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.

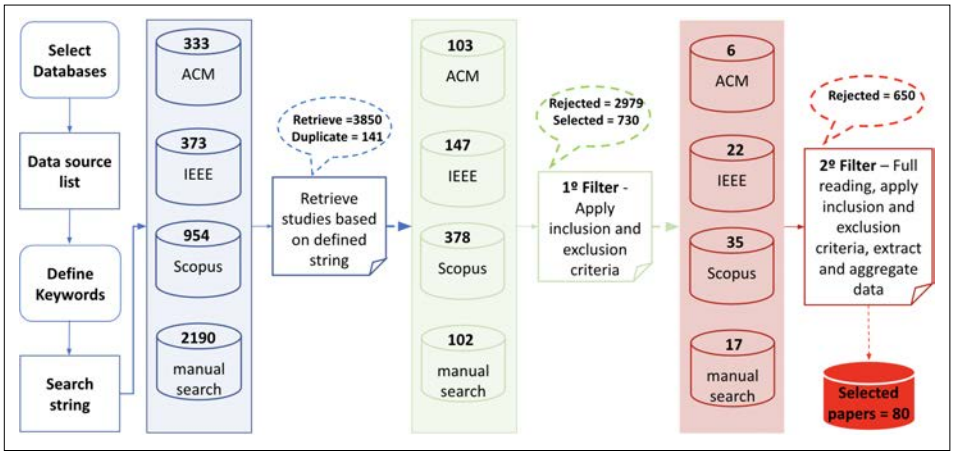

Fig. 1. Results of systematic mapping filters. Source: The authors.

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-

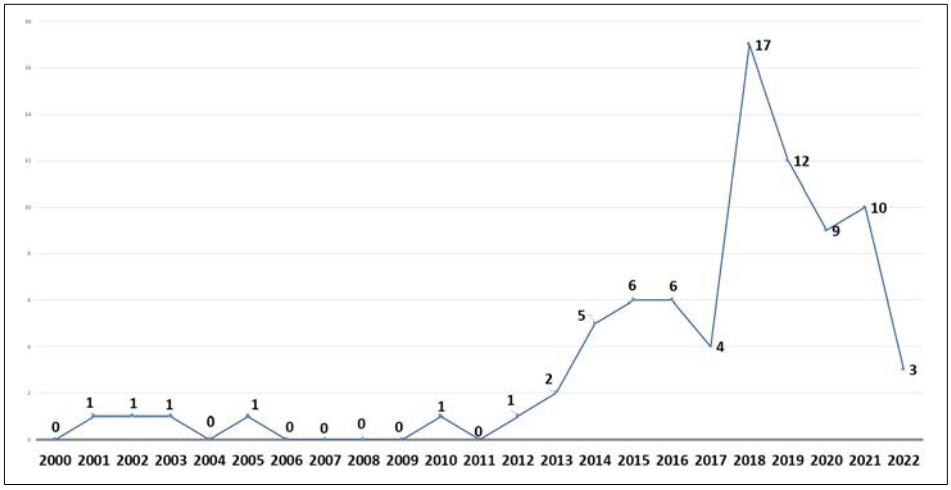

Fig. 2. Publication trend by year. Source: The authors.

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),

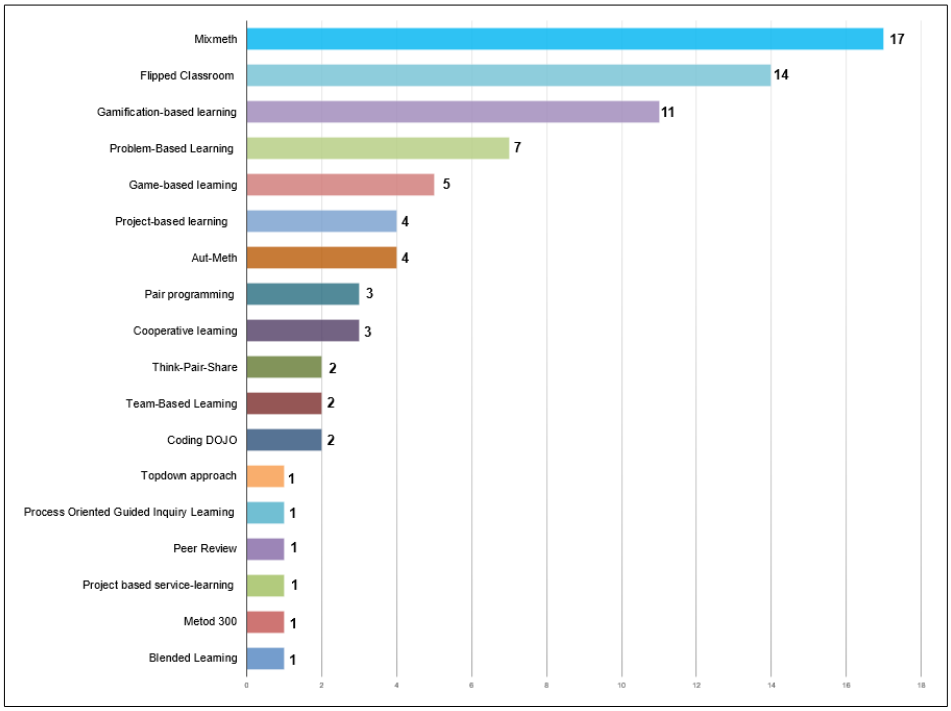

Fig. 3. Types ALMs adopted for teaching programming. Source: The authors.

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

Metho-dology	Technologies	#Publi-cation
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

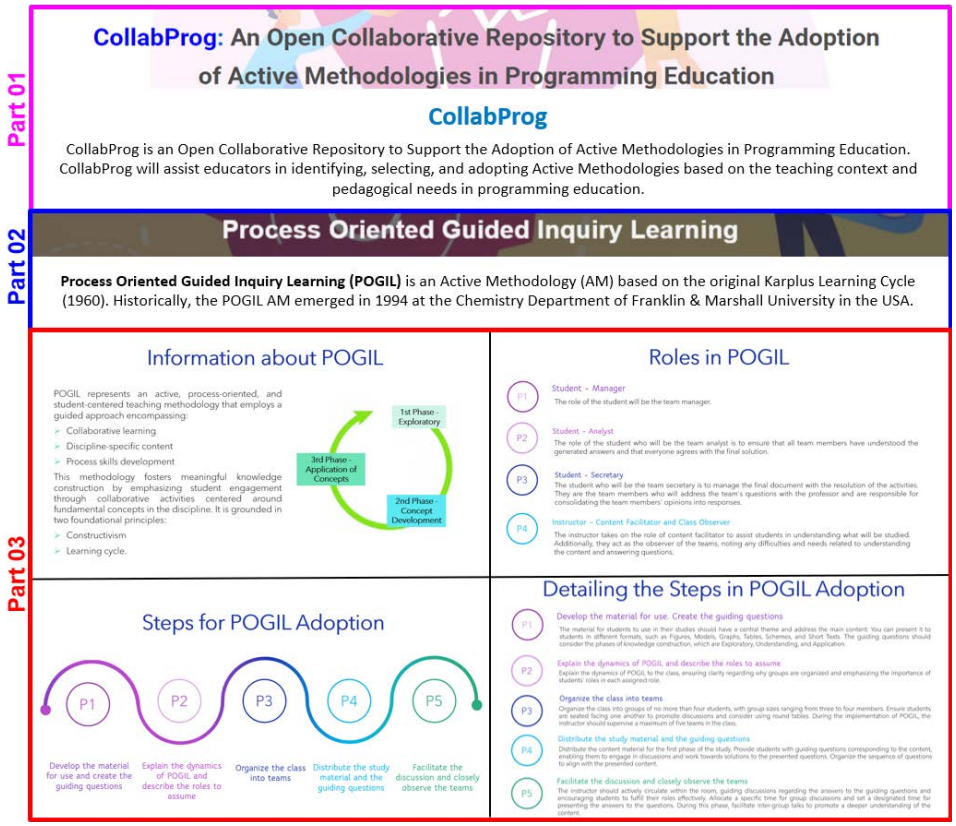

Fig. 4. First version of CollabProg. Source: The authors.

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

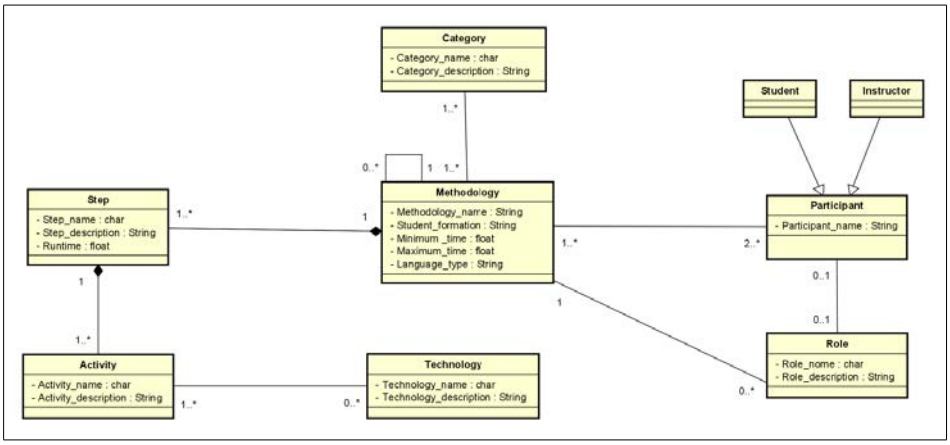

Fig. 5. Conceptual model of CollabProg. Source: The authors.

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)