Theories and Practices: A Rapid Review on Active Methodologies and Learning Theories in Software Engineering Education

Lucas Araújo
IComp, Federal University of Amazonas
Manaus, Brazil
lucas.sarmento@icomp.ufam.edu.br

Eduardo Souto
IComp, Federal University of Amazonas
Manaus, Brazil
esouto@icomp.ufam.edu.br

Ana Carolina Oran
IComp, Federal University of Amazonas
Manaus, Brazil
ana.oran@icomp.ufam.edu.br

Abstract

Software Engineering Education (SEE) has increasingly adopted learning theories and active methodologies to enhance students' engagement and learning outcomes. The diversity of approaches makes it difficult for instructors to select appropriate strategies for their contexts. This Rapid Review brings the main learning theories and active methodologies used in SEE, as well as their practical applications and supporting tools. Based on a structured search in the Scopus database with defined inclusion and exclusion criteria, 41 primary studies were selected through title and abstract screening and full-text analysis. The findings indicate a clear predominance of active methodologies, especially Game-Based Learning, Problem-Based Learning, Flipped Classroom, Gamification, and Project-Based Learning, while the explicit use of learning theories remains limited. A set of tools, such as virtual learning environments, collaborative platforms, and educational games, was identified, highlighting the connection between pedagogical approaches and technological support.

Keywords

Software Engineering Education, Educational Methods, Learning Theories

1 Introduction

Learning theories and active methodologies are educational approaches that have been extensively discussed in the literature, with researchers, scientists, and psychologists seeking to understand and explain how humans learn [42]. Learning theories provide the pedagogical foundations that support the understanding of the learning process, while active methodologies focus on practical strategies that encourage students to actively participate in the construction of their own knowledge [42].

With the advancement of computational systems and emerging technologies, the software industry has become increasingly dependent on highly qualified professionals capable of meeting the quality standards and complexity required in modern software development [46]. Consequently, Software Engineering Education (SEE) has faced the need to improve its educational practices and to provide learning environments that are more engaging, dynamic, and aligned with the demands of the technology sector [24, 36, 45].

Traditional teaching approaches, often centered on passive knowledge transmission, may not adequately prepare students for the collaborative, practical, and problem-solving nature of software engineering. In response, educators and researchers in SEE have increasingly explored the adoption of learning theories and active methodologies to enhance student engagement, critical thinking, and knowledge retention [8, 24]. These approaches aim to place students at the center of the learning process, encouraging reflection, collaboration, and the application of theoretical concepts in practical scenarios.

Several active methodologies and pedagogical approaches have been applied in SEE contexts; however, despite the increasing number of studies in this field, the diversity of approaches and the lack of consolidated evidence make it difficult for educators to identify which theories and methodologies are most commonly applied and how they contribute to the teaching and learning process in software engineering.

This study presents a rapid review aimed at identifying and analyzing the learning theories and active methodologies adopted in Software Engineering Education. It was identified the main learning theories, active methodologies, and supporting tools adopted in Software Engineering Education were identified through this Rapid Review. The findings reveal that active methodologies are considerably more prevalent than the explicit use of learning theories. By synthesizing the existing literature, this work intends to support educators and researchers in understanding current pedagogical trends and in making more informed decisions regarding teaching practices in software engineering courses.

2 Background

This section presents the theoretical background related to learning approaches adopted in educational contexts, with emphasis on Learning Theories and Active Learning methods. First, the section discusses the foundations of Learning Theories and their role in explaining how individuals acquire knowledge and develop skills. Next, it introduces the concept of Active Learning, highlighting educational practices that promote student engagement, participation, and knowledge construction.

2.1 Learning Theories

Learning has been an important subject among psychologists and educators, and its definition is far from being unique among researchers in this field. One of the most common definitions describes learning as a behavioral change, meaning that an individual starts to act differently after receiving instruction or being exposed to experiences that positively or negatively influence their behavior [51]. However, not every behavioral change can be considered learning, since some changes may result from temporary conditions such as fatigue or the ingestion of substances, which do not represent a permanent acquisition of knowledge or skills [31]. Therefore, learning is more accurately defined as a relatively permanent behavioral change resulting from experience, capable of modifying

an individual's disposition and ability to perform certain actions [31].

Over the decades, several scientists, philosophers, and scholars have sought to understand how human beings learn and develop new abilities. Different approaches were employed to investigate the learning process, including observations of children's development, studies conducted in school environments, and behavioral analyses [31, 51]. These investigations contributed to the creation of structured explanations regarding human learning and cognition.

The observations and findings produced by these researchers established the foundations for what became known as Learning Theories (LTs). Learning theories can be understood as organized sets of observations, hypotheses, principles, and assumptions that aim to explain human behavior and the learning process [31]. Throughout history, several theories have emerged, evolved, and, in some cases, been replaced as new scientific evidence became available. This evolution demonstrated that no single theory is capable of fully explaining all aspects involved in human learning [26].

Although learning theories seek to explain how people learn, many of them are primarily descriptive, focusing on interpreting observed behaviors rather than providing practical guidance on how to engage, motivate, or support learners in educational settings [7, 56].

In response to the limitations of traditional teaching methods, especially during the mid-20th century, educational researchers and practitioners began to advocate for approaches that encouraged greater student participation and engagement in the learning process [47]. This movement contributed to the emergence of Active Learning (AL), which emphasizes student-centered educational practices and the active construction of knowledge.

2.2 Active Learning

Before the 21st century, traditional educational approaches were predominantly based on passive learning, where students were expected to listen to lectures, copy information, and reproduce content presented by teachers [47]. In this model, learners had limited opportunities to reflect, argue, question, or critically think about the concepts discussed in the classroom. As a consequence, such approaches could negatively affect student engagement and motivation, leading some learners to lose interest in the subject or even abandon the learning process [47, 60].

One of the main contributors to the foundations of Active Learning (AL) was the philosopher and educator John Dewey. Dewey argued that students learn more effectively when theory and practice are integrated within the educational process [47]. According to his perspective, learning should occur through experience and practice, where teachers and educational materials act as guides during the learning journey, while students themselves become responsible for constructing their own knowledge [60].

Similar to Learning Theories, Active Learning does not have a single universally accepted definition. However, different definitions of AL share the common understanding that students should actively participate in the learning process through activities that stimulate reflection, analysis, communication, reading, writing, collaboration, and problem-solving [7, 60]. Instead of acting solely as passive recipients of information, students are encouraged to interact with the content, their peers, and instructors, contributing to the development of critical thinking, autonomy, and practical skills.

As a result, Active Learning has become an important educational approach in several fields, including Software Engineering Education, where student engagement, collaboration, and practical problem-solving are essential for preparing professionals capable of dealing with real-world challenges.

3 Study Design

A Rapid Review is a streamlined form of literature review designed to provide timely evidence synthesis while maintaining methodological transparency and rigor. It aims to identify, evaluate, and summarize the available research on a specific topic in a more time-efficient manner than traditional systematic reviews [9]. In this study, the Rapid Review approach was adopted to provide an overview of the theories and methods applied in SEE, while ensuring a structured and reproducible research process.

Prior to conducting the review, an exploratory search was carried out to identify key studies and establish an initial understanding of the topic. This preliminary investigation also supported the identification of control articles that guided the review process. Google Scholar was initially used to obtain a broad perspective of the literature and assist in refining the research scope.

The review protocol included the definition of the search string, the development of the data extraction form, and the selection of the digital library used for the search process. Inclusion and exclusion criteria were established to determine the relevance of the retrieved studies. The study selection process was divided into two stages. In the first stage, titles and abstracts were screened according to the predefined criteria. In the second stage, the selected studies underwent full-text reading and data extraction. The protocol adopted in this review was adapted from the guidelines proposed by Kitchenham [29].

3.1 Objective

The objective of this Rapid Review is to identify the main learning theories and educational methods applied in SEE. To support the definition of this objective, the Goal-Question-Metric (GQM) paradigm was adopted [3, 63], resulting in the following goal structure:

- **Analyze:** Scientific studies
- **For the purpose of:** identifying learning theories and educational methods applied in Software Engineering Education
- **With respect to their:** practical application in educational contexts
- **From the point of view of:** researchers and instructors
- **In the context of:** Software Engineering Education

3.2 Research Questions

To achieve the objective of this review, the following research questions (RQs) were defined:

- **RQ1:** "Which learning theories or educational methods are reported in the Software Engineering Education literature?"
- **RQ2:** "How are these learning theories and educational methods applied in Software Engineering Education contexts?"

- **RQ3:** "Which tools, techniques, or approaches are associated with the application of these theories or methods?"

RQ1 aims to identify the learning theories and educational methods discussed in the SEE literature, providing an overview of the current research landscape. Since these theories and methods may be applied in different educational scenarios, **RQ2** investigates how they are implemented in practice within Software Engineering Education contexts. Finally, **RQ3** seeks to identify the tools, techniques, and approaches adopted to support or operationalize these educational practices.

4 Research Strategy

This section describes the research process adopted in this Rapid Review, including the definition of the research scope, publication language, search string construction, selection criteria, and study selection.

Scope: The digital library selected for this study was **Scopus**. As this study follows the Rapid Review methodology, methodological simplifications were intentionally adopted to provide a timely synthesis while maintaining a transparent and reproducible review process. This choice was also motivated by Scopus' broad indexing coverage, which includes publications from major scientific publishers and databases such as ACM, IEEE, Springer, and others, ensuring access to relevant studies in SEE.

Language: The selected studies were limited to publications written in English and Portuguese. English was chosen because it is the predominant language in scientific communication, while Portuguese was included to capture relevant studies published in national conferences and symposiums related to computing and education.

4.1 Research String

The search string was constructed based on the PICOC approach [49], which supports the definition of search strategies in secondary studies. PICOC consists of five elements: Population, Intervention, Comparison, Outcome, and Context. For this review, the elements were defined as follows:

- **Population:** Studies related to learning theories and educational methods in SEE.
- **Intervention:** Application and use of learning theories and educational methods in SEE contexts.
- **Comparison:** Not applicable, since this study aims to characterize the literature rather than compare interventions.
- **Outcome:** Identification of practices, approaches, and strategies for applying learning theories and educational methods in SEE.
- **Context:** Undergraduate Software Engineering Education.

Based on the PICOC elements, groups of related terms were defined to compose the search string. Before its final application, the search string underwent an iterative refinement process. Different combinations of keywords and synonyms were evaluated to improve the balance between sensitivity and specificity, seeking to maximize the retrieval of relevant studies while reducing unrelated results.

Figure 1 presents the final search string adopted in this study. Additionally, during the execution of the search in Scopus, the subject area filter for Computing was applied to reduce unrelated results outside the SEE domain.

```
TITLE-ABS-KEY(("software engineering") AND ("Learning theory" OR "Learning
theories" OR "learning method" OR "learning methods" OR "learning process")
AND ("education" OR "teaching" OR "class" OR "training")) AND ( LIMIT-TO (
SUBJAREA,"COMP" ) )
```

Figure 1: Research String

4.2 Selection Criteria

To identify relevant studies, Inclusion Criteria (IC) and Exclusion Criteria (EC) were established based on the review objective and research questions. These criteria guided the study selection process and are presented in Table 1.

Table 1: Inclusion and exclusion criteria

Inclusion Criteria	Exclusion Criteria
The study mentions learning theories or educational methods in Software Engineering Education (IC1).	The study does not meet any inclusion criterion (EC1).
The study describes the application of a learning theory or educational method in SEE contexts (IC2).	The full text is not available (EC2).
	The study is not written in English or Portuguese (EC3).
	The study focuses on Machine Learning instead of learning theories or educational methods (EC4).

No specific publication time frame was defined for this Rapid Review, aiming to maximize the coverage of the available literature and to provide a broader overview of the learning theories and educational methods applied in SEE.

4.3 Study Selection

The study selection process followed four main steps:

(1) **Exploratory Search:** An initial exploratory search was conducted to provide a foundational understanding of the topic. During this stage, control articles relevant to the research objective were identified and used to support the refinement of the search strategy.
(2) **First Filter:** The first filtering stage consisted of screening the titles and abstracts of the studies retrieved from the Scopus search engine using the defined search string.
(3) **Second Filter:** In the second stage, the studies selected in the previous step underwent full-text reading to assess their relevance according to the established criteria.
(4) **Data Extraction:** Finally, the relevant information from the selected studies was extracted using a predefined extraction form designed to answer the research questions.

4.4 Review Execution

After the planning stage was completed, the execution of the Rapid Review was initiated. To reduce potential bias from the main researcher, the review protocol,including the search string and the criteria used in the first filtering stage, was reviewed and validated by an experienced doctoral researcher. In addition, a paired evaluation was conducted with a sample of 10 randomly selected studies to validate the study selection process.

The selected studies were independently classified according to the inclusion and exclusion criteria based on their titles and abstracts. To measure the level of agreement between reviewers, Cohen's Kappa coefficient was applied. The obtained coefficient was 0.9, indicating a strong agreement between reviewers, since values between 0.61 and 0.80 are already considered substantial agreement [13].

4.4.1 Selecation Execution. The search execution began with the application of the defined search string in the Scopus database, restricted to the Computing subject area. This process initially returned 715 studies. The retrieved data were exported to the Thoth web application to support the study selection process [16]. The tool assisted in identifying duplicate studies, which were manually reviewed by the researchers. After removing duplicates, a total of 709 studies remained for screening.

The first filtering stage consisted of reading the titles and abstracts of the retrieved studies. After this process, 227 studies were selected for the second filtering stage. In the second stage, the full texts of the selected studies were analyzed to verify their eligibility and extract relevant information. At the end of the process, 41 studies were included in the final extraction phase using the predefined extraction forms. The filtering process can be observed in the Figure 2.

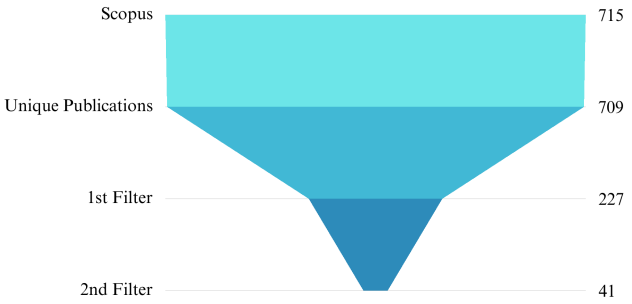

Figure 2: Study selection process

During the selection process, both inclusion criteria were considered throughout the two screening stages, although they were assessed at different levels of evidence. During the first screening stage (title and abstract), IC1 (see Table 1) was more directly observable, as many studies mentioned at least one learning theory or educational method without describing its practical application. Therefore, potentially relevant studies were retained for full-text analysis. During the second screening stage (full-text reading), both criteria were reassessed, with IC2 being fully verified by identifying studies that described the practical application of learning theories or educational methods.

5 Results

This section presents the main findings extracted from the selected studies, including publication distribution over the years, the identified learning theories and educational methods, and the main educational contexts in which these approaches were applied.

5.1 Publications per Year

Figure 3 presents the distribution of publications by year. An increase in the number of studies can be observed between 2020 and 2021, a period that coincides with the COVID-19 pandemic. This growth suggests an increased interest in adopting active learning approaches and innovative educational practices during remote and hybrid teaching scenarios.

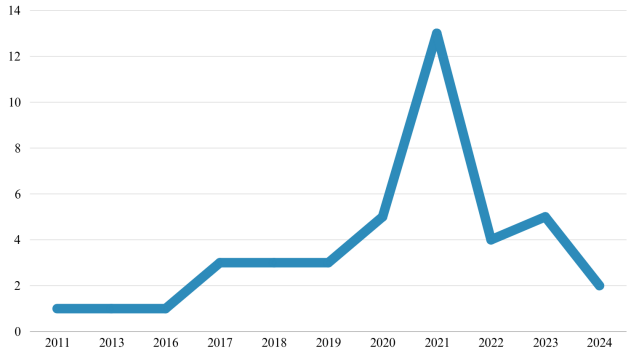

Figure 3: Publications per Year

The increased adoption of active methodologies during this period may reflect the efforts of educators and researchers to promote student engagement and maintain effective learning experiences under the challenges imposed by remote education [40, 44].

5.2 Learning Theories and Educational Methods

To answer **RQ1**, the learning theories and educational methods identified in the selected studies were extracted and cataloged (see Table 2). A total of 18 different approaches were identified in the literature.

Table 2: Learning theories and educational methods identified in the selected studies

Learning Theory / Method	
1. Activity-Oriented Teaching Strategy	10. Game-Based Learning
2. Analogy-Based Learning	11. Gamification
3. Bloom's Taxonomy	12. Kolb's Experiential Learning Cycle
4. Collaborative Learning	13. Lego Serious Play
5. Cooperative Learning Method	14. Narrative Learning
6. Crowd-based Learning	15. Problem-Based Learning
7. Example-Based Learning	16. Project-Based Learning
8. Experiential Learning	17. Self-regulated Learning
9. Flipped Classroom	18. Studio-Based Learning

The learning theories and active methods identified in the selected studies are presented below. Several approaches appeared in

multiple publications, and their frequency distribution is shown in Figure 4.

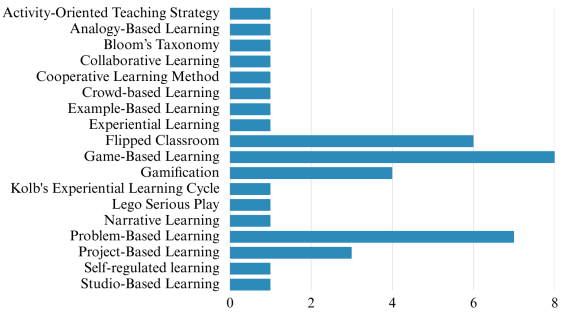

Figure 4: Frequency of learning theories and active methods

The most frequently adopted teaching approaches were Game-Based Learning (GBL), Flipped Classroom, Gamification, Problem-Based Learning (PBL), and Project-Based Learning (PjBL). Initially, this review aimed to identify which learning theories were applied in SEE. However, during the preliminary investigation and the first filtering stage, a substantial number of studies mentioning active methods were identified. Consequently, the final search string was expanded to include active methodologies, allowing the study to cover a broader range of teaching approaches.

At the end of the selection process, the number of active methods identified surpassed the number of learning theories. This distribution is illustrated in Figure 5.

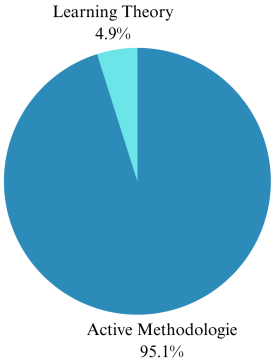

Figure 5: Proportion of learning theories and active methods

Among the 41 selected studies, only two explicitly mentioned the use of learning theories as the foundation of their teaching approach. The remaining studies primarily employed active methods to support the teaching of Software Engineering topics.

5.3 Learning Approaches and Tools

RQ2 investigates how these learning approaches are applied in SEE, while **RQ3** focuses on identifying the tools used alongside them. Since the employed tools are directly related to the implementation of the learning approaches, the findings for both research questions are discussed together in this subsection.

A considerable variety of tools was identified across the selected studies. The objective of this research is not to evaluate or compare these tools, but rather to identify which tools were reported by the authors, regardless of their availability or whether they were used in combination with other resources. Table 3 presents the tools adopted in the analyzed studies.

Table 3: Learning Theories/Methods and Supporting Tools

LT/MA	Tools
Activity-Oriented Teaching Strategy	Google Classroom; Online Docs
Analogy-Based Learning	Slides
Bloom's Taxonomy	Sensor Kinect; Oculus Quest; Emotiv EEG
Collaborative Learning	Facebook; CASE Tool
Cooperative Learning Method	N/A
Crowd-based Learning	StackOverflow; GitHub; Trustie; OSSEAN; CodePedia
Example-Based Learning	Digital Apps
Experiential Learning	JSpin; Pentimeter
Flipped Classroom	Zoom; Canvas; Google Forms; Microsoft Teams; Videos; Slides; Quiz; Moodle; Google Classroom
Game-Based Learning	LEGO® Kits; Unity; VR; Unity VR; OOAD-Game 3D; SoftBook; SimulES-W; TestEG; UbiRE Web Game; LEARN
Gamification	Kahoot; GamifCS; CleanGame
Kolb's Experiential Learning Cycle	Role-playing Games
Lego Serious Play	Lego Kits
Narrative Learning	Moodle; Discussion Forums
Problem-Based Learning	Slack; GitHub; GitHub Projects; Moodle; PBL-coach; BPMN; Amadeus; Google Classroom; Astah UML; Level-Learn
Project-Based Learning	GitHub; Socrative Quiz; Facebook Messenger; Google Drive; Google Docs
Self-regulated Learning	Moodle
Studio-Based Learning	Whiteboards; TVs; GitHub; Google Docs; SVN

Some learning approaches appear in only one study, so their use is more specific and detailed to the context in which the application was made. With two or more studies, it was necessary to generalize the description highlighting the similarity between them.

The **Activity-Oriented Teaching Strategy** (AOTS) was employed to foster active learning in students by shifting the instructional focus from passive content delivery to hands-on, student-centered activities. The strategy structured the learning process around carefully designed activities that required learners to actively apply theoretical concepts through problem-solving tasks, collaborative discussions, and project-based work. Students engaged in practical exercises aligned with real-world scenarios, which encouraged critical thinking, peer interaction, and continuous reflection on their learning process. The instructor acted primarily as a facilitator, guiding discussions, providing feedback, and supporting students as they constructed knowledge through experience. Overall, AOTS enhanced student engagement and understanding by integrating theory with practice in an iterative and participatory learning environment [30].

The **Analogy-based learning** was used to facilitate students' understanding of abstract concepts by relating them to familiar situations from everyday life. In this approach, complex topics were explained through carefully designed analogies that mapped key elements of the target concept to well-known domains, helping students build new knowledge upon their prior experiences. The analogies were integrated into classroom explanations and discussions, serving as a cognitive bridge between theory and practice and supporting student engagement and conceptual comprehension. [53].

Bloom's Taxonomy is a learning theory used as a guiding framework to structure learning activities across progressive cognitive levels, aligning instructional objectives with project-based learning tasks. Each stage of the project was intentionally mapped to a specific cognitive level, enabling students to gradually deepen their understanding while engaging in design, implementation, and evaluation activities. This structured alignment supported the development of higher-order thinking skills and promoted a coherent progression from theoretical knowledge to practical application [50].

The **Collaborative learning** was implemented through the creation of a Facebook group in which students actively engaged in didactic activities related to software analysis and modeling. The activities required students to individually produce artifacts, such as diagrams, and subsequently share their work in the online environment, fostering peer discussion, feedback exchange, and collective knowledge construction under the guidance of the instructor [15].

The **Cooperative Learning Method** was applied by organizing students into small groups, but each student was individually responsible for a specific part of the task, which was later integrated into a collective group output. During the reading for extraction, it was not specified which tool was used during the activity, but the students were involved in cooperative strategies such as the Jigsaw method and Group Investigation, promoting peer teaching, presentations, shared problem analysis, and joint construction of software engineering artifacts under instructor facilitation. [4]

In the **Crowd-based learning** approach, students interacted with question-and-answer platforms to obtain solutions to development problems and accessed open-source software repositories to study, reuse, and adapt existing artifacts. These practices supported learning through community-provided expertise, imitation of high-quality software artifacts, and reuse of open-source components to accomplish course projects[37].

Instructors used the **Example-Based Learning** with worked examples to teach class diagram modeling. Both correct and erroneous examples were presented to illustrate modeling concepts, common mistakes, and appropriate solutions so the students could see what they should or should not do. A support portal guided instructors in planning and conducting the lessons, assisting in the selection of examples and in structuring classroom activities focused on observation, analysis, and correction of class diagram models. [6]

On the **Experiential Learning** approach, instructional activities were organized according to Kolb's learning cycle, integrating concrete experience, reflection, conceptualization, and active experimentation. Students engaged in hands-on tasks such as poster creation, group presentations, and the implementation of model-checking problems using dedicated tools. These activities were supported by guided reflection and laboratory assignments, enabling learners to reinforce theoretical concepts through experiential practice [62].

Flipped Classroom approach enables instructors to combine pre-class study with active, student-centered learning during synchronous sessions. Common practices include the use of short instructional videos, study guides, and online quizzes to support individual preparation, followed by in-class or virtual activities such as group discussions, think-pair-share, role-playing, games, and collaborative problem solving in small teams. In online or hybrid settings, these activities can be adapted through videoconferencing tools with breakout rooms and learning platforms that support micro-projects, peer review, reflection, and immediate feedback [22, 23, 33, 39, 44, 61].

Game-Based Learning (GBL) offers opportunities to teach Software Engineering concepts through immersive and interactive game environments. Possible applications include role-based simulations in which students assume project roles (manager, developer or tester). Digital and tabletop games may embed questions, scoring systems, branching narratives, and automated feedback to guide learning, while in-game scenarios can emulate realistic project contexts, support requirements elicitation, architectural decision making, and post-game reflection through performance reports and guided classroom discussions [27, 32, 35, 40, 43, 57–59].

Gamification can enhance student engagement by integrating game elements into both synchronous and asynchronous learning activities. Typical strategies include the use of missions and challenges aligned with course content, supported by points, badges, rankings, and multiple leaderboards. Gamified activities may involve tasks such as evaluating code snippets, identifying code smells, and submitting solutions for collective feedback, with automated scoring, hints, and progress tracking used to support motivation, participation, and formative assessment[18, 20, 38, 41].

During the analysis of the studies, some misunderstandings by the authors were observed on the difference between GBL and Gamification, as both use games as their basis. GBL uses **games as tools** during learning activities, and Gamification utilizes **game elements** to engage students [28]. Some authors pointed out that Gamification was used to build their activity, but reading the dynamics' description, it's possible to observe that the activity is a GBL approach.

Kolb's Experiential Learning Cycle is a learning theory that frames learning as a continuous process in which knowledge is constructed through experience and reflection. In educational practice, the cycle can be applied by designing activities that guide students through concrete experiences, followed by reflective observation, abstraction of concepts, and active experimentation. For example, learners may engage in simulated or role-playing game (RPG) activities to experience realistic problem contexts, reflect on their actions and outcomes, connect these reflections with theoretical principles, and subsequently apply the acquired understanding to new or more complex tasks, reinforcing learning through iterative practice [64].

LEGO® Serious Play is as experiential and participatory learning methodology in which students use LEGO bricks to externalize ideas, explore concepts, and collaboratively construct shared representations of software engineering knowledge. Through metaphorical modeling and iterative building, students engaged actively with abstract concepts, enhanced teamwork and communication skills, and connected theoretical knowledge with practical understanding [34].

Through the use of structured storytelling, **Narrative Learning** situates educational content within simulated professional contexts. Learners engage with fictional characters, scenarios, and unfolding challenges that frame learning tasks and guide decision making. This approach supports immersion, contextualization of technical

concepts, and metacognitive activities such as reflection and self-monitoring, helping students relate abstract knowledge to realistic problem-solving situations [25].

Problem-Based Learning (PBL) can be implemented by centering instruction on authentic, real-world problems drawn from Software Engineering practice. Typical applications involve presenting realistic scenarios or client-defined problems that require students to work individually or in teams to analyze requirements, conduct research, and propose viable solutions. Learning activities may include collaborative documentation, iterative development of artifacts, tutor-guided sessions, laboratory work with assigned roles, and regular presentations of results [2, 5, 11, 14, 19, 21, 55].

A course can be developed around **Project-Based Learning** (PjBL) by the development of a complete software project. Instruction may interweave active learning methods across theoretical classes, seminars, and laboratory sessions, with design thinking principles embedded in assessments to support problem framing and solution ideation. Collaborative digital tools, such as messaging platforms and shared document repositories, can be used to facilitate communication, coordination, and collaborative requirements elicitation throughout the project lifecycle [1, 10, 54].

Self-Regulated Learning was applied through integration of activities that encouraged students to plan, monitor, and reflect on their learning process. Instructional strategies included periodic reflective tasks and cognitive challenges, implemented as guided prompts and quizzes, which required learners to self-assess their progress, identify difficulties, and define subsequent learning actions. These activities can be embedded within an online learning environment and complemented by formative feedback, fostering learners' autonomy, self-awareness, and continuous regulation of cognitive, motivational, and behavioral aspects of learning [48].

In the **Studio-Based Learning**, the students are engaged in long-term and authentic software projects that simulate professional development environments. Learning activities were organized around collaborative studio sessions in which students worked in small teams on realistic projects, iteratively developing requirements, design, implementation, testing, and documentation. The instructor acted primarily as a coach, providing continuous critique, mentoring, and short just-in-time guidance but never interfering deeply in the students' activities. [52]

6 Threats to Validity

This work addresses a Rapid Review that brought the scenario of learning theory and active methods. Even though the research where adapted from the guidelines [29] for this type of study, one threat to validity that arose is the researcher's bias, which was mitigated by the revision of a doctoral teacher and the Cohen's Kappa test execution [13].

Another potential threat to the validity of this review is the use of a single digital library (Scopus) as the source for identifying primary studies. Although Scopus provides broad coverage and indexes publications from major publishers in Software Engineering and Computing, relying on a single database may have resulted in the omission of relevant studies that are not indexed by Scopus.

7 Discussions

A good number of learning approaches that are used in practices of SEE in the real world where identified, it possible to see that LT and AM can be used in different contexts and adapted to a large number of scenarios, and beyond this, there is a large amount of learning approaches spread in the literature, which makes their study and research a hard-working activity [12]. Not all university teachers have a training on educational bases that can improve their teaching activity, and with all the existing learning approaches, it may not be possible for them to acquire knowledge of all the approaches and scenarios . Given this scenario, one way to help software engineering teachers is to build a form to recommend learning approaches to help teachers in their teaching activities.

The adoption of learning theories and pedagogical methods in Software Engineering education remains challenging for many instructors. At the same time, the widespread availability of Artificial Intelligence (AI) has introduced new educational challenges, as some students may misuse these tools as substitutes for critical thinking and problem-solving. Nevertheless, AI should not be viewed as a threat, but rather as a pedagogical assistant that can support both teaching and learning when used responsibly. As with previous technological advances, such as the adoption of computers in education, initial skepticism is natural; however, AI has the potential to become an important resource for improving educational practices [17].

8 Conclusion and Future Work

This Rapid Review investigated the use of learning theories and educational methods in Software Engineering Education by analyzing 41 primary studies selected through a structured review process. The results showed that active methodologies are considerably more prevalent than explicit learning theories, with Game-Based Learning, Problem-Based Learning, Flipped Classroom, Gamification, and Project-Based Learning being the most frequently adopted approaches. In addition, the review identified a wide range of supporting tools, including virtual learning environments, collaborative platforms, educational games, and software development tools, demonstrating how pedagogical approaches are operationalized in educational practice. These findings provide an overview of the current state of educational approaches in SEE and can support instructors and researchers in selecting teaching strategies that better fit their educational contexts.

As future work, this research can be extended by expanding the search sources beyond the Scopus database to include other relevant digital libraries and indexing platforms, or executing a backward and forward snowballing from the selected primary studies. The inclusion of additional databases may increase the coverage of studies and provide a broader perspective on the application of learning theories and educational methods in Software Engineering Education.

Furthermore, future studies may refine and expand the search string, include additional educational approaches, and investigate new publication venues related to computing education and pedagogy.

Artifact Availability

The artifacts and supporting materials used in this study are publicly available in a online repository and can be accessed through the following DOI: https://doi.org/10.6084/m9.figshare.33023468

Acknowledgements

We would like to thank the financial support granted by CNPq 406506/2025-6; CAPES- Financing Code 001; Amazonas State Research Support Foundation - FAPEAM - through POSGRAD 2026-2027. We also knowledge using ChatGPT tools to support the translation, and rewriting to improve text clarity.

References

[1] LF Al-Qora'n. 2021. Social RE-PBL: An approach for teaching requirements engineering using PBL, SNSs, and cloud storages and file-sharing services. *Int. J. Inf. Educ. Technol* 11, 7 (2021), 342–347.

[2] Abir AlSideiri, Ragad M Tawafak, Sohail Iqbal Malik, Ghaliya Alfarsi, Baidaa Hamza Khudayer, and Roy Mathew. 2023. Examining the Impact of Software Engineering Problem-Based Learning on Student Performance. In *2023 24th International Arab Conference on Information Technology (ACIT)*. IEEE, 1–6.

[3] V.R. Basili and H.D. Rombach. 1988. The TAME project: towards improvement-oriented software environments. *IEEE Transactions on Software Engineering* 14, 6 (1988), 758–773. doi:10.1109/32.6156

[4] Shuib Basri, Aliza Sarlan, Norshakirah A Aziz, and Ahmad Syahir Ahmad Zahiri. 2017. Teaching software engineering course with cooperative learning method: a pilot study. In *2017 7th World Engineering Education Forum (WEEF)*. IEEE, 251–256.

[5] Bruno Bessa and Simone Santos. 2017. A Virtual Environment for Problem-Based Learning in Software Engineering Education.. In *SEKE*. 535–540.

[6] Tiago Piperno Bonetti, Matheus Molina Dias, Williamson Silva, and Thelma Elita Colanzi. 2023. Students' Perception of Example-Based Learning in Software Modeling Education. In *Proceedings of the XXXVII Brazilian Symposium on Software Engineering*. 67–76.

[7] Cynthia Brame. 2016. Active learning. *Vanderbilt University Center for Teaching* (2016), 1–6.

[8] Ivanilse Calderon, Williamson Silva, and Eduardo Feitosa. 2022. CollabProg: Um Repositório Colaborativo Aberto para Apoiar na Adoção de Metodologias Ativas no Ensino de Programação. In *Simpósio Brasileiro de Educação em Computação (EDUCOMP)*. SBC, 36–39.

[9] Bruno Cartaxo, Gustavo Pinto, and Sergio Soares. 2018. The role of rapid reviews in supporting decision-making in software engineering practice. In *Proceedings of the 22nd International Conference on Evaluation and Assessment in Software Engineering 2018*. 24–34.

[10] Edgar Ceh-Varela, Carlos Canto-Bonilla, and Dhimitraq Duni. 2023. Application of Project-Based Learning to a Software Engineering course in a hybrid class environment. *Information and Software Technology* 158 (2023), 107189.

[11] Alex Q Chen, Indriyati Atmosukarto, and Serena Hui Lin Goh. 2023. Transforming Student Learning through Industry-Driven Software Development Projects. In *2023 IEEE Frontiers in Education Conference (FIE)*. IEEE, 1–9.

[12] Sridhar Chimalakonda and Kesav V. Nori. 2011. Can we make software engineering education better by applying learning theories?. In *2011 24th IEEE-CS Conference on Software Engineering Education and Training (CSEET)*. 561–561. doi:10.1109/CSEET.2011.5876157

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

[14] Thelma Colanzi, Leandro Silva, Andressa Medeiros, Paulo Gonçalves, Eniuce Souza, Douglas Farias, and Greicy Amaral. 2023. Practicing the Extension in Software Engineering Education: an Experience Report. In *Proceedings of the XXXVII Brazilian Symposium on Software Engineering*. 514–523.

[15] Dunia María Colomé Cedeño, Arlenys Palmero Ortega, Ailec Granda Dihigo, and Taire Faife Rodríguez. 2020. The collaborative learning of the analysis and modeling of software with the use of Facebook. In *International Conference on Remote Engineering and Virtual Instrumentation*. Springer, 718–728.

[16] Diego Comis and Elder de Macedo Rodrigues. 2025. Thoth 2.0: Advancing an RSL Tool for Enhanced Snowballing Support. In *Simpósio Brasileiro de Engenharia de Software (SBES)*. SBC, 1010–1016.

[17] Marian Daun and Jennifer Brings. 2023. How ChatGPT will change software engineering education. In *Proceedings of the 2023 Conference on Innovation and Technology in Computer Science Education V. 1*. 110–116.

[18] Vitor de Souza Castro and Sandro Ronaldo Bezerra Oliveira. 2022. An Analysis of Application the Kahoot! Tool in a Gamified Approach to Face-to-face and Emergency Remote Teaching and Learning of Software Engineering. In *2022 IEEE Frontiers in Education Conference (FIE)*. IEEE, 1–8.

[19] Luiz Eduardo Guarino de Vasconcelos, Leandro Guarino de Vasconcelos, Leonardo B Oliveira, Gabriel Guimarães, and Felipe Ayres. 2018. Gamification applied in the teaching of agile scrum methodology. In *Information Technology-New Generations: 15th International Conference on Information Technology*. Springer, 207–212.

[20] Hoyama Maria dos Santos, Vinicius HS Durelli, Maurício Souza, Eduardo Figueiredo, Lucas Timoteo da Silva, and Rafael S Durelli. 2019. Cleangame: Gamifying the identification of code smells. In *Proceedings of the XXXIII Brazilian Symposium on Software Engineering*. 437–446.

[21] Simone C. dos Santos, Ana Cláudia Monte, and Ariane Rodrigues. 2013. A PBL approach to process management applied to Software Engineering education. In *2013 IEEE Frontiers in Education Conference (FIE)*. 741–747. doi:10.1109/FIE.2013.6684925

[22] El-Sayed M El-Alfy. 2020. Using mashup and web 2.0 to foster inquiry-based flipped classroom in online teaching of technical curriculum: A case study. In *2020 Sixth International Conference on e-Learning (econf)*. IEEE, 28–34.

[23] Isaac Souza Elgrably and Sandro Ronaldo Bezerra Oliveira. 2022. Using flipped classroom to promote active learning and engagement in a Software Testing subject remotely during the COVID-19 pandemic. In *2022 IEEE Frontiers in Education Conference (FIE)*. IEEE, 1–6.

[24] Víctor M. Flores Fonseca and Jesica Gómez. 2017. Applying Active Methodologies for Teaching Software Engineering in Computer Engineering. *IEEE Revista Iberoamericana de Tecnologias del Aprendizaje* 12, 4 (2017), 182–190. doi:10.1109/RITA.2017.2778358

[25] Mario Madureira Fontes, Daniela Pedrosa, Tânia Araújo, Ceres Morais, Aline Costa, José Cravino, and Leonel Morgado. 2021. Narrative-Driven Immersion and Students' Perceptions in an Online Software Programming Course. In *2021 7th International Conference of the Immersive Learning Research Network (iLRN)*. IEEE, 1–8.

[26] Knud Illeris. 2015. *Teorias contemporâneas da aprendizagem*. Penso Editora.

[27] Irum Inayat, Zubaria Inayat, and Rooh Ul Amin. 2016. Teaching and learning object-oriented analysis and design with 3D game. In *2016 International Conference on Frontiers of Information Technology (FIT)*. IEEE, 46–51.

[28] Dimitrios N Karagiorgas and Shari Niemann. 2017. Gamification and game-based learning. *Journal of Educational Technology Systems* 45, 4 (2017), 499–519.

[29] Barbara Kitchenham, Stuart Charters, et al. 2007. Guidelines for performing systematic literature reviews in software engineering. (2007).

[30] Jeevamol Joy Kochumarangolil and VG Renumol. 2018. Activity oriented teaching strategy for software engineering course: An experience report. *Journal of Information Technology Education. Innovations in Practice* 17 (2018), 181.

[31] Guy R Lefrançois. 2016. Teorias da aprendizagem: o que o professor disse. *Tradução Solange A. Visconte. São Paulo: Cengage Learning* (2016).

[32] Thaiana Lima, Beatriz Campos, Rodrigo Santos, and Claudia Werner. 2012. UbiRE: A game for teaching requirements in the context of ubiquitous systems. In *2012 XXXVIII Conferencia Latinoamericana En Informatica (CLEI)*. IEEE, 1–10.

[33] Yen-Ting Lin. 2021. Effects of flipped learning approaches on students' learning performance in software engineering education. *Sustainability* 13, 17 (2021), 9849.

[34] Daniel Lopez-Fernandez, Aldo Gordillo, Fernando Ortega, Agustín Yagüe, and Edmundo Tovar. 2021. LEGO® serious play in software engineering education. *IEEE Access* 9 (2021), 103120–103131.

[35] Daniel López-Fernández, Edmundo Tovar, Aldo Gordillo, Joaquín Gayoso-Cabada, Carlos Badenes-Olmedo, and Andrea Cimmino. 2024. Comparing a LEGO® serious play activity with a traditional lecture in software engineering education. *IEEE Access* 12 (2024), 74045–74053.

[36] Phelipe Silva Malheiros, Rayfran Rocha Lima, and Ana Carolina Oran. 2024. Impact of Generative AI Technologies on Software Development Professionals' Perceptions of Job Security. In *Proceedings of the XXIII Brazilian Symposium on Software Quality*. 169–178.

[37] Xinjun Mao, Yao Lu, and Yi Yang. 2020. Exploiting Crowd-based Learning Method in Software Engineering Course. In *2020 15th International Conference on Computer Science & Education (ICCSE)*. IEEE, 39–44.

[38] Matheus Serrão Marinato, Socorro Vânia Lourenço Alves, and Enoque Calvino Melo Alves. 2020. Analysis of the effects of the use of Gamification as a teaching strategy in disciplines related to the area of Software Engineering. In *2020 XV Conferencia Latinoamericana de Tecnologias de Aprendizaje (LACLO)*. IEEE, 1–10.

[39] Bruce R Maxim, Thomas Limbaugh, and Jeffrey J Yackley. 2021. Student engagement in an online software engineering course. In *2021 IEEE Frontiers in Education Conference (FIE)*. IEEE, 1–9.

[40] Jesus Mayor and Daniel López-Fernández. 2021. Scrum vr: Virtual reality serious video game to learn scrum. *Applied Sciences* 11, 19 (2021), 9015.

[41] Qing Mi, Jacky Keung, Xiupei Mei, Yan Xiao, and WK Chan. 2018. A gamification technique for motivating students to learn code readability in software engineering. In *2018 International Symposium on Educational Technology (ISET)*. IEEE, 250–254.

[42] Emily Oh Navarro. 2011. On the role of learning theories in furthering software engineering education. In *Instructional design: Concepts, methodologies, tools and applications*. IGI Global Scientific Publishing, 1645–1666.

[43] Bruno Oliveira, Paulo Afonso, and Heitor Costa. 2016. Testeg—a computational game for teaching of software testing. In *2016 35th International Conference of the Chilean Computer Science Society (SCCC)*. IEEE, 1–10.

[44] Mayara Olivindo, Necio Veras, Windson Viana, Mariela Cortés, and Lincoln Rocha. 2021. Gamifying flipped classes: An experience report in software engineering remote teaching. In *Proceedings of the XXXV Brazilian Symposium on Software Engineering*. 143–152.

[45] Sofia Ouhbi, Ali Idri, José Luis Fernández-Alemán, and Ambrosio Toval. 2015. Requirements engineering education: a systematic mapping study. *Requirements Engineering* 20, 2 (2015), 119–138.

[46] Sofia Ouhbi and Nuno Pombo. 2020. Software Engineering Education: Challenges and Perspectives. In *2020 IEEE Global Engineering Education Conference (EDUCON)*. 202–209. doi:10.1109/EDUCON45650.2020.9125353

[47] Marilyn Page. 1990. Active Learning: Historical and Contemporary Perspectives. (1990).

[48] Daniela Pedrosa, Mario Madureira Fontes, Tânia Araújo, Ceres Morais, Teresa Bettencourt, Pedro Duarte Pestana, Leonel Morgado, and José Cravino. 2021. Metacognitive challenges to support self-reflection of students in online Software Engineering Education. In *2021 4th International Conference of the Portuguese Society for Engineering Education (CISPEE)*. IEEE, 1–10.

[49] Mark Petticrew and Helen Roberts. 2008. *Systematic reviews in the social sciences: A practical guide*. John Wiley & Sons.

[50] Montri Phothisonothai, Korawit Orkphol, Natthapon Pannurat, Apirath Limmanee, Warayost Lamaisri, and Chaowat Euafua. 2024. Towards Integrating Physiological Data in UI/UX Design Learning and Engineering Education. In *2024 Tenth International Conference on Communications and Electronics (ICCE)*. IEEE, 573–576.

[51] Julio J Ramirez and Matthew H Olson. 2020. *An introduction to theories of learning*. Routledge.

[52] Daniela Rosca. 2018. Acquiring professional software engineering skills through studio-based learning. In *2018 17th International Conference on Information Technology Based Higher Education and Training (ITHET)*. IEEE, 1–6.

[53] Pawan Saxena, Sanjay K Singh, and Gopal Gupta. 2022. Teaching Complex Software Engineering Concepts through Analogies. In *2022 IEEE Frontiers in Education Conference (FIE)*. IEEE, 1–9.

[54] Camelia Serban and Andreea Vescan. 2020. Towards an Evaluation Process around Active Learning based Methods.. In *FIE*. 1–7.

[55] Maurício Serrano, Milene Serrano, and André Barros de Sales. 2021. Project-Based Learning Focused on Professional Skills: An Approach Applied on Human-Computer Interaction and Software Requirements Under-Graduation Courses. In *Iberoamerican Workshop on Human-Computer Interaction*. Springer, 150–163.

[56] Aliya Sikandar. 2016. John Dewey and his philosophy of education. *Journal of education and Educational Development* (2016).

[57] Lyrene Fernandes Silva and Wallace Carvalho Jacome. 2017. SoftBook: Software Development as an Adventure. *IEEE Latin America Transactions* 15, 6 (2017), 1205–1211.

[58] Tamires AS Sousa and Anna BS Marques. 2020. Learn board game: A game for teaching software architecture created through design science research. In *Proceedings of the XXXIV Brazilian Symposium on Software Engineering*. 834–843.

[59] Elizabeth Suescún Monsalve, Mauricio Toro, Raúl Mazo, David Velasquez, Paola Vallejo, Juan F Cardona, Rafael Rincón, Vera Maria Werneck, and Julio Cesar Sampaio do Prado Leite. 2018. SimulES-W: A collaborative game to improve software engineering teaching. *Computación y Sistemas* 22, 3 (2018), 953–983.

[60] Shannon Vanhorn, Susan M Ward, Kimberly M Weismann, Heather Crandall, Jonna Reule, and Robert Leonard. 2019. Exploring active learning theories, practices, and contexts. *Communication research trends* 38, 3 (2019), 1.

[61] Simona Vasilache. 2022. More Than Just Listening: Active Learning in Face-to-Face and Online Multicultural Software Engineering Classes. In *2022 IEEE Frontiers in Education Conference (FIE)*. IEEE, 1–4.

[62] Andreea Vescan and Camelia Serban. 2020. Facilitating model checking learning through experiential learning. In *Proceedings of the 2nd ACM SIGSOFT International Workshop on Education through Advanced Software Engineering and Artificial Intelligence*. 13–19.

[63] Claes Wohlin, Per Runeson, Martin Höst, Magnus C Ohlsson, Björn Regnell, Anders Wesslén, et al. 2012. *Experimentation in software engineering*. Vol. 236. Springer.

[64] Wen-Hsiung Wu, Wen-Cheng Yan, Hao-Yun Kao, Wei-Yang Wang, and Yen-Chun Jim Wu. 2016. Integration of RPG use and ELC foundation to examine students' learning for practice. *Computers in Human Behavior* 55 (2016), 1179–1184.