluntary, and all participants signed the consent form, agreeing to participate in the study and allowing their results to be used for analysis. After completing the questionnaires, the instructors received instructions and explanations about the study. The script required the instructors to plan a class using an ALM to teach the topic ''Variables and Constants'' from a typical Programming I course.

For this task, we provided the instructors with **(a)** a lesson plan template to complete and **(b)** the online version of CollabProg, which they were instructed to use as a support tool for creating the lesson plan by following the guidelines and recommendations available in the repository. At the end of the task, the instructors submitted their completed lesson plans. The focus was not on evaluating the correctness of the plan but on determining whether CollabProg assisted the instructor in planning the methodology across all stages of the class. We emphasize that the instructors should choose the best methodologies that are aligned with their theoretical and practical knowledge, skills, and teaching context. After completing the planning, we invited the instructors to answer an evaluation questionnaire in which they shared their experience using CollabProg.

4) DATA ANALYSIS

The evaluation questionnaire for CollabProg was designed based on the indicators of the Technology Acceptance Model (TAM) [67]. TAM is a framework used to gather information about participants' perceptions regarding the key factors influencing the acceptance or rejection of a particular technology. The indicators defined were [67]: **(i)** Perceived Usefulness, which measures the extent to which instructors believe CollabProg can enhance their performance in adopting ALMs; **(ii)** Perceived Ease of Use, which measures the extent to which instructors believe using CollabProg would be effortless; and **(iii)** Perceived Intention to Use measures the extent to which instructors believe they will use CollabProg in the future. We focused on these indicators because they strongly correlate with instructors' acceptance of CollabProg.

Using CollabProg, instructors provided their perceptions based on their level of agreement with the statements established in the TAM. The instructors carefully evaluated each statement using a five-point Likert scale, ranging from Strongly Disagree to Strongly Agree, ensuring a systematic and comprehensive approach to the research. Table 4 presents the statements answered by the instructors aligned with the TAM indicators. Additionally, we included two open-ended questions to gain deeper insights into the instructors' responses. We conducted a qualitative analysis using coding techniques based on the responses received.

TABLE 4. Questions to be answered by the instructors.

Perceived Usefulness	
PU1	Using the CollabProg repository improved my performance in lesson planning by adopting ALMs.
PU2	Using the CollabProg repository improved my productivity in adopting ALMs.
PU3	Using the CollabProg repository allowed me to fully report the aspects of my experience in adopting active methodologies (ALMs).
PU4	I find the CollabProg repository useful for reporting my experience in adopting active methodologies (ALMs).
Perceived Ease of Use	
PEU1	The CollabProg repository was clear and easy to understand
PEU2	Using the CollabProg repository did not require much mental effort
PEU3	I think the CollabProg repository is easy to use.
PEU4	I find it easy to report my experience of adopting MAs using the CollabProg repository.
Perceived Intention to Use	
PIU1	Assuming I have access to the ColabProg repository, I intend to use it to apply AIs in programming education.
PIU2	Given that I have access to the ColabProg repository, I foresee using it to support me in adopting AIs in programming education.
PIU3	I intend to use the ColabProg repository to assess my experience with adopting an AI in the next month.
Open-Ended Questions	
OQ1	What were the main challenges/negative points perceived by you when using ColabProg?
OQ2	What were the main positive aspects you noticed when using ColabProg?

5) ANALYSIS OF INSTRUCTORS' LESSON PLANS

In the lesson plans evaluated by instructors who used CollabProg, the importance of organized lesson planning for the success of ALMs and effective teaching was evident. The platform provides a clear structure that enables instructors to systematically organize their lessons, following a logical sequence of steps, simplifying the creation of well-structured plans. Most instructors participating in the evaluation adopted the TPS methodology, followed by POGIL, which is highlighted by the platform as effective for fostering collaborative and student-centered learning. When properly implemented, these lesson plans demonstrate that ALMs can positively impact the teaching process.

Regarding content selection, instructors were able to identify relevant and appropriate topics tailored to their students' learning levels, aligning lesson objectives with the specific content. This clarity in defining content and purpose reflects the instructors' thoughtful approach to effective teaching. By leveraging CollabProg, instructors strategically selected relevant and suitable content for their students' stage of learning. As a result, the planning process became more focused and directed toward the most essential elements, leading to greater clarity and objectivity.

When defining lesson objectives, instructors established clear goals and ensured alignment with the selected ALMs. This deliberate reflection on objectives was key in delivering lessons more effectively. By employing methodologies such as TPS or POGIL, instructors created active learning environments that fostered greater student engagement and participation. As a result, the lessons moved beyond mere knowledge transmission, emphasizing learning through discussions, problem-solving, and collaborative work—core principles of the adopted methodologies.

Additionally, the lesson plans included a detailed analysis of the required resources, demonstrating that instructors were well-prepared to address the challenges of teaching. Using CollabProg to organize these resources provided a broader and more practical perspective on lesson execution. The instructors followed the step-by-step methodology offered by the platform, enabling them to structure activities effectively and allocate appropriate time for each phase, from group discussions to hands-on activities. Assessment strategies were also carefully planned, allowing instructors to measure student learning throughout the lesson and ensure continuous and effective monitoring of student progress. In this way, CollabProg supported the organization of lesson planning and encouraged instructors to reflect deeply on the best approaches to teaching and assessment, ultimately enhancing educational outcomes.

6) QUANTITATIVE RESULTS OF THE TAM QUESTIONNAIRE

Figure 5 presents the overall results of the participants' perceptions of CollabProg, based on the TAM statements outlined in Table 4. These results provide insights into the instructors' experiences regarding the platform's usefulness, ease of use, and intention to use the repository.

Regarding the instructors' perceptions of CollabProg's Perceived Usefulness, we observed that in all statements (PU1, PU2, PU3, PU4), all instructors strongly agreed that CollabProg helps plan programming classes using ALMs. Furthermore, CollabProg is a tool capable of enhancing or supporting instructors' productivity in their teaching practice. CollabProg is a support tool that enables instructors to use their experiences to select an ALM to implement in their classes. Finally, the results reflect the instructors' acceptance of CollabProg as a valuable resource for adopting ALMs in programming education.

The three statements (PEU1, PEU2, and PEU3) regarding the perceived ease of using CollabProg received strong overall agreement. Instructors said reporting their experiences of adopting ALMs using CollabProg is easy. They also noted that using the platform required minimal mental effort and was easy to understand and use, particularly for meeting the daily needs of programming teaching practice. Overall, all instructors found CollabProg straightforward, easy to understand, and simple. The only exception was in statement PEU2, where D2 partially agreed regarding the ease of use of CollabProg.

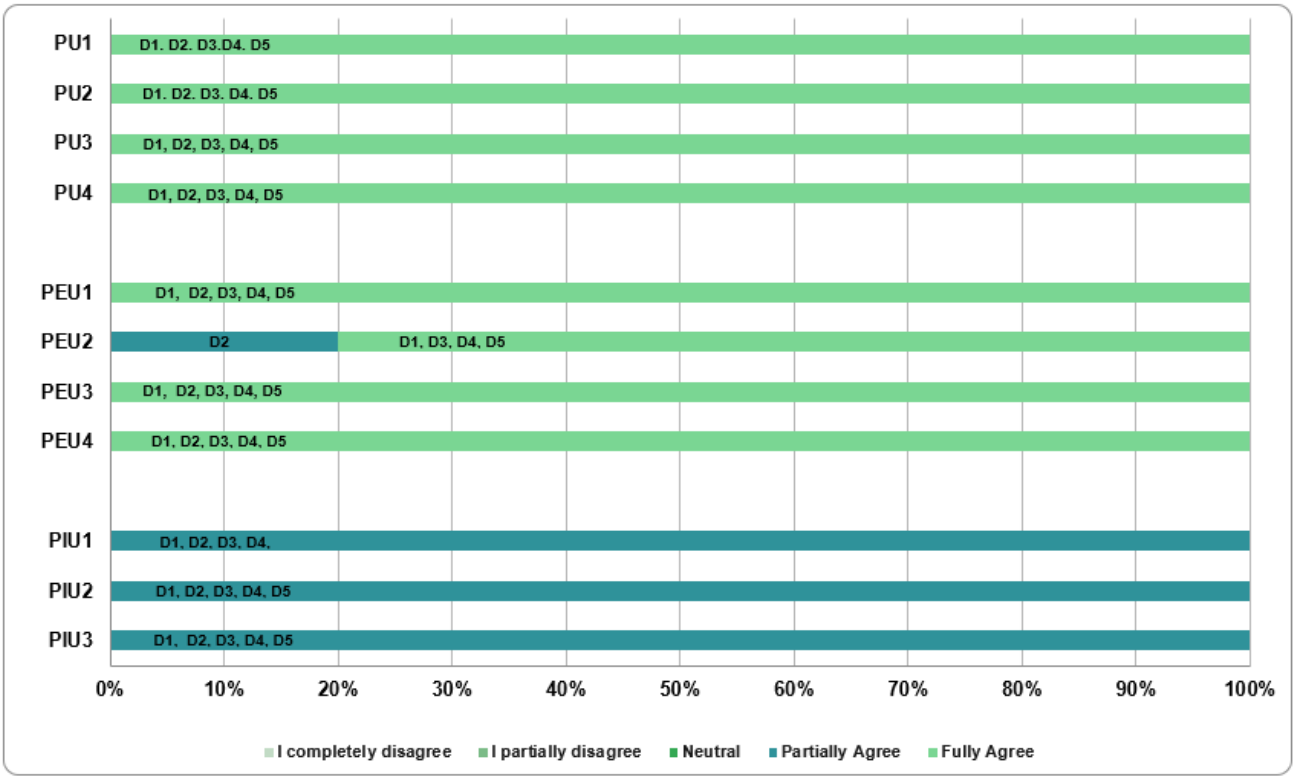


FIGURE 5. General results of perceptions about CollabProg.

Finally, the Perceived Future Use Intention of CollabProg, all instructors partially agreed with the three statements (PIU1, PIU2, PIU3). The intention to use CollabProg is crucial for assessing the community's interest in the tool and its acceptance as a support resource for programming education. Instructors evaluated the CollabProg repository positively and expressed their intention to use it in the future.

7) PARTICIPANTS' PERCEPTIONS ON THE USE OF CollabProg

To assess the instructors' experience using CollabProg, we analyzed the following statement: **"Using CollabProg contributed to adopting active methodologies in my programming teaching classes."** Overall, the perception was positive, with several relevant aspects regarding the experience of using CollabProg. Regarding the positive points, we identified four subcategories that address the benefits of the repository.

In the first subcategory, **detailed explanation of the steps**, I1 commented: *I had no prior knowledge about the methodologies, and CollabProg allowed me to apply them easily.*" I3 stated, *CollabProg greatly facilitated the understanding of the ALM available in the repository.*" I3 also mentioned that *on previous occasions, they had considered using POGIL in their classes*, but its documentation was extensive. They added that "the way it was presented in CollabProg was much more intuitive for understanding how this ALM works and planning the classes." In the second subcategory, **increased productivity in implementing ALMs**, I2 highlighted that "without CollabProg, I would rarely seek out the details of an ALM to teach programming."

Regarding the third subcategory, **usefulness of practical examples**, I3 expressed that *"the presented examples were very helpful in understanding how to adopt the methodology."* The instructor added that they often read about methodologies, but the content remains very abstract. They noted that "having the steps and examples makes it much easier to understand how to apply the methodology." Finally, regarding the fourth subcategory, **encouragement of collaborative work and active student participation**, D4 stated that "collaborative work enriches learning," and I5 shared that *"the main advantage, in my opinion, is encouraging students to participate more actively in class, leading to better learning outcomes."*

The instructors' perceptions confirmed their interest in using CollabProg and contributing to its evolution. Regarding the platform's usefulness, all instructors agreed that the CollabProg repository could significantly contribute to class planning by adopting ALMs. Additionally, the majority found the content presented in the repository highly helpful.

The instructors also identified some negative points. The first is related to **difficulty in understanding the steps and concepts**. I1 commented on having difficulty in *"understanding the active methodologies (some steps I had*

to read several times)." I2 added, "*Although it is very well organized, I still had difficulties setting up the step-by-step.*" The second negative point is related to **challenges in setting up the step-by-step and confusion in specific points**. In this regard, I3 emphasized that "*some points were confusing during the reading of the active methodology, especially POGIL, which I adopted.*" I5 stated that their main difficulty was "*building a lesson plan that accurately reflected the active methodology in question.*"

In conclusion, the implementation of ALMs revealed several challenges identified by the instructors. A significant issue was the difficulty in understanding the steps and concepts, with some instructors struggling to fully grasp the methodologies and often needing to reread the steps multiple times. Additionally, there were challenges in setting up the step-by-step process and confusion around specific points. This was particularly evident with methodologies like POGIL, where the complexity and lack of clarity made it difficult for instructors to create lesson plans accurately reflecting the intended active learning strategies. These challenges underscore the need for improved instructional materials and training to better support educators in effectively adopting and implementing active methodologies.

The results of the first design cycle provided valuable insights into the repository's acceptance and feasibility. Focused on supporting the adoption of ALMs in programming education, CollabProg was evaluated by instructors from various educational institutions across Brazil. Based on these findings, we made efforts to improve CollabProg, particularly addressing the negative aspects and needs identified by the instructors. Consequently, we conducted a new design cycle and developed the second version of CollabProg, aiming to meet its users' demands and expectations better.

VII. SECOND DESIGN CYCLE: CollabProg 2.0

CollabProg was evaluated by instructors from various higher education institutions who voluntarily participated. Based on the study results, we analyzed CollabProg from the perspective of Design Criteria, which outline user expectations for the artifact. We observed that Requirement **DC1**—the artifact should provide a variety of ALMs, including detailed descriptions, application examples, and usage contexts—was not fully met during the study. This limitation was identified through instructor feedback, which highlighted the need for a more significant number of active methodologies available in the system. To address this issue, we planned a more comprehensive curation process, and the inclusion of methodologies used in combination with various approaches, as well as methodologies developed and implemented by the authors of primary studies, as detailed in the results of the SMS, available in [29].

Regarding **DC2**, which states that the artifact should provide clear and structured guidelines for implementing each ALM, covering aspects such as planning, execution, and evaluation, most evaluating instructors found the information available on the portal adequate and valuable. Regarding **DC3**, which requires the artifact to provide curated ALMs (a process of carefully selecting and organizing ALM content), including critical analyses, evidence-based recommendations, and feedback from instructors who have implemented these methodologies, this process ensured that the methodologies were scientifically grounded and relevant for practical application in programming education. The evaluation of CollabProg from the perspective of Behavior Criteria, which refer to the artifact's contribution to programming teaching practices, revealed several areas for improvement.

Instructors suggested improvements to the repository regarding **BC1**, which requires the artifact to support users in understanding active methodologies. They emphasized the need to make explanations of the steps and concepts of the methodologies clearer and more accessible, facilitating user comprehension. Implementing detailed tutorials and practical examples could help better meet this requirement. As for **BC2**, which aims to motivate instructors to adopt ALM, CollabProg was well-received. Instructor D3 mentioned that the intuitive presentation of the methodologies in CollabProg made it easier to understand their operation and plan lessons than other documentation sources. This positive feedback indicates that CollabProg is effectively promoting the adoption of ALMs.

Regarding **BC3**, which requires the presentation of clear and detailed information about ALMs, we identified the need for improvements in documentation and the examples provided. Participants emphasized the importance of more detailed explanations regarding the assignment of roles in ALMs, aiming to reduce confusion and facilitate implementation. Enhancing the documentation with specific case studies and step-by-step descriptions could help better meet this requirement. Finally, regarding **BC4**, which requires that guidelines be detailed to facilitate their implementation, CollabProg partially met this requirement. Assessments by instructors D1 and D3 indicated that, while CollabProg facilitated the application of methodologies and improved understanding of the available ALMs, there is still room to make the guidelines more detailed and practical. Incorporating checklists, flowcharts, and additional visual resources could enhance the effectiveness of the guidelines.

Based on the evaluation of CollabProg and the instructors' suggestions, the repository represents a valuable tool for supporting programming education with ALMs. CollabProg was praised for its ability to motivate the adoption of ALMs and facilitate understanding of their implementation. Opportunities for improvement were identified, particularly in enhancing the clarity and simplicity of explanations, providing more detailed documentation of methodologies, and clarifying the roles assigned in each. Additionally, the positive feedback on the ease of implementing ALMs demonstrates CollabProg's potential as a valuable and effective tool for instructors seeking to adopt active approaches to teaching programming.

Figure 6 shows the homepage of CollabProg version 2.0 (in Portuguese). The labeled menus and their respective icons

are displayed on the left side (Part 1 of Figure 6). This design aims to provide a cleaner, more intuitive, and aesthetically pleasing interface, enhancing the user experience. The **Home** menu directs users to the CollabProg homepage. Part 2 of Figure 6 presents information about the ALMs and details about CollabProg itself.

Figure 7 shows the **Register Methodology** menu, which directs users to the methodology registration page. On this page, instructors are invited to provide details about the ALM, including its description, educational objectives, implementation steps, suggested activities, and necessary resources. Each field is accompanied by a detailed explanation of how to complete it, ensuring accurate registration and providing helpful information for the community using the methodology.

For the registration of methodology details, users are guided through five pages, each containing specific fields for collecting the required information. Figure 7 presents this functionality's first and second pages. This feature enriches the tool by expanding the diversity of ALMs available to other instructors. To register a methodology, users must complete mandatory fields, such as the methodology description, taught disciplines, related content or categories, programming languages used, necessary materials, methodology principles, methodology planning, and steps for adopting the methodology. This information is essential for the community to use and follow the tried-and-tested step-by-step process effectively.

Figure 8 shows the **Methodologies** menu. On this page, CollabProg provides detailed information about each methodology, including the step-by-step implementation process, the roles of students and instructors, the necessary materials and tools, the average time required for lesson planning, the steps for adopting the methodology, and guidance on assessing learning, among other details. These guidelines help instructors understand how to implement the ALM in their classrooms effectively.

After accessing the ''Methodologies'' menu, instructors are presented with a list of available methodologies for implementation. Upon selecting a methodology of interest, they are directed to the initial screen for the chosen methodology. On this screen, depicted in Figure 9, general information is displayed, such as the time required for applying the methodology and the recommended class size. Additionally, specific objectives of the methodology are provided, helping instructors make informed decisions based on their current context.

Additionally, in the **Methodologies** menu, instructors can access the **View Feedback** button (Figure 10). This feature allows instructors to see evaluations from other instructors about specific methodologies. Evaluations are shared through star ratings and comments, providing insights into the implementation experience of the methodology in different contexts. We believe the ''View Feedback'' feature is essential for the academic community. It promotes transparency and trust by allowing users to access evaluations and testimonials from other instructors about the implemented methodologies. This feature facilitates sharing experiences, offering valuable lessons learned and best practices that can benefit new users. By enabling more informed choices and inspiring contextual adaptations, the ''View Feedback'' feature strengthens collaboration and community engagement, fostering an environment of mutual support and continuous learning.

We will systematically collect and analyze user feedback to ensure the CollabProg platform remains relevant and practical. This strategy will encourage instructors to share their experiences after applying methodologies, detailing the results, challenges encountered, and any adaptations made. Aggregated feedback will be analyzed to identify trends, strengths, and areas for improvement, providing valuable insights to guide the platform's evolution. This includes updating methodology models, improving usability, and addressing specific user needs.

The systematic approach adopted by CollabProg to collect and analyze user feedback is structured into stages that ensure the acquisition of relevant data, efficient analysis, and practical application in improving the platform. This approach includes the following key elements:

- **Structured Feedback Collection:** Users are encouraged to provide feedback after implementing a methodology through tools integrated into the platform. These tools include simple evaluation forms, star ratings, and fields for detailed comments.
- **Feedback Analysis and Categorization:** All feedback received is categorized based on factors such as the application context (e.g., course type, class size), methodology strengths, challenges encountered, and suggestions for improvement.
- **Practical Application and Updates:** Insights derived from feedback are directly incorporated into CollabProg updates. This may include improving methodology descriptions, developing new features, or refining the platform's usability.

By adopting this systematic approach, CollabProg ensures that users' voices are central to the platform's continuous development. This fosters an environment that addresses instructors' real needs and enhances the effectiveness of programming education.

The **Recommendations** menu is currently in the study and development phase. This feature aims to implement a fundamental resource for personalizing the user experience. The idea is to use intelligent algorithms to analyze user preferences and context, such as taught disciplines, level of education, and educational objectives, to recommend ALMs that best suit their profile. By providing personalized recommendations, this feature will help instructors explore new teaching approaches aligned with their specific needs and goals. Additionally, it will promote the discovery of innovative and practical methodologies, increasing the diversity and quality of teaching practices adopted by the

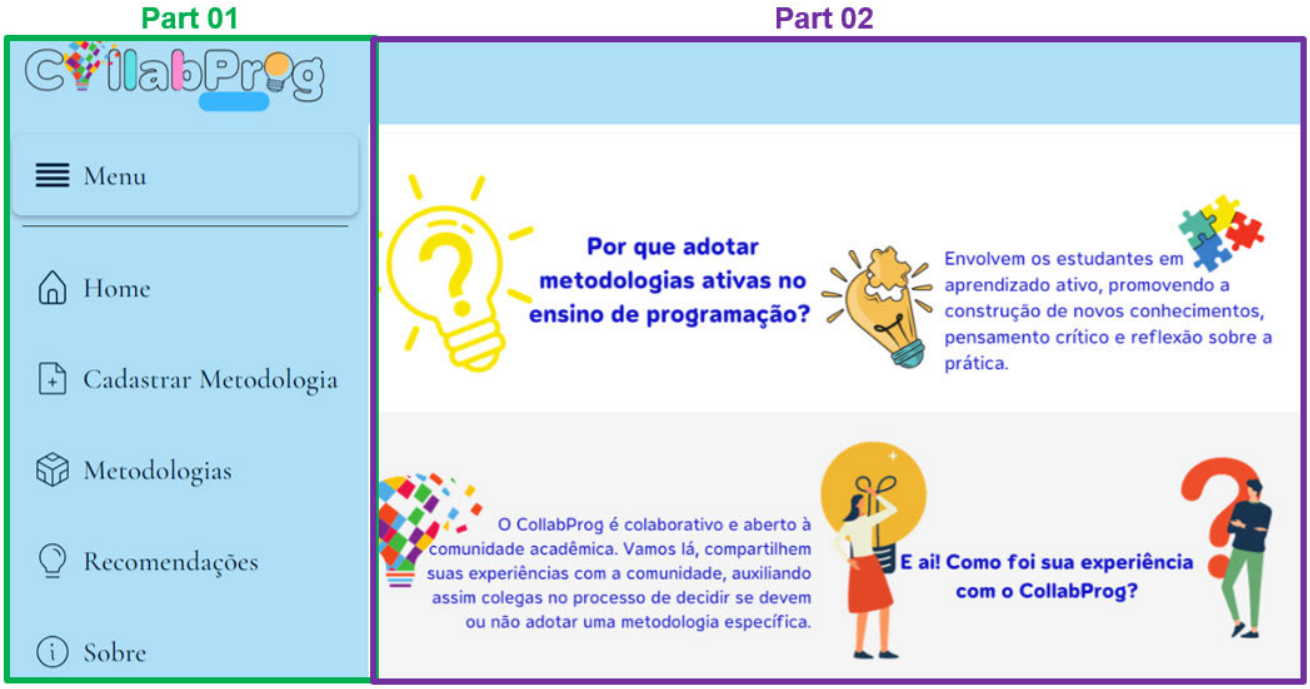


FIGURE 6. CollabProg 2.0 Homepage.

academic community. The careful implementation of the ''Recommendations'' feature, therefore, not only enhances the tool's usability but also significantly enriches the learning and teaching experience in programming and computer science.

VIII. RIGOR CYCLE

The rigor of this research in developing CollabProg has been considered an essential aspect. Research rigor is associated with credibility, reliability, precision, and integrity, demanding theoretical and methodological rigor [60]. Rigor is ensured when researchers follow a previously established and validated research method, preferably one widely recognized and accepted by the academic community. The rigor of this research guided us to utilize solid theoretical foundations and existing technical knowledge.

Thus, we employed research methods to document the steps taken during the DSRM cycles, ensuring the necessary rigor in this research. Notable among these methods are the SMS presented in Section V and the experimental study using the TAM, as detailed in Section VI-E. The TAM model is frequently used in technology acceptance and adoption research due to its theoretical robustness and broad applicability across various contexts. An essential stage in the rigor cycle is for researchers to report the main contributions of their research. The significant contributions of this research so far include:

- **Identification, classification, and analysis of existing evidence:** We cataloged and analyzed evidence on the types of ALMs applied in programming education in Computer Science.
- **Understanding and analysis of instructors' perceptions:** We investigated and comprehended instructors' perceptions and difficulties in adopting ALMs, specifically in programming disciplines.
- **Application of Design Science Research:** We have uniquely applied the DSRM to develop, evaluate, and evolve a collaborative and open repository, CollabProg. This innovative approach can potentially inspire other researchers and instructors in the field.
- **Conducting an exploratory study:** We conducted an exploratory study that has direct implications for using and accepting CollabProg from the instructors' perspective, providing practical insights for instructors.
- **Practical evidence of support for instructors**: We evidenced that CollabProg assisted instructors from five different educational institutions in adopting ALMs in programming education.
- **Practical evidence of ALMs' effectiveness:** Our research not only identified the efficacy of the ALMs in the literature but also demonstrated their practical application in the classroom. This implementation led to a tangible improvement in the teaching process in programming disciplines.

These contributions reflect the breadth and depth of the work undertaken, providing a solid foundation for the continuous evolution of CollabProg and improving pedagogical practices in programming education within Computer Science.

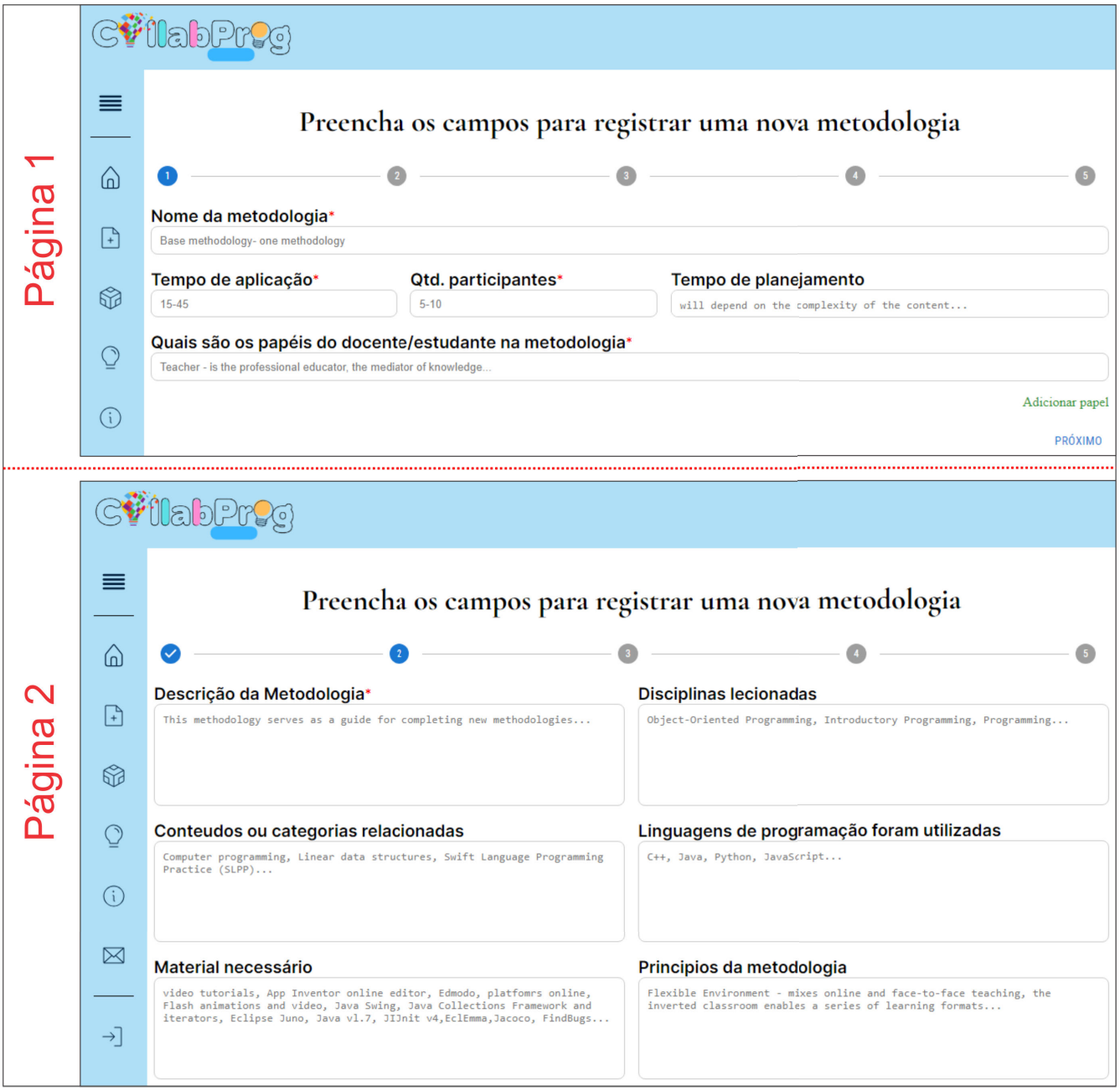


FIGURE 7. Methodology registration page.

IX. LIMITATIONS

While CollabProg offers significant advantages in supporting instructors with ALMs for programming education, its implementation has challenges or limitations. Despite the progress made with CollabProg, several limitations still need to be addressed so that the repository can reach its full potential as a robust tool for selecting ALMs. Currently, the automated recommendation system for choosing ALMs is still under development. We are exploring algorithms and methods that can suggest the most appropriate pedagogical approaches based on the specific context provided by instructors, considering factors such as student experience levels, learning objectives, and available resources. The absence of this functionality limits CollabProg's ability to offer personalized and automated recommendations, requiring more significant manual intervention from instructors.

Another significant challenge lies in the initial stage of community collaboration features. Although the repository was designed to allow instructors to share experiences and

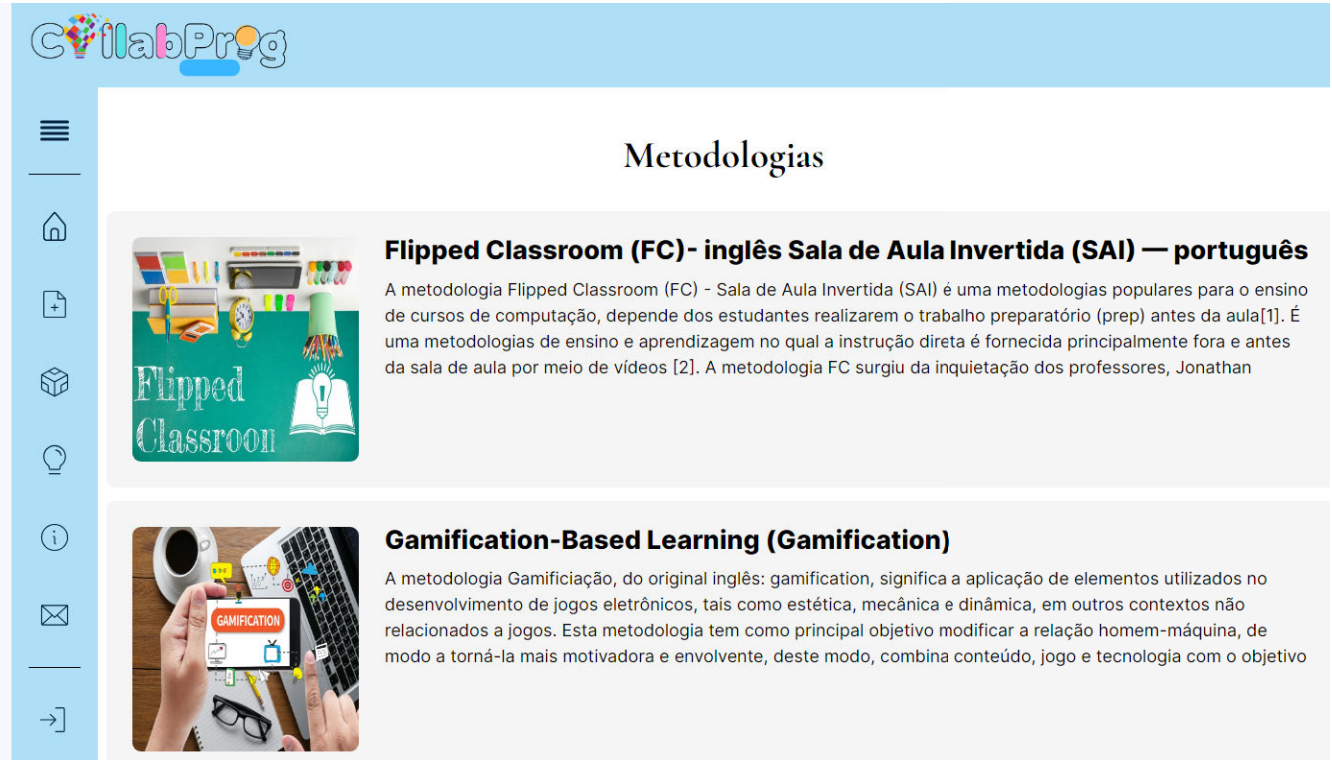


FIGURE 8. Active learning methodologies page.

insights on using different ALMs in programming education, active community participation remains limited. We are implementing enhancements to facilitate user interaction, enabling them to contribute new practices, comment on existing methodologies, and benefit from the experiences of other educators. These improvements aim to foster a culture of knowledge sharing and strengthen CollabProg's utility as a genuinely collaborative repository.

Finally, the curation process for the methodologies available in the repository remains partially dependent on the project's internal team. While this initial approach has enabled a thorough analysis of the included methodologies, it also introduces potential biases and limits the scope of curation. We are exploring strategies to allow contributions from other educators and automation solutions to improve the evaluation of ALMs and ensure greater diversity and objectivity in the selection process. Overcoming these challenges will be essential for expanding CollabProg's impact and providing more comprehensive and reliable support to the educational community. However, these limitations do not hinder the current use of the repository; they represent challenges that will be addressed over time.

X. RESEARCH IMPLICATIONS

The research presents several implications that could positively impact teaching practices, the work of researchers in computer science education, and students' academic development.

CollabProg offers instructors a collaborative portal with diverse educational resources on adopting active learning methodologies, including tool recommendations, student assessment guidance, and the development of supplementary teaching materials.

With CollabProg developed, instructors will have access to a portal offering diversified educational guidance related to adopting ALMs. As a collaborative repository, CollabProg will provide opportunities for instructors to access, for example, recommendations for tools that can be used with ALM adoption, guidance on assessing students during classes and preparing complementary teaching materials. Instructors may also choose to develop complementary materials for the resources in the repository, providing additional guidance and contextualizing learning within the classroom curriculum.

instructors could be encouraged to adapt their teaching methods and commit to ongoing professional development to maximize the resources' potential and effectively support student learning.

Instructors can benefit from collaborating with colleagues to share effective teaching practices and strategies for using the repository. This may include exchanging ideas on integrating repository resources into different disciplines or educational contexts related to programming education. Finally, using the collaborative repository will require

FIGURE 9. Methodology details.

FIGURE 10. View feedback button *feedback*.

instructors to adapt their teaching practices and commit to continuous professional development, ensuring they can maximize the available resources and effectively support their students' learning journeys.

> CollabProg enhances teaching by promoting active learning through programming projects, facilitating student collaboration, and enabling regular assessments to identify and address learning difficulties.

Given this scenario of new possibilities for teaching practices using CollabProg, the implications for students are centered on active and practical learning. Instructors will have the support to promote hands-on programming activities and projects where students can apply theoretical concepts learned in the classroom. This approach will reinforce learning and enable students to develop practical problem-solving skills. Another critical aspect is fostering collaboration among students, whether through group projects, pair programming activities, or classroom discussions, which can encourage knowledge sharing, collaborative problem-solving, and the development of essential social skills. Additionally, with the support of CollabProg, instructors can plan and incorporate regular formative assessments, such as quick quizzes, code reviews, and group discussions, to identify areas of difficulty and adjust instruction as needed, ensuring a solid understanding of concepts.

> Researchers can explore gaps in computer science education, focusing on integrating ALMs to enable the development of personalized teaching methods and investigating best practices to enhance computing education practices.

Finally, researchers may identify research gaps in computer science education, particularly in integrating new technologies and methodologies, such as ALMs, and their applications in educational contexts. This presents opportunities to develop new teaching and assessment methodologies incorporating ALM concepts, promoting a more personalized and adaptive approach to computing education. Additionally, conducting investigations into best practices for ALM adoption in computer science education—including implementation strategies, challenges faced, and lessons learned—can inform pedagogical practices and educational policies for computing education.

XI. FINAL CONSIDERATIONS AND FUTURE WORK

The adoption of ALMs is becoming increasingly prominent in computer science courses. However, instructors face various barriers that hinder the adoption of these methodologies. Inspired by the DSRM, we explored ways to assist instructors in adopting ALMs in programming education. The application of DSRM enabled the definition of the research problem and the development, evaluation, and enhancement of an artifact.

To deepen our understanding of the research problem, we initially conducted an SMS to identify the ALMs used by instructors in programming education. This study revealed 37 different types of ALMs adopted in programming education. Additionally, we identified 17 publications that address the combination of multiple ALMs and four others that present proposals for new methodologies. Instructors developed or proposed these latter approaches, demonstrating the diverse strategies employed in their teaching practices to promote active learning during programming classes. The detailed results of this study can be found in [29].

After completing the SMS, the curation process of the ALMs integrated into CollabProg began. This process prioritized sharing only content and tool support options available in the literature for instructors' use. The goal was to avoid frustrations for repository users by ensuring that only knowledge and content supported by scientific evidence, experimentation, or proven relevance would be presented. The selection of ALMs was meticulous, prioritizing methodologies grounded in solid scientific evidence and excluding those lacking experimentation or whose theoretical basis was deemed irrelevant to the community within the context of this research.

Based on the SMS results and the curation process of primary studies in the Design Cycle of the DSRM, we developed, evaluated, and enhanced the CollabProg artifact. CollabProg is a collaborative and open repository designed to support instructors in adopting the most appropriate ALMs for their teaching context in programming education. We conducted an experimental study with five higher education institutions in Brazil to assess the feasibility and acceptance of CollabProg from the instructors' perspectives. This highlights the importance of seeking strategies to support instructors in programming education and to motivate students in programming learning, as these are critical factors for successful instruction. This factor is particularly relevant in collaborative learning contexts, where social interaction plays a significant role [68] in the adoption of ALMs.

In future work, we intend to develop and refine a model of challenges related to adopting ALMs in programming education to evaluate and validate CollabProg. This comprehensive model will be built based on the results of the experimental study. The goal is to design a model from the perspectives

and experiences of programming course instructors. A survey will be conducted with instructors to evaluate the model, and their feedback and evaluations will guide its evolution. Additionally, we will validate the model through instructors' applications. This approach will ensure that CollabProg is evaluated based on diverse experiences, needs, and contexts of teaching programming.

In addition to refining the SMS of ALMs based on the latest literature, we will continuously update the methodologies available on CollabProg to ensure the platform remains aligned with evolving pedagogical trends and the practical needs of instructors. This ongoing effort is essential for integrating innovative and effective practices into programming education. We also aim to enhance engagement with the teaching community, particularly among instructors who continue to rely on traditional teaching methods. By offering tutorials and awareness resources, we seek to highlight the benefits of ALMs and make their adoption more accessible and appealing to educators. Lastly, we plan to translate the platform into multiple languages, enabling us to reach a global audience of educators. The internationalization of CollabProg will not only broaden its impact but also foster a worldwide network of instructors who can share best practices and contribute to the continuous innovation of programming education across diverse educational contexts.

In conclusion, we expect CollabProg to serve as a technological support platform that consolidates strategies for adopting various ALMs in programming education within a single internet portal within a single internet portal. It will provide examples, activity suggestions, support options, tools adopted by the community, experiences of methodology adoption in different scenarios, results achieved by other instructors, and insights into the positive and negative aspects of the adopted ALMs.

ACKNOWLEDGMENT

The authors would like to acknowledge the use of Grammarly and ChatGPT 4.o for improving the spelling, grammar, vocabulary, and style of the text. All suggestions were carefully examined, tested, and often corrected by them, whereby they take full responsibility for the form and content of the article.

REFERENCES

[1] S. C. dos Santos, P. B. S. Reis, J. F. S. Reis, and F. Tavares, "Two decades of PBL in teaching computing: A systematic mapping study," *IEEE Trans. Educ.*, vol. 64, no. 3, pp. 233–244, Aug. 2021.
[2] J. Eickholt, "Barriers to active learning for computer science faculty," 2018, *arXiv:1808.02426*.
[3] Y.-C. Liao and M. Ringler, "Backward design: Integrating active learning into undergraduate computer science courses," *Cogent Educ.*, vol. 10, no. 1, Dec. 2023, Art. no. 2204055.
[4] O. E. Holo, E. N. Kveim, M. S. Lysne, L. H. Taraldsen, and F. O. Haara, "A review of research on teaching of computer programming in primary school mathematics: Moving towards sustainable classroom action," *Educ. Inquiry*, vol. 14, no. 4, pp. 513–528, Oct. 2023.
[5] I. Eteng, S. Akpotuzor, S. O. Akinola, and I. Agbonlahor, "A review on effective approach to teaching computer programming to undergraduates in developing countries," *Sci. Afr.*, vol. 16, Jul. 2022, Art. no. e01240.
[6] C. W. Okonkwo and A. Ade-Ibijola, "Synthesis of nested loop exercises for practice in introductory programming," *Egyptian Informat. J.*, vol. 24, no. 2, pp. 191–203, Jul. 2023.
[7] C. L Corritore and B. Love, "Redesigning an introductory programming course to facilitate effective student learning: A case study," *J. Inf. Technol. Educ., Innov. Pract.*, vol. 19, pp. 91–135, Aug. 2020.
[8] R. Caceffo, G. Gama, and R. Azevedo, "Exploring active learning approaches to computer science classes," in *Proc. 49th ACM Tech. Symp. Comput. Sci. Educ.*, Feb. 2018, pp. 922–927.
[9] S. R. Sobral, "Strategies on teaching introducing to programming in higher education," in *Advances in Intelligent Systems and Computing*, vol. 3. Cham, Switzerland: Springer, 2021, pp. 133–150.
[10] F. W. Da Silva Garcia, E. Da Costa Carvalho, and S. R. B. Oliveira, "Use of active methodologies for the development of a teaching plan for the algorithms subject," in *Proc. IEEE Frontiers Educ. Conf. (FIE)*, Oct. 2021, pp. 1–9.
[11] P. Remit and S. R. Sobral, "Project based learning with peer assessment in an introductory programming course," *Int. J. Inf. Educ. Technol.*, vol. 11, no. 7, pp. 337–341, 2021.
[12] V. V. Lima, "Constructivist spiral: An active learning methodology," *Interface (Botucatu).[Internet]*. vol. 21, no. 61, pp. 421–434, Jul. 2022.
[13] S. Arik and M. Yilmaz, "The effect of constructivist learning approach and active learning on environmental education: A meta-analysis study," *Int. Electron. J. Environ. Educ.*, vol. 10, no. 1, pp. 44–84, 2020.
[14] I. Sasson, I. Yehuda, S. Miedijensky, and N. Malkinson, "Designing new learning environments: An innovative pedagogical perspective," *Curriculum J.*, vol. 33, no. 1, pp. 61–81, Mar. 2022.
[15] J. F. Travers, S. N. Elliott, and T. R. Kratochwill, *Educational Psychology: Effective Teaching, Effective Learning*. Brown & Benchmark/Wm. C. Brown Publ, 1993.
[16] J. H. Berssanette and A. C. de Francisco, "Active learning in the context of the teaching/learning of computer programming: A systematic review," *J. Inf. Technol. Educ., Res.*, vol. 20, pp. 201–220, Jan. 2021.
[17] K. A. Nguyen, M. Borrego, C. J. Finelli, M. DeMonbrun, C. Crockett, S. Tharayil, P. Shekhar, C. Waters, and R. Rosenberg, "Instructor strategies to aid implementation of active learning: A systematic literature review," *Int. J. STEM Educ.*, vol. 8, no. 1, pp. 1–18, Dec. 2021.
[18] V. Sharma, K. K. Bhagat, H.-H. Huang, and N.-S. Chen, "The design and evaluation of an AR-based serious game to teach programming," *Comput. Graph.*, vol. 103, pp. 1–18, Apr. 2022.
[19] L. Freire, J. Coutinho, V. Lima, and N. Lima, "Uma proposta de encontros de tutoria baseada em metodologias ativas para disciplinas de Programação Introdutória," in *Proc. Anais dos Workshops do VIII Congresso Brasileiro de Informática na Educação (CBIE)*, Nov. 2019, p. 298.
[20] N. M. Bigolin, S. R. Silveira, C. Bertolini, I. C. D. Almeida, M. Geller, F. J. Parreira, G. B. D. Cunha, and R. T. Macedo, "Metodologias ativas de aprendizagem: Um relato de experiência nas disciplinas de programação e estrutura de dados," *Res., Soc. Develop.*, vol. 9, no. 1, Jan. 2020, Art. no. e74911648.
[21] A. Luxton-Reilly, Simon, I. Albluwi, B. A. Becker, M. Giannakos, A. N. Kumar, L. Ott, J. Paterson, M. J. Scott, J. Sheard, and C. Szabo, "Introductory programming: A systematic literature review," in *Proc. Companion 23rd Annu. ACM Conf. Innov. Technol. Comput. Sci. Educ.*, Jul. 2018, pp. 55–106.
[22] A. G. S. Raj, J. Patel, and R. Halverson, "Is more active always better for teaching introductory programming?" in *Proc. Int. Conf. Learn. Teaching Comput. Eng. (LaTICE)*, Apr. 2018, pp. 103–109.
[23] S. Acharya and M. N. Gayana, "Enhanced learning and improved productivity of students' using project based learning approaches for programming courses," *J. Eng. Educ. Transformations*, vol. 34, p. 524, Jan. 2021.
[24] J. Penney, "Anticipating user needs: Insights from design fiction on conversational agents for computational thinking," in *Proc. Int. Workshop Chatbot Res. Design*. Cham, Switzerland: Springer, 2023, pp. 204–219.
[25] T. Beaubouef and J. Mason, "Why the high attrition rate for computer science students: Some thoughts and observations," *ACM SIGCSE Bull.*, vol. 37, no. 2, pp. 103–106, Jun. 2005.
[26] A. Luxton-Reilly, "Learning to program is easy," in *Proc. ACM Conf. Innov. Technol. Comput. Sci. Educ.*, Jul. 2016, pp. 284–289.
[27] P. Denny, A. Luxton-Reilly, E. Tempero, and J. Hendrickx, "Understanding the syntax barrier for novices," in *Proc. 16th Annu. joint Conf. Innov. Technol. Comput. Sci. Educ.*, Jun. 2011, pp. 208–212.
[28] P. Parsons, "Preparing computer science graduates for the 21st Century," *Teach. Innov. Projects*, vol. 1, no. 1, pp. 1–6, 2011.

[29] I. Calderon, W. Silva, and E. Feitosa, "Active learning methodologies for teaching programming in undergraduate courses: A systematic mapping study," *Informat. Educ.*, vol. 23, pp. 279–322, Sep. 2023.
[30] M. J. O'Grady, "Practical problem-based learning in computing education," *ACM Trans. Comput. Educ.*, vol. 12, no. 3, pp. 1–16, Jul. 2012.
[31] A. Imbulpitiya, N. Kodagoda, A. Gamage, and K. Suriyawansa, "Using active learning integrated with pedagogical aspects to enhance student's learning experience in programming and related concepts," in *Proc. Int. Conf. Interact. Collaborative Learning*. Cham, Switzerland: Springer, 2020, pp. 218–228.
[32] L. Bacich and J. Moran, *Metodologias Ativas Para Uma Educação Inovadora: Uma Abordagem Teórico-Prática*. New York, NY, USA: Penso Editora, 2017.
[33] D. Witt, K. Teixeira, D. S. Avanilde, and L. Mulazani, "Resolução de problemas: Abordagens aplicadas no ensino de computação," *Anais do Comput. Beach*, vol. 9, pp. 731–740, May 2018.
[34] D. Pundak and S. Rozner, "Empowering engineering college staff to adopt active learning methods," *J. Sci. Educ. Technol.*, vol. 17, no. 2, pp. 152–163, Apr. 2008.
[35] R. F. Behnagh and S. Yasrebi, "An examination of constructivist educational technologies: Key affordances and conditions," *Brit. J. Educ. Technol.*, vol. 51, no. 6, pp. 1907–1919, Nov. 2020.
[36] 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.
[37] N. Yannier, "Active learning: 'Hands-on' meets 'minds-on,'" *Science*, vol. 374, no. 6563, pp. 26–30, 2021.
[38] K. Matsushita, "An invitation to deep active learning," in *Deep Active Learning: Toward Greater Depth in University Education*, 2018, pp. 15–33.
[39] Ö. Tutal and T. Yazar, "Active learning promotes more positive attitudes towards the course: A meta-analysis," *Rev. Educ.*, vol. 10, no. 1, Apr. 2022, Art. no. e3346.
[40] N. F. Hassan and S. Puteh, "A survey of technology enabled active learning in teaching and learning practices to enhance the quality of engineering students," *Adv. Sci. Lett.*, vol. 23, no. 2, pp. 1104–1108, Feb. 2017.
[41] M. Elahi, F. Ricci, and N. Rubens, "A survey of active learning in collaborative filtering recommender systems," *Comput. Sci. Rev.*, vol. 20, pp. 29–50, May 2016.
[42] J. Bishop and M. Verleger, "The flipped classroom: A survey of the research," in *Proc. ASEE Annu. Conf. Expo.*, 2013, p. 23.
[43] X. Suo, O. Glebova, D. Liu, A. Lazar, and D. Bein, "A survey of teaching PDC content in undergraduate curriculum," in *Proc. IEEE 11th Annu. Comput. Commun. Workshop Conf. (CCWC)*, Jan. 2021, pp. 1306–1312.
[44] B. L. Wiggins, S. L. Eddy, L. Wener-Fligner, K. Freisem, D. Z. Grunspan, E. J. Theobald, J. Timbrook, and A. J. Crowe, "ASPECT: A survey to assess student perspective of engagement in an active-learning classroom," *CBE—Life Sci. Educ.*, vol. 16, no. 2, p. ar32, Jun. 2017.
[45] V. Villas-Boas, O. M. Neto, L. C. Campos, and B. Aguiar, "A survey of active learning in Brazilian engineering schools," in *Proc. Act. Learn. Eng. Educ. Workshop*, 2012, pp. 1–12.
[46] J. V. V. Lima, C. A. D. Silva, F. M. R. de Alencar, and W. B. Santos, "Metodologias ativas como forma de reduzir os desafios do ensino em engenharia de software: Diagnóstico de um survey," in *Proc. Anais do XXXI Simpósio Brasileiro de Informática na Educação (SBIE)*, Nov. 2020, pp. 172–181.
[47] B. A. Kitchenham and S. L. Pfleeger, "Personal opinion surveys," in *Guide To Advanced Empirical Software Engineering*. London, U.K.: Springer, 2008, pp. 63–92.
[48] M. Oliveira, S. R. B. Oliveira, and S. Meira, "Condução de uma Fábrica de software e o processo de aprendizagem em cursos de Graduação de TI: Uma Aplicação de um survey sobre a Percepção da importância," in *Proc. Simpósio Brasileiro de Informática na Educação*, vol. 1, Oct. 2017, p. 92.
[49] J. A. Coelho, G. H. Souza, J. Alburquere, "Desenvolvimento de questionários e aplicação na pesquisa em Informática na Educação," in *Metodologia de Pesquisa em Informática na Educação: Abordagem Quantitativa de Pesquisa* (Série Metodologia de Pesquisa em Informática na Educação), vol. 2. Porto Alegre, Brazil: SBC, 2020.
[50] P. M. Nardi, *Doing Survey Research: A Guide to Quantitative Methods*. Evanston, IL, USA: Routledge, 2018.
[51] D. Mendez, D. Graziotin, S. Wagner, and H. Seibold, "Open science in software engineering," in *Contemporary Empirical Methods in Software Engineering*. Berlin, Germany: Springer, 2020, pp. 477–501.
[52] I. Calderon, W. Silva, and E. Feitosa, "Um Mapeamento Sistemático da Literatura sobre o uso de Metodologias Ativas durante o Ensino de Programação no Brasil," in *Proc. Anais do XXXII Simpósio Brasileiro de Informática na Educação*, 2021, pp. 1152–1161.
[53] S. R. Sobral, "Two different experiments on teaching how to program with active learning methodologies: A critical analysis," in *Proc. 15th Iberian Conf. Inf. Syst. Technol. (CISTI)*, Jun. 2020, pp. 1–7.
[54] M. L. Kovarik, J. K. Robinson, and T. J. Wenzel, "Why use active learning?" in *Active Learning in the Analytical Chemistry Curriculum*. Olympia, WA, USA: American Chemical Society, Jan. 2022, pp. 1–12.
[55] R. M. D. Castro and S. Siqueira, "ALCASYSTEM–Um portal com Técnicas de aprendizagem ativa para disciplinas da Área da Computação," in *Proc. Anais dos Workshops do VIII Congresso Brasileiro de Informática na Educação (CBIE)*, Nov. 2019, p. 1243.
[56] W. A. F. Silva, "OPENSMALS: Um repositório aberto para auxiliar no ensino de modelagem de software empregando estratégias de aprendizagem ativa," 262f. Tese (Doutorado em Informática)-Universidade Federal do Amazonas, Manaus.
[57] J. Lima, F. Alencar, and W. Santos, "A preliminary guide for assertive selection of active methodologies in software engineering education," in *Proc. Brazilian Symp. Softw. Eng.*, Sep. 2021, pp. 170–179.
[58] R. Ahshan, "A framework of implementing strategies for active student engagement in remote/online teaching and learning during the COVID-19 pandemic," *Educ. Sci.*, vol. 11, no. 9, p. 483, Aug. 2021.
[59] A. Hevner and S. Chatterjee, "Design science research in information systems," in *Design Research in Information Systems: Theory and Practice*, 2010, pp. 9–22.
[60] R. J. Wieringa, *Design Science Methodology for Information Systems and Software Engineering*. Cham, Switzerland: Springer, 2014.
[61] A. C. B. Angeluci, *Design Science Research Como Método Para Pesquisas em TIC na Educação*. São Carlos, Brazil: Anais CIET, Horizonte, 2020.
[62] B. A. Kitchenham, "Systematic review in software engineering: Where we are and where we should be going," in *Proc. 2nd Int. Workshop Evidential Assessment Softw. Technol.*, Sep. 2012, pp. 1–2.
[63] A. Dresch, D. Lacerda, D. Pacheco Jr., and J. A. V. Antunes, *Design Science Research: Método de Pesquisa Para Avanço da Ciência e Tecnologia*. Porto Alegre, Brazil: Bookman Editora, 2020.
[64] H. Sobrinho, "Organizando o conhecimento sobre técnicas de aprendizagem colaborativas," *Nuevas Ideas em Informatica Educativa*, vol. 12, pp. 152–156, Jan. 2016.
[65] W. Silva, B. Gadelha, I. Steinmacher, and T. Conte, "Towards an open repository for teaching software modeling applying active learning strategies," in *Proc. IEEE/ACM 42nd Int. Conf. Softw. Eng., Softw. Eng. Educ. Training (ICSE-SEET)*, Oct. 2020, pp. 162–172.
[66] A.-P. Correia, "As múltiplas facetas da curadoria de conteúdos digitais," *Revista Docência e Cibercultura*, vol. 2, no. 3, pp. 14–32, 2018.
[67] 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.
[68] L. M. Serrano-Cámara, M. Paredes-Velasco, C.-M. Alcover, and J. Á. Velazquez-Iturbide, "An evaluation of students' motivation in computer-supported collaborative learning of programming concepts," *Comput. Hum. Behav.*, vol. 31, pp. 499–508, Feb. 2014.
[69] A. I. Safana and M. Nat, "Students' perception of a blended learning approach in an African higher institution," *J. Univers. Comput. Sci.*, vol. 25, no. 5, pp. 515–540, 2019.
[70] L. Pollock and M. Jochen, "Making parallel programming accessible to inexperienced programmers through cooperative learning," *ACM SIGCSE Bull.*, vol. 33, no. 1, pp. 224–228, Mar. 2001.
[71] M. Kumar, V. G. Renumol, and S. Murthy, "Flipped classroom strategy to help underachievers in Java programming," in *Proc. Int. Conf. Learn. Teaching Comput. Eng. (LaTICE)*, Apr. 2018, pp. 44–49.
[72] D. Dicheva and A. Hodge, "Active learning through game play in a data structures course," in *Proc. 49th ACM Tech. Symp. Comput. Sci. Educ.*, Feb. 2018, pp. 834–839.
[73] B. Gonçalves, E. Nascimento, E. Monteiro, C. Portela, and S. Oliveira, "Elementos de Gamificação aplicados no ensino-aprendizagem de Programação Web," in *Proc. Anais do Workshop Sobre Educação em Computação (WEI)*, Jul. 2019, pp. 1–10.
[74] A. A. D. Castro Jr., L. M. Cheung, E. J. S. Batista, and A. C. D. Lima, "Uma Análise preliminar da Aplicação do Método 300 em turmas de algoritmos e Programação," in *Proc. Anais do XXIX Workshop sobre Educação em Computação (WEI)*, Jul. 2021, pp. 171–180.

[75] S. C. d. Santos, É. Santana, L. Santana, P. Rossi, L. Cardoso, U. Fernandes, C. Carvalho, and P. Tôrres, "Applying PBL in teaching programming: An experience report," in *Proc. IEEE Frontiers Educ. Conf. (FIE)*, Oct. 2018, pp. 1–8.
[76] N. Avouris, S. Kaxiras, O. Koufopavlou, K. Sgarbas, and P. Stathopoulou, "Teaching introduction to computing through a project-based collaborative learning approach," in *Proc. 14th Panhellenic Conf. Informat.*, Sep. 2010, pp. 237–241.
[77] S. A. Turner, M. A. Pérez-Quiñones, and S. H. Edwards, "Peer review in CS2: Conceptual learning and high-level thinking," *ACM Trans. Comput. Educ.*, vol. 18, no. 3, pp. 1–37, Sep. 2018.
[78] A. Joshi, M. Schmidt, S. Panter, and A. Jain, "Evaluating the benefits of team-based learning in a systems programming class," in *Proc. IEEE Frontiers Educ. Conf. (FIE)*, Oct. 2020, pp. 1–7.
[79] L. N. Gamage, "A bottom-up approach for computer programming education," *J. Comput. Sci. Colleges*, vol. 36, no. 7, pp. 66–75, 2021.
[80] A. Kothiyal, S. Murthy, and S. Iyer, "Think-pair-share in a large CS1 class: Does learning really happen?" in *Proc. Conf. Innov. Technol. Comput. Sci. Educ. (ITiCSE)*, 2014, pp. 51–56.
[81] I. Calderon, W. Silva, and E. Feitosa, "Collabprog: Um repositório colaborativo aberto para apoiar na adoção de metodologias ativas no ensino de programação," in *Anais Estendidos do II Simpósio Brasileiro de Educação em Computação (SBC)*, 2022, pp. 36–39.
[82] C. Mayfield, S. K. Moudgalya, A. Yadav, C. Kussmaul, and H. H. Hu, "POGIL in CS1: Evidence for student learning and belonging," in *Proc. 53rd ACM Tech. Symp. Comput. Sci. Educ.*, Feb. 2022, pp. 439–445.

IVANILSE CALDERON received the bachelor's degree in information systems from the Lutheran University of Brazil (ULBRA), and the M.Sc. degree in geography from the Federal University of Rondônia (UNIR). She is currently pursuing the Ph.D. degree in informatics with the Federal University of Amazonas (UFAM). She is a Professor of basic, technical, and technological education with the Federal Institute of Education, Science, and Technology of Rondônia (IFRO). She is a member of the Research Group on Technology and Education in Computing (GPComp).

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 Pampa (UNIPAMPA) and a member of the Graduate Program in Software Engineering (PPGES-UNIPAMPA). 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 main research interests include computer science education, informatics in education, requirements engineering, software quality, experimental software engineering, information systems, usability and user experience, conversational agents, chatbots, artificial intelligence, and machine learning. He is a member of the LESSE Research Group (Laboratory of Empirical Studies in Software Engineering), 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 the Brazilian Computer Society (SBC). He is also part of the Interest Group on Active Methodologies linked to the Special Commission on Computer Education.

EDUARDO FEITOSA received the bachelor's degree in data processing from the Federal University of Amazonas (UFAM), in 1998, the M.Sc. degree in computer science from the Federal University of Rio Grande do Sul, in 2001, and the Ph.D. degree in computer science from the Federal University of Pernambuco, in 2010. He is an Associate Professor with the Institute of Computing (IComp) and a permanent Professor of the Graduate Program in Informatics (PPGI), UFAM. He is the Coordinator of the Graduate Program in Informatics (PPGI), UFAM. He is one of the leaders of the Research Group on Emerging Technologies and System Security (ETSS). He collaborates with various research groups both nationally and internationally and has coordinated projects with national and international institutions. He is a reviewer of several journals and a member of committees of various conferences. He is a member of the Special Commission on Information and System Security (C.E.Seg.), SBC.