Journal of
Information Systems Engineering
and Business Intelligence

Vol.9, No.2, October 2023
Available online at: http://e-journal.unair.ac.id/index.php/JISEBI

A Systematic Literature Review of Student Assessment Framework in Software Engineering Courses

Reza Fauzan [1)*], Daniel Siahaan [2)], Mirotus Solekhah [3)], Vriza Wahyu Saputra [4)], Aditya Eka Bagaskara [5)], Muhammad Ihsan Karimi [6)]

[1)] *Politeknik Negeri Banjarmasin, Banjarmasin, Indonesia*
[1)]reza.fauzan@poliban.ac.id

[2)3)4)5)] *Institut Teknologi Sepuluh Nopember, Surabaya, Indonesia*
[2)]daniel@if.its.ac.id, [3)]6025211025@mhs.its.ac.id, [4)]6025211022@mhs.its.ac.id, [5)]6025211023@mhs.its.ac.id

[6)] *Dassault Systemés, Berlin, Germany*
[6)]karimikarimi@hotmail.de

Abstract

Background: Software engineering are courses comprising various project types, including simple assignments completed in supervised settings and more complex tasks undertaken independently by students, without the oversight of a constant teacher or lab assistant. The imperative need arises for a comprehensive assessment framework to validate the fulfillment of learning objectives and facilitate the measurement of student outcomes, particularly in computer science and software engineering. This leads to the delineation of an appropriate assessment structure and pattern.
Objective: This study aimed to acquire the expertise required for assessing student performance in computer science and software engineering courses.
Methods: A comprehensive literature review spanning from 2012 to October 2021 was conducted, resulting in the identification of 20 papers addressing the assessment framework in software engineering and computer science courses. Specific inclusion and exclusion criteria were meticulously applied in two rounds of assessment to identify the most pertinent studies for this investigation.
Results: The results showed multiple methods for assessing software engineering and computer science courses, including the Assessment Matrix, Automatic Assessment, CDIO, Cooperative Thinking, formative and summative assessment, Game, Generative Learning Robot, NIMSAD, SECAT, Self-assessment and Peer-assessment, SonarQube Tools, WRENCH, and SEP-CyLE.
Conclusion: The evaluation framework for software engineering and computer science courses required further refinement, ultimately leading to the selection of the most suitable technique, known as learning framework.

Keywords: Computer science course, Software engineering course, Student assessment, Systematic literature review

Article history: Received 16 May 2023, first decision 16 June 2023, accepted 15 August 2023, available online 28 October 2023

I. INTRODUCTION

The primary goal of every institution is to ensure high-quality education, and to achieve this objective, each institution develops and offers excellent academic programs [1], [2]. Quality standards were determined by program outcomes, assessments, metrics, and comparisons to meet target benchmarks and the evolving requirements of contemporary society [3]. Moreover, the choice of a school or program significantly influences the lives of individuals, showcasing the importance of informed decisions for parents and students to make wise choices [4], [5].

Programming assignments serve as a common method for computer science instructors to evaluate course objectives [6]–[8]. Computer science courses typically entail various assignments, including smaller tasks, which may be completed in supervised environments such as labs or classrooms, or more substantial assignments requiring students to develop solutions without constant instructor supervision. However, evaluating assignments alone cannot determine when students have successfully mastered the learning objectives [9]–[11]. It is crucial for students to learn

* Corresponding author

ISSN 2443-2555 (online) 2598-6333 (print)
doi: http://dx.doi.org/10.20473/jisebi.9.2.264-275

programming in early computing courses, but a more insightful assessment framework is needed to gauge the achievement of learning objectives [12]–[15].

Before implementing this framework, establishing a culture of assessment is essential. The Practical Assessment of Computer Science Capstone Projects and Student Outcomes rubrics were utilized, simplifying the selection process for assessors [16]–[18]. It is crucial to acknowledge that the implementation of this system comes with some practical challenges. Evaluators need to be trained in using the rubrics [19], [20], and they should be dedicated to assessing all measurable artifacts, including essays, reports, and presentations. Time constraints in the review process pose a significant obstacle [21]–[23].

According to Areekkuzhiyil [24], the course assessment comprises three primary components, namely test quality, domain dependencies, and measurement. These factors significantly influence how effectively students learn, emphasizing the importance of assessment in education. The study focuses on software engineering courses [25], [26], which play a critical role in modern software development, bridging the gap between business and technology. Software engineers, guided by principles and methods, create dependable, secure, and effective software solutions. The evaluation of software design is a specific example within the learning process [27], [28]. As commonly known, different designs can yield similar results, such as a scenario in which a teacher needs to evaluate numerous students with diverse responses. This situation is likely to compromise the objectivity and evaluation standards, hence, an assessment framework is crucial in a software engineering class.

An assessment framework is essential for measuring student outcomes, particularly in computer science and software engineering courses. The aim was to fill this knowledge gap by examining published evidence on assessment in computer science and software engineering courses. Furthermore, the goal is to understand how to effectively measure student outcomes in these fields.

II. Related Works

The study literature on Student Assessment Frameworks for Software Engineering Courses featured several noteworthy reviews. Firstly, Vihavainen et al. [29] examined the methodologies employed in introductory programming instruction. The examination centered on the impact of different interventions on the successful completion rate of basic programming courses. In the "Course Setup, Assessment, Resourcing" section of their paper, they expounded upon the assessment criteria relevant to programming courses.

Garcia et al. [30] conducted a review to gauge the utilization of original review techniques in contemporary computer science e-learning tools. The primary focus of the evaluation rested on e-learning tools, assessing whether technologies facilitating review techniques absent from the initial taxonomy had developed.

Luxton-Reilly et al. [31] performed a comprehensive review of the literature surrounding introductory programming. This literature was categorized into papers addressing students, teaching methods, curriculum, and assessment. The papers aimed to identify trends, pinpoint advancements in knowledge over the preceding 15 years, and propose potential areas for further investigation.

Keuning et al. [32] conducted a thorough review aimed at gaining a comprehensive understanding of specific tools. Those tools provided automated feedback, describing the types of feedback utilized in assisting students with programming. The paper scrutinized the relationship between feedback content and technology while assessing the adaptability of those tools. Despite several other studies examining those tools, none specifically explored the feedback they offered, the connection between feedback content and technological aspects, or the flexibility of the tools.

Lorås et al. [33] reviewed with the objective of uncovering insights into the study habits of computer science students and how they are influenced by instructional design. To discern and characterize study behaviors, processes, techniques, habits, and strategies, a taxonomy of study behaviors was formulated. This aimed to establish the correlation between the study behaviors and the educational environment.

Previous studies [29]-[33] provided a comprehensive exploration of the learning process holistically. However, it diverged by focusing exclusively on the assessment process, specifically within the context of software engineering courses. The intent was to create a detailed systematic literature review that offered an overarching perspective on the assessment framework within the context of software engineering and computer science courses, thereby bridging existing study gaps. Meanwhile, prior reviews underscored the significance of student assessments in the activities, this study focused squarely on the student assessment framework tailored to software engineering and computer science courses.

III. METHODS

This section explained the study method, which was based on the guideline proposed by [34]–[37]. Subsequently, the most important stages, including the formulation of study questions, search procedure, study selection, quality evaluation, and data extraction were outlined.

A. The Study Question

The primary objective of this study was to comprehend the student assessment framework for software engineering courses, hence the following questions were formulated.

RQ1: What assessment methods were employed in software engineering courses?

RQ2: What challenges did the assessment framework in software engineering courses encounter?

B. Search Process

Based on the guidelines [34], [35], the search process was initiated by defining the criteria employed. Papers from journals, workshops, and proceedings were obtained from six digital libraries, namely ScienceDirect, Mendeley, IEEEXplore, ACM Digital Library, Springer Link, and Emerald, within the timeframe of January 2012 to October 2021. The search criteria were divided into two parts, C1 and C2, as outlined below:

C1 is a string of keywords related to assessment frameworks, such as assessment and evaluation.

C2 is a string of keywords related to software engineering, computer science, and programming courses.

The following is an example of a search performed in electronic databases:

(Assessment Framework OR Evaluation Framework) AND (Software Engineering OR Computer Science OR Programming Course).

Inclusion and exclusion criteria, based on Inayat et al. [36], were applied to refine and select previously searched papers. The inclusion criteria for the study consisted of (I1) Peer-reviewed publications, (I2) Publications written in English, (I3) Publications related to the search keywords, (I4) Publications falling into categories such as papers, experience reports, or workshop papers, and (I5) Publications published between January 2012 and October 2021. The exclusion criteria for this study comprised, (E1) reviews not specifically addressing assessment in the fields of software engineering or computer science, (E2) investigations lacking discussions on assessment frameworks in software engineering, and (E3) those failing to meet the predefined inclusion criteria.

C. Study Selection

Searches were conducted in selected electronic databases using the technique described in Section B to acquire pertinent studies. Table 1 showed the initial search results, yielding 4,962 studies but a considerable number of them failed to meet the inclusion criteria. Many results were excluded due to limitations in applying the search string across complete articles. In Round 1, study analysts thoroughly examined study titles and abstracts, adhering to the inclusion criteria, resulting in the selection of 78 samples. In Round 2, the exclusion criteria (E1, E2, and E3) were applied to reassess the preselected studies. Face-to-face discussions were held to address consensus issues raised during evaluations. When consensus could not be reached on an article, the entire paper was read and those based on the established exclusion criteria were rejected. After applying the inclusion criteria to the 78 preselected papers, 58 were discarded as they did not directly address the question. Detailed readings of the remaining 78 were conducted, ultimately excluding 58 that were irrelevant.

TABLE 1
A NUMBER OF STUDIES WERE DISCERNED IN THE VARIOUS PHASES OF OUR SYSTEMATIC SEARCH

Database	Retrieved	Round 1		Round 2	
		Include	Exclude	Include	Exclude
Science Direct	351	13	338	6	7
Mendeley	2525	3	2522	2	1
IEEE Explore	414	14	400	5	9
ACM Digital Library	1265	25	1240	3	22
Springer Link	277	14	263	3	11
Emerald	129	9	120	1	8
Total	4962	78	4884	20	58

D. Quality Assessment

The quality criteria proposed by Guyatt et al. [38] were utilized in this systematic review to assess the methodological quality of the selected primary studies. These criteria comprised inquiries designed to gauge the

completeness, trustworthiness, and significance of the studies, with the aim of evaluating the value of the synthesis findings and facilitating interpretation.

An ordinal scale, based on the quality evaluation criteria rather than a binary measure, was employed to categorize and grade each study, following Guyatt et al. [38]. The first criterion (C1) entailed establishing the goal, with an affirmative response rate of 87% in the studies. The second criterion (C2) evaluated the management and documentation of the study environment, also receiving an 87% affirmative response rate. For the third (C3) and final criterion (C4), an evaluation was made to determine whether each study provided a concise summary of its results, with 83% supporting this aspect. It should be noted that the quality ratings (C4) were derived based on these three criteria.

E. Data Extraction

In accordance with the guidelines of Kitchenham et al. [35], a data extraction procedure was devised to collect pertinent data from the primary studies included in the investigation. A template was crafted to document the theoretical foundations, methodological innovations, and empirical results, ensuring a higher-level interpretation.

IV. RESULTS

This systematic review aimed to examine frameworks or tools employed by educators for the identification of student competencies within software engineering courses. A total of twenty studies [39], [40], [49]–[58], [41]–[48], as previously mentioned, were identified. Among the studies reviewed, 55 percent (11 studies) were presented at conferences, while 45 percent (9) found their place in journals. The samples were uniformly distributed across various publication platforms, with each source featuring one or two papers, indicating an absence of preference among authors for any particular source.

It should be noted that no significant studies relating to this topic were published before 2012. The articles under scrutiny spanned diverse fields, covering the period between 2012 and 2021 as shown in Fig. 1. Meanwhile, 2019 witnessed a substantial discourse on assessment frameworks, and 2018 showed relatively less emphasis on this subject.

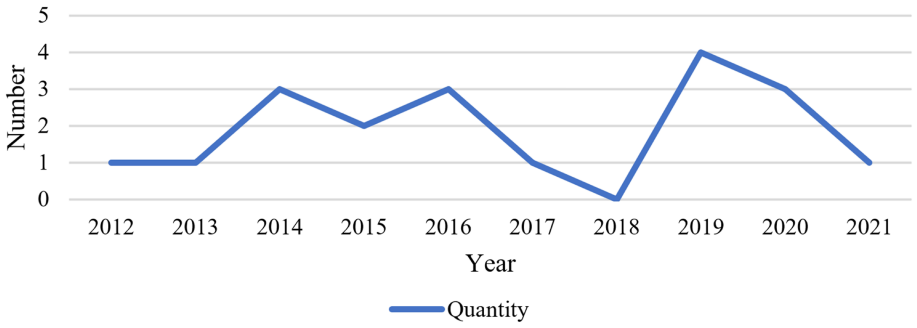

Fig. 1 A breakdown by year of selected studies

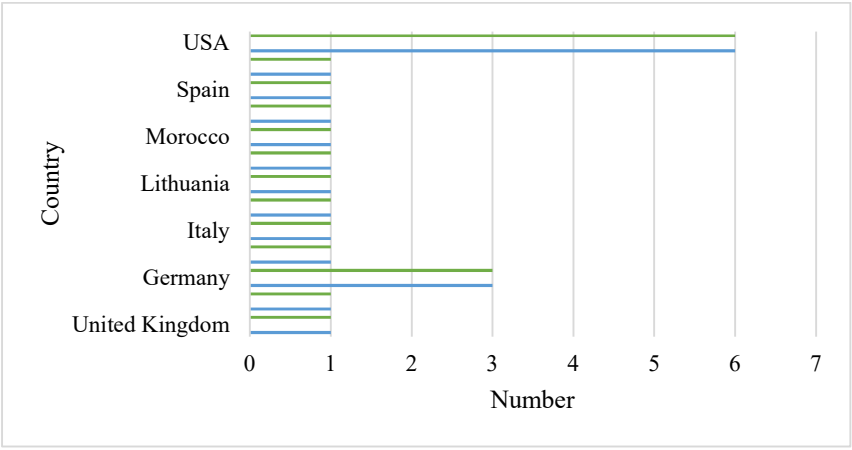

Fig. 2 Distribution of selected study by country

The distribution of papers by country is visually shown in Fig. 2. Papers were predominantly distributed in the USA, with a presence in Europe, but no representation from South America or Australia. Unfortunately, geographical information pertaining to the authors of each paper remained undisclosed.

A selection of 20 papers was made, comprising three empirical reviews categorized as "empirically appraised," evaluating methods or instruments without experimental investigations and case studies, and 17 "empirically grounded" studies employing qualitative techniques, such as surveys and interviews. Empirical-based studies were further classified into case studies, surveys, and observations, with 12 papers adopting case studies, 4 utilizing surveys, and 1 employing observation as seen in Fig. 3.

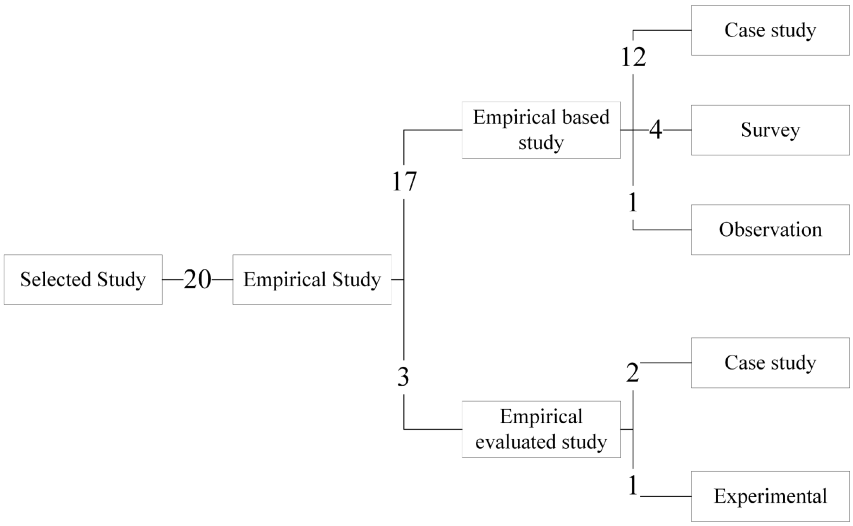

Fig. 3 Variations in study distributions based on the technique employed

V. DISCUSSION

A. RQ1: What are the assessment methods in the software engineering course?

The subsequent section described 13 assessment frameworks identified and their utilization across the studies.

1) Assessment Matrix

In the study conducted by Zeid [39], the challenge of waning student interest in software engineering courses was addressed through a student competition. Evaluation of student performance in this competition hinged upon an assessment matrix. Similarly, Traverso-Ribon et al. [40] engaged in project-based learning assessment via a web-based technique, also incorporating the use of an assessment matrix.

The utility of an assessment matrix was instrumental in safeguarding the validity of evaluations [59], [60]. Valid assessments accurately gauged the subject matter they were intended to assess, ensuring that they faithfully represented the desired learning outcomes for a course unit. However, it should be noted that reliance on an assessment matrix could introduce subjectivity into the assessment process, contingent upon the judgment of the assessor.

2) Automatic Assessment

WebWolf, a user-friendly framework designed for automated grading of assignments in introductory web programming courses, appeared as a viable solution. The WebWolf software demonstrated the ability to load web pages, identify and assess components, navigate hyperlinks, and establish expected outcomes. An evaluation of software performance included assignments from three distinct classes, incorporating a set of dispatch tasks deliberately embedded with errors. Impressively, WebWolf accurately identified errors within the submissions [41]. This automated assessment approach extended its utility to the evaluation of UML class diagrams [42]. Furthermore, it provided significant benefits in terms of assessment efficiency and objectivity for educators [43]–[46].

Even though automatic assessment had evident advantages, particularly in the domain of multiple-choice and short-answer questions, it also posed certain limitations. When faced with case study questions commonly encountered in software engineering courses, automated assessment presented challenges. Specific question types required specialized computations for precise scoring. Consequently, assessing diverse question types demanded distinct methodologies. For example, the evaluation of use case diagram questions [61] necessitated a different technique than the assessment of class diagram questions [62] due to their inherent distinctions. The efficacy of automatic assessment heavily relied on the accuracy of the answer key formulated by the instructor.

3) CDIO (Conceive, Design, Implement, and Operate)

The imperative to elevate the quality of educational offerings within Higher Education Institutions (HEI) necessitated the development of enhanced assessment frameworks in line with contemporary standards. Existing evaluation frameworks were informal and unsuitable for continual quality enhancement. A potential solution arose through the proposal of a novel assessment model and measurement methodology rooted in the CDIO Standard [47].

CDIO had three overarching objectives [63], [64], firstly, it aspired to foster a comprehensive grasp of the fundamental technical aspects of software engineering. Secondly, it aimed to empower individuals to take the lead in conceiving and executing new software solutions. Thirdly, it underscored the importance of comprehending the broader significance and strategic implications of studies and technological advancements within society. CDIO assessments exhibited a high degree of complexity, spanning multiple levels including program philosophy and curriculum development. Implementing this assessment paradigm within a software engineering course necessitated harmonious integration across all faculty members. Additionally, reusing CDIO assessments presented a challenge, requiring continuous updates and incurring additional costs.

4) Cooperative Thinking (CooT)

The concepts of computational thinking (CT) and agile values (AV) focused on the ability of an individual for algorithmic thinking and the basics of collaborative software development. A model for addressing computational problems in teams was known as cooperative thinking (CooT). Cooperative thinking referred to a new skill aimed at promoting collaborative problem-solving of technical information relevant to alleviating challenging software engineering problems. CooT was recommended for educational purposes to teach students how to create software to enhance their performance both individually and collectively [48].

As opposed to the previous exam, CooT assessed the ability of students for cooperative thinking in carrying out tasks rather than their assignment performance. CooT in the software engineering course consisted of five evaluation concepts, namely complex negotiation [65], [66], continuous learning [67], group awareness [68]–[70], group organization [71], and social adaptation [72]–[74].

5) Formative and Summative Assessment

This framework was designed for long-term evaluation, hence assessors could evaluate from the beginning until the completion of the learning process. Summative assessments [75], [76] evaluated the current levels of knowledge and the skills of students. The instructional process included a phase called formative assessment [77], which was integrated into classroom teaching to provide the information necessary to adapt instruction and learning. In software engineering classes, traditional educators also used this assessment. Through a well-developed software, such as Instructional Module Development System (IMODS) [49], evaluations that were conducted could be reviewed.

An open-source, web-based course design tool called IMODS provided a framework for presenting curricula, specifically in STEM fields. It guided users through the development process, and the evaluation method selected for this course combined formative and summative assessments. The model developed examined the consistency of learning domains, performance, and criteria needed to match the assessments selected for the course with the learning objectives [49].

6) Game

Video game technology, in the context of education (also known as "serious games"), was one of the methods used for assessments. This was because games were often exciting and inspiring, thereby motivating students to study programming structures in a fun and familiar setting, and apply learning from those environments to understand the fundamentals of computer programming through coding. There was a significant need for explicit instruction and analysis of how games could be created specifically to enhance problem-solving skills [50], [51].

Games increased student engagement and provided a setting for accurate and relevant assessment [78]. In addition to previous achievements and traditional assessments, games improved the success levels of students [79]. It should be noted that games were designed to assess the quality of individual understanding. In the software engineering

course, students generally enjoyed games because they frequently interacted with electronic devices. Therefore, assessment using this system was an effective tool for software engineering students. However, this framework had its weaknesses, as every game built had to be updated to be in line with the curriculum and the times. Games also required audio/visual technologies to support assessment [80].

7) Generative Learning Robot

E-learning was analyzed through a variety of viewpoints and methodologies. When the field was examined holistically and within the context of the Generative Learning Robot (GLO), a heterogeneous meta-program designed to teach computer science (CS) topics, namely programming, the process-based perspective appeared to be the most relevant. Drawing inspiration from the software engineering field, a feature-based technique was employed to model the inherent unpredictability of CS learning [52].

8) NIMSAD (Normative Information Model-based Systems Analysis and Design)

NIMSAD served as a framework for evaluating methods, industry/business practices, and problem-solving techniques. Meanwhile, methodologies could be oriented toward provisioning and enhancing operational efficiency, evaluating the suitability for addressing specific "problem situations" might be quite challenging. This was because the technique offered a general framework for reform [53].

NIMSAD found utility in problem-solving scenarios within software engineering courses [81]–[83], particularly in case studies focused on customer issues and solutions devised by students. NIMSAD evaluated three key factors, namely the problem solver, the contextual elements of the issue, and the procedure used to solve the problem.

9) SECAT (Software Engineering Competency Assessment Tool)

Evaluation often entailed two dimensions, firstly, there was an overarching view that considered the outcomes of various strategies, comprising an assessment of all course participants collectively. For example, this included comparing the competencies of students at the beginning of a course to those at the end. Secondly, there was an evaluation of competence concerning each student, which could be quantified on an absolute scale or in relation to their peers in the same course. Instructors evaluating performance gained valuable insights into the competencies of students by considering assessments from multiple assessors. Therefore, encouraging consistency among raters when using SECAT conventions enhanced its validity and reliability [54].

10) Self-Assessment and Peer Assessment

Proficiency in basic programming skills served as a prerequisite for Information Technology (IT) students. Mere memorization of programming concepts was considered insufficient, but, to gain a thorough understanding, students had to engage in practical tasks personally. Assessment and evaluation were conducted within the Problem-Based Learning (PBL) framework, assessing both the process and product. From a process perspective, it was crucial to consider and assess the soft skills of students, including self-learning and teamwork abilities, developed through study and project work. This perspective comprised two sub-aspects, namely self-evaluation and peer review [55]. Criteria for evaluation entailed learning motivation, self-directed learning capabilities, teamwork skills, and communication abilities. It was worth noting that the subjective element remained a limitation in this assessment.

11) Tools SonarQube

Programming referred to a hands-on discipline, and assessing the practical computer skills of students, specifically in large cohorts, was challenging. It should be noted that issues consisting of potential plagiarism and variations in the entire task difficulty often arose. To mitigate this problem, a multidimensional evaluation model for computer science courses was developed, drawing from extensive practice data within an e-learning system. This model comprised three dimensions, namely accuracy, originality, and quality detection, allowing for an in-depth examination of learning processes. Correctness was fundamental, while originality assessment entailed categorizing student behavior after detecting potential plagiarism in assignments. Additionally, teachers had the flexibility to participate in evaluations according to their needs. SonarQube was employed to detect code quality and suggest higher code standards [56].

12) WRENCH

Casanova et al. [57] devised a collection of pedagogical modules that could be periodically integrated into university courses to address these challenges. These courses incorporated simulation-driven exercises providing students with hands-on exposure to critical application and platform scenarios. Simulators, built using the WRENCH [84], [85] simulation framework, supported these endeavors. Following the introduction and outline of their approach, the study

described and evaluated the outcomes based on assessments conducted during two consecutive iterations of an undergraduate university course.

13) SEP-CyLE (Software Engineering and Programming Cyber learning environment)

SEP-CyLE employed a cognitive walkthrough coupled with a think-aloud methodology and a heuristic evaluation technique. The aim of the UI/UX evaluation was to enhance cyberlearning and deliver a cyberlearning environment tailored to specific users [58]. Subsequently, network-based analysis was employed to identify statistically significant correlations within the heuristic assessment survey, particularly regarding the perceptions of students in using SEP-CyLE. The application of this framework had some similarities with MOODLE [86], [87].

B. RQ2: What are the challenges of the assessment framework in a software engineering course?

In this section, the challenges associated with assessment frameworks in software engineering courses were explored.

1) Students have difficulty understanding the material, making it tedious to carry out an assessment

An existing assessment framework was identified as inappropriate for the task at hand, necessitating adjustments. Previous studies [50], [51] recognized a common issue, such as students struggling to grasp computer programming concepts. This challenge hindered the evaluation of students, as they often possessed a superficial understanding of the subject and lacked effective problem-solving skills. Zeid [39] addressed declining student interest in software engineering courses by introducing a competition. However, assessing performance in these competitions was quite challenging, and this was the reason the assessment matrix was also employed for evaluating students. Competency assessment [54] became a complex task due to vague definitions. Various solutions, including the use of game-based intermediaries [88], [89], were proposed to make the material more engaging.

2) Differences in educational practice make it difficult to find the proper assessment framework

For an extended period, educational practices centered around instructors had been considered the cornerstone of effective teaching. This created the perception that students primarily memorized material to pass examinations, as their focus remained solely on classroom lessons. Furthermore, due to the limited exposure to real-world scenarios, individuals often encountered difficulties when presented with practical challenges. In the context of Information Technology (IT), establishing a robust foundation in programming had been deemed essential. Simply memorizing programming concepts had proved inadequate, but to attain a comprehensive understanding, students needed to actively engage in practical exercises. Although methodologies had often prioritized provisioning and operational efficiency, evaluating the applicability to specific real-world situations remained a complex task. This was the reason a framework facilitating the analysis of methodologies, industry practices, and problem-solving techniques became necessary. Assessment was conducted using the Problem-Based Learning (PBL) framework, evaluating both the process and the product [55].

3) Assessors must read and analyze student source code, which takes a long time

The assessment of student work in web page assignments presented challenges and consumed a significant amount of time. This process entailed the inspection and analysis of source code to ensure it adhered to the requirements. Consequently, there was likely a reduction in the number of assignments given, and educators dedicated substantial time to grading, which limited their interactions with students and hindered other essential tasks. The page should also be examined in the browser for necessary features, such as operational links [41]–[46].

4) Plagiarism on student assignments

In the field of computer science, assessing the computer skills of a large student population using computer-assisted tools proved to be intricate. Challenges included allegations of plagiarism and variations in the entire difficulty level.

C. Implication and limitation

The geographical distribution of studies was emphasized, and the results showed that the majority of investigations originated in the USA. This was not surprising, considering the esteemed reputation of the country in the aspect of education, particularly within the fields of computer science and software engineering. On the other hand, studies addressing assessment frameworks in software engineering courses from Asian countries represented only a small fraction.

This study identified 13 assessment frameworks, including the Assessment Matrix, Automatic Assessment, CDIO, Cooperative Thinking, formative and summative assessment, Game, Generative Learning Robot, NIMSAD, SECAT,

Self-assessment and Peer Assessment, SonarQube Tools, WRENCH, and SEP-Cycle. As a result, alterations to the framework were considered necessary due to the recurring issues that were recorded.

Adjustments were implemented in the assessment framework to facilitate effective student evaluation. Some of the adjustments assisted in simplifying the process of evaluating student work.

This review had significant implications as technology advanced and the field of computer education evolved. This evolution presented challenges in assessing computer courses, necessitating extensive study to create appropriate assessment frameworks that would be in line with the learning system.

It should be noted that this systematic review encountered two primary constraints, namely potential bias in study selection and data extraction from source variables. To mitigate these biases, a multistage technique was employed in which two investigators independently assessed the relevance of each study based on predetermined inclusion and exclusion criteria. Initially, searches were conducted using similar keywords in electronic databases, then papers were selected based on predefined rules, and each study result went through meticulous evaluation.

VI. Conclusions

In conclusion, this paper provided an overview of the literature concerning assessment frameworks in software engineering and computer engineering courses. An initial pool of 4,962 papers was obtained from reputable electronic databases, and 20 relevant papers were selected through a meticulous, step-by-step stratified transfer process that went through independent validation at each stage. The examination of these papers revealed a recurring theme, such as the necessity for adjustments within the assessment framework. This need came from an inconsistency between the framework and learning activities in progress. Consequently, the task of accurately assessing student work presented a formidable challenge. Across the papers dedicated to assessment frameworks in software engineering and computer engineering courses, a total of 13 distinct frameworks were selected, each crafted to address these issues. This exploration shed light on several challenges inherent in the assessment framework of software engineering courses. The challenges comprised struggles faced by students in comprehending course materials, disparities in educational practices, the intricacies of detailed assessments, and concerns related to plagiarism in student assignments.

This study possessed substantial value in its endeavor to map the encountered problems and the corresponding solutions in the assessment framework, particularly within the domains of software engineering and computer science education. These insights could serve as foundational references for educational institutions aiming to establish assessment frameworks that seamlessly fitted with the selected pedagogical techniques.

Author Contributions: *Reza Fauzan*: Conceptualization, Methodology, Writing - Original Draft, Writing - Review & Editing, Supervision. *Daniel Siahaan*: Conceptualization, Methodology, Supervision. *Mirotus Solekhah*: Investigation, Data Curation, Writing - Original Draft. *Vriza Wahyu Saputra*: Investigation, Data Curation, Writing - Original Draft. *Aditya Eka Bagaskara*: Investigation, Data Curation, Writing - Original Draft. *Muhammad Ihsan Karimi*: Writing - Review & Editing.

All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by Politeknik Negeri Banjarmasin and Institut Teknologi Sepuluh Nopember under Postdoctoral research.

Conflicts of Interest: *Siahaan* is member of Editorial Teams, but had no role in the decision to publish this article. No other potential conflict of interest relevant to this article was reported.

Data Availability: Data sharing is not applicable to this article as no new data were created or analyzed in this study.

Informed Consent: There were no human subjects.

Animal Subjects: There were no animal subjects.

ORCID:
Reza Fauzan: https://orcid.org/0000-0002-4113-9848

Daniel Siahaan: https://orcid.org/0000-0001-6560-2975
Mirotus Solekhah: N/A
Vriza Wahyu Saputra: N/A
Aditya Eka Bagaskara: N/A
Muhammad Ihsan Karimi: N/A

REFERENCES

[1] M. Murtiningsih, M. Kristiawan, and B. Lian, "The correlation between supervision of headmaster and interpersonal communication with work ethos of the teacher," *Eur. J. Educ. Stud.*, 2019.
[2] E. Balbachevsky, H. Sampaio, and C. Y. de Andrade, "Expanding access to higher education and its (limited) consequences for social inclusion: The Brazilian experience," *Soc. Incl.*, vol. 7, no. 1, pp. 7–17, 2019.
[3] H. A. M. A. M. Abdeljaber, S. Ahmad, and A. Sultan, "Program outcomes assessment method for multi-academic accreditation bodies: Computer science program as a case study," *Int. J. Emerg. Technol. Learn.*, vol. 12, no. 5, p. 23, 2017, doi: 10.3991/ijet.v12i05.6410.
[4] A. Shafi, S. Saeed, Y. A. Bamarouf, S. Z. Iqbal, N. Min-Allah, and M. A. Alqahtani, "Student outcomes assessment methodology for ABET accreditation: A case study of computer science and computer information systems programs," *IEEE Access*, vol. 7, pp. 13653–13667, 2019, doi: 10.1109/ACCESS.2019.2894066.
[5] P. Peretti-Watel, J. K. Ward, C. Vergelys, A. Bocquier, J. Raude, and P. Verger, "'I think I made the right decision… I hope I'm not wrong'. Vaccine hesitancy, commitment and trust among parents of young children," *Sociol. Health Illn.*, vol. 41, no. 6, pp. 1192–1206, 2019.
[6] S. Fulton and D. Schweitzer, "Impact of Giving Students a Choice of Homework Assignments in an Introductory Computer Science Class.," *Int. J. Scholarsh. Teach. Learn.*, vol. 5, no. 1, p. n1, 2011.
[7] R. Fojtík, "Problems of distance education," *Icte J.*, vol. 7, no. 1, pp. 14–23, 2018.
[8] B. Prevalla and H. Uzunboylu, "Flipped learning in engineering education," *TEM J.*, vol. 8, no. 2, p. 656, 2019.
[9] F. M. Newmann, "Higher order thinking in teaching social studies: A rationale for the assessment of classroom thoughtfulness," *J. Curric. Stud.*, vol. 22, no. 1, pp. 41–56, 1990.
[10] I. Chirikov, T. Semenova, N. Maloshonok, E. Bettinger, and R. F. Kizilcec, "Online education platforms scale college STEM instruction with equivalent learning outcomes at lower cost," *Sci. Adv.*, vol. 6, no. 15, p. eaay5324, 2020.
[11] E. Care, H. Kim, A. Vista, and K. Anderson, "Education System Alignment for 21st Century Skills: Focus on Assessment.," *Cent. Univers. Educ. Brookings Inst.*, 2018.
[12] L. Fritz, "Effective Assessment for Early Courses in Computer Science: Instruments Other than Out-of-Class Programming Assignments.," *J. Instr. Res.*, vol. 8, no. 2, pp. 118–121, 2019, doi: 10.9743/jir.2019.8.2.17.
[13] V. Kioupi and N. Voulvoulis, "Education for sustainable development: A systemic framework for connecting the SDGs to educational outcomes," *Sustainability*, vol. 11, no. 21, p. 6104, 2019.
[14] A. Gacs, S. Goertler, and S. Spasova, "Planned online language education versus crisis-prompted online language teaching: Lessons for the future," *Foreign Lang. Ann.*, vol. 53, no. 2, pp. 380–392, 2020.
[15] F. Martin, A. Ritzhaupt, S. Kumar, and K. Budhrani, "Award-winning faculty online teaching practices: Course design, assessment and evaluation, and facilitation," *Internet High. Educ.*, vol. 42, pp. 34–43, 2019.
[16] R. E. Tractenberg, "The assessment evaluation rubric: Promoting learning and learner-centered teaching through assessment in face-to-face or distanced higher education," *Educ. Sci.*, vol. 11, no. 8, p. 441, 2021.
[17] C. Culver, "Learning as a peer assessor: evaluating peer-assessment strategies," *Assess. Eval. High. Educ.*, pp. 1–17, 2022.
[18] T. Y. Pang, A. Kootsookos, K. Fox, and E. Pirogova, "Does an Assessment Rubric Provide a Better Learning Experience for Undergraduates in Developing Transferable Skills?.," *J. Univ. Teach. Learn. Pract.*, vol. 19, no. 3, p. 3, 2022.
[19] Z. Beasley, A. Friedman, L. Pieg, and P. Rosen, "Leveraging peer feedback to improve visualization education," in *2020 IEEE Pacific Visualization Symposium (PacificVis)*, 2020, pp. 146–155.
[20] D. Chang, G.-J. Hwang, S.-C. Chang, and S.-Y. Wang, "Promoting students' cross-disciplinary performance and higher order thinking: A peer assessment-facilitated STEM approach in a mathematics course," *Educ. Technol. Res. Dev.*, vol. 69, pp. 3281–3306, 2021.
[21] F. A. K. A. Salem, I. W. Damaj, L. A. Hamandi, and R. N. Zantout, "Effective Assessment of Computer Science Capstone Projects and Student Outcomes.," *Int. J. Eng. Pedagog.*, vol. 10, no. 2, pp. 72–93, 2020, doi: 10.3991/ijep.v10i2.11855.
[22] C. Allen and D. M. A. Mehler, "Open science challenges, benefits and tips in early career and beyond," *PLoS Biol.*, vol. 17, no. 5, p. e3000246, 2019.
[23] M. Ianniello, S. Iacuzzi, P. Fedele, and L. Brusati, "Obstacles and solutions on the ladder of citizen participation: a systematic review," *Public Manag. Rev.*, vol. 21, no. 1, pp. 21–46, 2019.
[24] S. Areekkuzhiyil, "Issues and Concerns in Classroom Assessment Practices.," *Online Submiss.*, 2021.
[25] A. M. Moreno, M.-I. Sanchez-Segura, F. Medina-Dominguez, and L. Carvajal, "Balancing software engineering education and industrial needs," *J. Syst. Softw.*, vol. 85, no. 7, pp. 1607–1620, 2012.
[26] V. Garousi, G. Giray, E. Tuzun, C. Catal, and M. Felderer, "Closing the gap between software engineering education and industrial needs," *IEEE Softw.*, vol. 37, no. 2, pp. 68–77, 2019.
[27] R. Fauzan, D. O. Siahaan, S. Rochimah, and E. Triandini, "A Novel Approach to Automated Behavioral Diagram Assessment using Label Similarity and Subgraph Edit Distance," *Comput. Sci.*, vol. 22, no. 2, pp. 191–207, 2021.
[28] R. Fauzan, D. Siahaan, S. Rochimah, and E. Triandini, "A Different Approach on Automated Use Case Diagram Semantic Assessment," *Int. J. Intell. Eng. Syst.*, vol. 14, no. 1, pp. 496–505, Feb. 2021.
[29] A. Vihavainen, J. Airaksinen, and C. Watson, "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*, 2014, pp. 19–26. doi: 10.1145/2632320.2632349.
[30] R. Garcia, K. Falkner, and R. Vivian, "Systematic literature review: Self-Regulated Learning strategies using e-learning tools for Computer Science," *Comput. Educ.*, vol. 123, no. April, pp. 150–163, 2018, doi: 10.1016/j.compedu.2018.05.006.
[31] A. Luxton-Reilly *et al.*, "Introductory programming: a systematic literature review," in *Proceedings companion of the 23rd annual ACM conference on innovation and technology in computer science education*, 2018, pp. 55–106. doi: 10.1145/3293881.3295779.

[32] H. Keuning, J. Jeuring, and B. Heeren, "A systematic literature review of automated feedback generation for programming exercises," *ACM Trans. Comput. Educ.*, vol. 19, no. 1, 2018, doi: 10.1145/3231711.
[33] M. Lorås, G. Sindre, H. Trætteberg, and T. Aalberg, "Study behavior in computing education—a systematic literature review," *ACM Trans. Comput. Educ.*, vol. 22, no. 1, pp. 1–40, 2021.
[34] B. Kitchenham, O. P. Brereton, D. Budgen, M. Turner, J. Bailey, and S. Linkman, "Systematic literature reviews in software engineering–a systematic literature review," *Inf. Softw. Technol.*, vol. 51, no. 1, pp. 7–15, 2009.
[35] B. Kitchenham *et al.*, "Systematic literature reviews in software engineering-A tertiary study," *Inf. Softw. Technol.*, vol. 52, no. 8, pp. 792–805, 2010, doi: 10.1016/j.infsof.2010.03.006.
[36] I. Inayat, S. S. Salim, S. Marczak, M. Daneva, and S. Shamshirband, "A systematic literature review on agile requirements engineering practices and challenges," *Comput. Human Behav.*, vol. 51, pp. 915–929, 2015, doi: 10.1016/j.chb.2014.10.046.
[37] E. Triandini, R. Fauzan, D. O. Siahaan, S. Rochimah, I. G. Suardika, and D. Karolita, "Software similarity measurements using UML diagrams: A systematic literature review," *Regist. J. Ilm. Teknol. Sist. Inf.*, vol. 8, no. 1, p. 10, 2021, doi: 10.26594/register.v8i1.2248.
[38] G. H. Guyatt *et al.*, "Users' guides to the medical literature: XXV. Evidence-based medicine: principles for applying the users' guides to patient care," *Jama*, vol. 284, no. 10, pp. 1290–1296, 2000.
[39] A. Zeid, "A framework to evaluate software engineering student contests: Evaluation and integration with academic programs," in *2013 35th International Conference on Software Engineering (ICSE)*, 2013, pp. 1083–1089.
[40] I. Traverso-Ribon, A. Balderas-Alberico, J.-M. Dodero, I. Ruiz-Rube, and M. Palomo-Duarte, "Open data framework for sustainable assessment of project-based learning experiences," *Program*, vol. 50, no. 4, pp. 380–398, 2016.
[41] A. C. Siochi and W. R. Hardy, "WebWolf: Towards a simple framework for automated assessment of webpage assignments in an introductory web programming class," in *Proceedings of the 46th ACM Technical Symposium on Computer Science Education*, 2015, pp. 84–89.
[42] T. Reischmann and H. Kuchen, "Towards an E-assessment tool for advanced software engineering skills," in *Proceedings of the 16th Koli Calling International Conference on Computing Education Research*, 2016, pp. 81–90.
[43] S. Zougari, M. Tanana, and A. Lyhyaoui, "Towards an automatic assessment system in introductory programming courses," in *2016 International Conference on Electrical and Information Technologies (ICEIT)*, 2016, pp. 496–499.
[44] J. A. Sánchez *et al.*, "Cloud service as the driver for university's software engineering programs digital transformation," *Procedia Comput. Sci.*, vol. 149, pp. 215–222, 2019.
[45] G. Polito, M. Temperini, and A. Sterbini, "2tsw: Automated assessment of computer programming assignments, in a gamified web based system," in *2019 18th International Conference on Information Technology Based Higher Education and Training (ITHET)*, 2019, pp. 1–9.
[46] V.-A. Valavosiki, E. Stiakakis, and A. Chatzigeorgiou, "Development of a Framework for the Assessment of Soft Skills in the ICT Sector," in *Operational Research in the Digital Era–ICT Challenges: 6th International Symposium and 28th National Conference on Operational Research, Thessaloniki, Greece, June 2017*, 2019, pp. 105–123.
[47] S. Rouvrais and C. Lassudrie, "An assessment framework for engineering education systems," in *Software Process Improvement and Capability Determination: 14th International Conference, SPICE 2014, Vilnius, Lithuania, November 4-6, 2014, Proceedings 14*, 2014, pp. 250–255.
[48] P. Ciancarini, M. Missiroli, and D. Russo, "Cooperative Thinking: Analyzing a new framework for software engineering education," *J. Syst. Softw.*, vol. 157, p. 110401, 2019.
[49] S. Bansal, A. Bansal, and O. Dalrymple, "Outcome-based Education model for computer science Education," *J. Eng. Educ. Transform.*, vol. 28, no. 2, pp. 113–121, 2015.
[50] C. Kazimoglu, M. Kiernan, L. Bacon, and L. Mackinnon, "A serious game for developing computational thinking and learning introductory computer programming," *Procedia-Social Behav. Sci.*, vol. 47, pp. 1991–1999, 2012.
[51] Y. S. Wong, M. Y. M. Hayati, W. H. Tan, and L. C. Yap, "A Game-Based Learning Assessment Framework for Learning Ubiquitous Computational Thinking," in *The Impact of the 4th Industrial Revolution on Engineering Education: Proceedings of the 22nd International Conference on Interactive Collaborative Learning (ICL2019)–Volume 2 22*, 2020, pp. 607–615.
[52] V. Štuikys, R. Burbaitė, K. Bespalova, and G. Ziberkas, "Model-driven processes and tools to design robot-based generative learning objects for computer science education," *Sci. Comput. Program.*, vol. 129, pp. 48–71, 2016.
[53] L. Sadath and S. Gill, "ETHICS and SSM—A critical human element evaluation in software engineering using the NIMSAD framework," in *2017 International Conference on Infocom Technologies and Unmanned Systems (Trends and Future Directions)(ICTUS)*, 2017, pp. 370–375.
[54] Y. Sedelmaier and D. Landes, "A multi-perspective framework for evaluating software engineering education by assessing students' competencies: SECAT—A software engineering competency assessment tool," in *2014 IEEE Frontiers in Education Conference (FIE) Proceedings*, 2014, pp. 1–8.
[55] P. Panwong and K. Kemavuthanon, "Problem-based learning framework for junior software developer: Empirical study for computer programming students," *Wirel. Pers. Commun.*, vol. 76, pp. 603–613, 2014.
[56] J. Luo, F. Lu, and T. Wang, "A multi-dimensional assessment model and its application in E-learning courses of computer science," in *Proceedings of the 21st Annual Conference on Information Technology Education*, 2020, pp. 187–193.
[57] H. Casanova, R. Tanaka, W. Koch, and R. F. da Silva, "Teaching parallel and distributed computing concepts in simulation with wrench," *J. Parallel Distrib. Comput.*, vol. 156, pp. 53–63, 2021.
[58] H. W. Alomari, V. Ramasamy, J. D. Kiper, and G. Potvin, "A User Interface (UI) and User eXperience (UX) evaluation framework for cyberlearning environments in computer science and software engineering education," *Heliyon*, vol. 6, no. 5, 2020.
[59] B. M. Olds and R. L. Miller, "An assessment matrix for evaluating engineering programs," *J. Eng. Educ.*, vol. 87, no. 2, pp. 173–178, 1998.
[60] A. Cun, S. Abramovich, and J. M. Smith, "An assessment matrix for library makerspaces," *Libr. Inf. Sci. Res.*, vol. 41, no. 1, pp. 39–47, 2019.
[61] F. Zulfa, D. O. Siahaan, R. Fauzan, and E. Triandini, "Inter-Structure and Intra-Structure Similarity of Use Case Diagram using Greedy Graph Edit Distance," *2020 2nd Int. Conf. Cybern. Intell. Syst. ICORIS 2020*, pp. 3–8, 2020.
[62] R. Fauzan, D. Siahaan, S. Rochimah, and E. Triandini, "Automated Class Diagram Assessment using Semantic and Structural Similarities," *Int. J. Intell. Eng. Syst.*, vol. 14, no. 2, 2021.
[63] E. Crawley, J. Malmqvist, S. Ostlund, D. Brodeur, and K. Edstrom, "Rethinking engineering education," *CDIO approach*, vol. 302, no. 2, pp. 60–62, 2007.
[64] C. Lassudrie, J. Kontio, and S. Rouvrais, "Managing the Continuous Improvement Loop of Educational Systems: Students as key actors in program evaluation," in *CDIO 2013: 9th International conference: Engineering Leadership in Innovation and Design.*, 2013.
[65] K. Beck, *Extreme programming explained: embrace change*. addison-wesley professional, 2000.

[66] K. Avruch, "Culture and negotiation pedagogy," *Negot. J.*, vol. 16, no. 4, pp. 339–346, 2000.
[67] D. A. Schön, *Educating the reflective practitioner: Toward a new design for teaching and learning in the professions*. Jossey-Bass, 1987.
[68] D. Bodemer and J. Dehler, "Group awareness in CSCL environments," *Comput. Human Behav.*, vol. 27, no. 3, pp. 1043–1045, 2011.
[69] P. Kristiansen and R. Rasmussen, *Building a better business using the Lego serious play method*. John Wiley & Sons, 2014.
[70] J.-P. Steghöfer, "Providing a baseline in software process improvement education with lego scrum simulations," in *Proceedings of the 40th International Conference on Software Engineering: Software Engineering Education and Training*, 2018, pp. 126–135.
[71] W. Damon and E. Phelps, "Critical distinctions among three approaches to peer education," *Int. J. Educ. Res.*, vol. 13, no. 1, pp. 9–19, 1989.
[72] G. R. Adams, "Social competence during adolescence: Social sensitivity, locus of control, empathy, and peer popularity," *J. Youth Adolesc.*, vol. 12, no. 3, pp. 203–211, 1983.
[73] M. Kuhrmann and J. Münch, "When teams go crazy: An environment to experience group dynamics in software project management courses," in *Proceedings of the 38th International Conference on Software Engineering Companion*, 2016, pp. 412–421.
[74] H. Burden, J.-P. Steghöfer, and O. H. Svensson, "Facilitating entrepreneurial experiences through a software engineering project course," in *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET)*, 2019, pp. 28–37.
[75] D. D. Dixson and F. C. Worrell, "Formative and summative assessment in the classroom," *Theory Pract.*, vol. 55, no. 2, pp. 153–159, 2016.
[76] J. D. Kibble, "Best practices in summative assessment," *Adv. Physiol. Educ.*, vol. 41, no. 1, pp. 110–119, 2017.
[77] R. E. Bennett, "Formative assessment: A critical review," *Assess. Educ. Princ. policy Pract.*, vol. 18, no. 1, pp. 5–25, 2011.
[78] K. L. McClarty, A. Orr, P. M. Frey, R. P. Dolan, V. Vassileva, and A. McVay, "A literature review of gaming in education," *Gaming Educ.*, vol. 1, no. 1, pp. 1–35, 2012.
[79] D. Dziob, "Board game in physics classes—A proposal for a new method of student assessment," *Res. Sci. Educ.*, vol. 50, no. 3, pp. 845–862, 2020.
[80] F. Bellotti, B. Kapralos, K. Lee, P. Moreno-Ger, and R. Berta, "Assessment in and of serious games: An overview," *Adv. human-computer Interact.*, vol. 2013, p. 1, 2013.
[81] N. Jayaratna, *Understanding and evaluating methodologies: NIMSAD, a systematic framework*. McGraw-Hill, Inc., 1994.
[82] H. Armstrong and N. Jayaratna, "Applying the NIMSAD Framework to Evaluating IA Education Projects," in *Proceedings from the Ninth Colloquium for Information Systems Security Education (CISSE9)*, 2005, pp. 124–129.
[83] J. Koskinen, H. Lintinen, H. Sivula, and T. Tilus, "Evaluation of software modernization estimation methods using NIMSAD meta framework," *Publ. Inf. Technol. Res. Inst.*, vol. 15, 2004.
[84] H. Casanova, S. Pandey, J. Oeth, R. Tanaka, F. Suter, and R. F. Da Silva, "Wrench: A framework for simulating workflow management systems," in *2018 IEEE/ACM Workflows in Support of Large-Scale Science (WORKS)*, 2018, pp. 74–85.
[85] H. Casanova *et al.*, "Developing accurate and scalable simulators of production workflow management systems with wrench," *Futur. Gener. Comput. Syst.*, vol. 112, pp. 162–175, 2020.
[86] H. Athaya, R. D. A. Nadir, D. Indra Sensuse, K. Kautsarina, and R. R. Suryono, "Moodle Implementation for E-Learning: A Systematic Review," in *Proceedings of the 6th International Conference on Sustainable Information Engineering and Technology*, 2021, pp. 106–112.
[87] S. H. P. W. Gamage, J. R. Ayres, and M. B. Behrend, "A systematic review on trends in using Moodle for teaching and learning," *Int. J. STEM Educ.*, vol. 9, no. 1, pp. 1–24, 2022.
[88] M. D. Ayastuy and D. Torres, "Adaptive gamification in collaborative location collecting systems: a case of traveling behavior detection," *J. Comput. Sci. Technol.*, vol. 22, no. 1, pp. e05–e05, 2022.
[89] J. Miguel, A. Chimuris Gimenez, N. Garrido, M. Bassi, G. Velazquez, and M. D. Panizzi, "State of the art on the conceptual modeling of serious games through a systematic mapping of the literature," *J. Comput. Sci. Technol.*, vol. 22, 2022.