A Systematic Literature Review of Teaching and Learning on Object-Oriented Programming Course

Efan*, Krismadinata, Jalius Jama, and Rudi Mulya

***Abstract*—Object-oriented programming is a paradigm that allows us to write programs by modeling real-world things in the form of classes and objects. The main goal of Object-oriented programming is to develop software that integrates various attributes, such as reusability, maintainability, and reliability. Many researchers have identified various problems and proposed concrete solutions regarding the teaching and learning process of Object-oriented programming courses. However, a map of problems and solutions is needed that summarizes what has been discussed by previous researchers so that it can be seen what the core problems related to learning in Object-oriented programming courses. In this study, we conducted a systematic literature review on course problems and solutions in OOP learning. For that purpose, we considered research works published from 2017 to 2021 in IEEE Xplore, ACM Digital Library, and Google Scholar as many as 9664 articles based on search keywords. The method used in the SLR performs three stages, namely planning, implementation, and reporting. From the SLR process, there are 60 articles for our research and review them from related issues from three aspects, namely students, content and technology, and lecturers. Based on the analysis, four major problems were identified, namely the complexity and abstraction of the material, the learning model, the lecture time, and the background and experience of students. The majority of researchers propose the development of learning media or the development of learning models.**

***Index Terms*—Object-oriented programming, teaching, learning, systematic literature review.**

I. Introduction

The object-oriented paradigm in design and programming has been the main approach in software development for more than two decades. In the field of computer science, one of the courses that apply this paradigm is the Object-Oriented Programming (OOP) course [1, 2]. The main goal of OOP is to develop a software system that combines various quality attributes, such as reusability, maintainability, and reliability [3]. Then students can construct software by representing it as close as possible to the real world, where all entities are treated as objects and each object has characteristics such as the ability to perform relationships with other objects [4].

OOP is a programming paradigm designed to represent objects into procedure blocks by taking into account the objects, classes, properties, methods, etc. contained in each of these objects [5]. Objects are the basic entities of OOP and are instances of classes, so objects are needed for any development of any program [6]. Objects act as intermediaries between programs and the methods and properties added to them. While the class is a blueprint of a particular object that contains properties and methods and forms the basic unit of software development. Classes are a key element in the development and maintenance of OOP software [7] by using properties and methods to operate the given data. Furthermore, the class format can be improved by using the concept of inheritance, polymorphism, and so on.

Various problems in the OOP learning process have been identified by previous researchers. For example, Wong and Hayati *et al.* [8] identified that first-year students who do not have basic programming experience will find it increasingly difficult to learn OOP, then propose the development of mobile games combined with game-based learning; Seng and Mohamad Yatim *et al.* [9] consider traditional classroom-based learning methods to be ineffective and the available time is insufficient to complete the full OOP curriculum, then they also propose the development of game-based learning; Silva and Dora [4] found that learning programming for students is a very expensive task that slows down the learning process and results in increased difficulty in learning more advanced concepts. Next, they proposed the development of an expert system model. Boudia and Bengueddach *et al.* [10] stated programming is a difficult task for students because it requires metacognitive skills such as abstraction, deep understanding, tenacity, and the ability to perform problem-solving stages in programming. Then they tried to demonstrate the effectiveness of the collaborative learning strategy and the impact on the learning process they had developed. Ardiana and Loekito [11] discussed the problem of low student motivation during class sessions due to passive learning styles, lack of understanding of previous programming classes, and trying to design games (game-based learning) to increase student motivation and involvement in OOP courses.

Many researchers have identified various problems and proposed solutions regarding the learning process of OOP courses. However, a map of the problems and solutions is needed that summarizes what has been discussed by previous researchers so that it can be known what the core problems related to learning in OOP courses are. This study aims to map the aspects of the problems in OOP courses that have been discussed by previous researchers and the various concrete solutions they have offered. This research was conducted by reviewing previous studies published from 2017 to 2021 and related to the teaching and learning process in OOP courses. The results of this study can be used as a reference for practitioners and future researchers to present

Manuscript received May 26, 2022, revised August 30, 2022, accepted September 29, 2022.

Efan is with Universitas Negeri Padang, Padang, West Sumatra, Indonesia and with the Institut Teknologi Pagar Alam, Pagar Alam, South Sumatra, Indonesia.

Krismadinata, Jalius Jama and Rudi Mulya are with the Universitas Negeri Padang, Padang, West Sumatra, Indonesia (e-mail: krisma@ft.unp.ac.id, jamajalius@yahoo.com, rudimuly4@gmail.com).

*Correspondence: efan@itpa.ac.id

more innovative solutions in the future by prioritizing the core problems related to OOP learning.

II. Method

Systematic Literature Review (SLR) is a process of identifying, assessing, and interpreting all available research phenomena to provide answers to research questions [12]. SLR is carried out in 3 main stages, namely planning, conducting, and reporting. Overall, the three main stages are constructed into 9 steps. Step 1 is done to get the reasons why SLR is done which are defined in the introduction section of this article. Step 2 (Protocol review) is designed to facilitate the execution of subsequent review processes. The review protocol defined the research question (RQ), search strategy, article selection process based on inclusion and exclusion criteria, article quality assessment, data extraction, and synthetic analysis process. The review protocol was represented in the A, B, C, and D sections of this chapter. Furthermore, step 3 was carried out and developed repeatedly during the conducting and reporting stages. Fig. 1 shows the complete process.

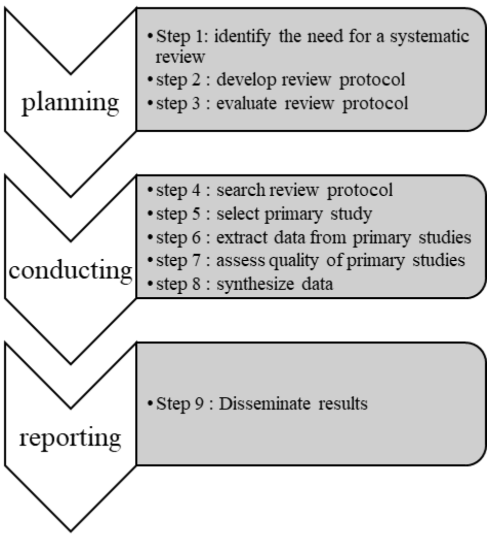
planning
• Step 1: identify the need for a systematic review
• step 2 : develop review protocol
• step 3 : evaluate review protocol
conducting
• step 4 : search review protocol
• step 5 : select primary study
• step 6 : extract data from primary studies
• step 7 : assess quality of primary studies
• step 8 : synthesize data
reporting
• Step 9 : Disseminate results

Fig. 1. SLR steps (adapted from [12]).

A. Research Question

Identifying research questions is one of the important steps in the SLR which is stated specifically so that the focus of the review is maintained, namely as follows.

RQ1: What is the core problem in the learning process of OOP courses from 2017 to 2021?

RQ2: What solutions have been offered for the core problems in the learning process of OOP courses from 2017 to 2021?

B. Search Process

The first activity was to determine the search string as shown in Table I.

TABLE I: Search String

Search String
(“teaching” ***OR*** *“learning”* ***OR*** *“education”* ***OR*** *“pedagogy”)* ***AND*** *(“object-oriented programming”* ***OR*** *“object-oriented paradigm”* ***OR*** *“object-oriented model”* ***OR*** *“object-oriented”* ***OR*** *“OOP”)*

The next activity was to determine the digital library that will be accessed as a search source. The following are the digital libraries accessed in this study:

1) IEEE Xplore (IEEE)
2) ACM Digital Library (ACM-DL)
3) Google Scholar (GS)

The following criteria are used as a reference in determining the articles to be reviewed:

1) Published between 2017 to 2021
2) Documents must be articles in reputable international journals or international seminar proceedings
3) The title must contain the phrase “object-oriented programming” or its acronyms
4) The article is written in English
5) Articles can be downloaded
6) The article is a primary research
7) Focus on models, methods, strategies, or learning approaches (teaching and learning)
8) Preference will be given to those who have a focus on a learning model
9) Meet the article quality assessment

C. Article Quality Assessment

Quality assessment criteria are defined based on the objectives of the SLR by answering the following questions:

Q1: Does the article state the research problem?

Q2: Does the article clearly state the proposed solution?

Each question was given a score as shown in Table II.

TABLE II: Article Quality Assessment Score Guide

Question	Y (1)	P (0.5)	N (0)
Q1	If the article states the problem very unequivocally	If the article states the problem less emphatically	If the article does not state the problem unequivocally
Q2	If the article states the proposed solution very clearly?	If the article states the proposed solution is less clear?	If the article does not clearly state the proposed solution?

D. Study Selection

Based on search keywords, document type, and publication period, IEEE displays 802 titles (152 journal article titles and 650 proceedings article titles), ACM-DL displays 1682 titles (321 journal article titles and 1361 proceedings article titles), GS displays 7180 article titles journals and proceedings. A list of titles that have been displayed in each digital library containing the phrase “object-oriented programming” was selected. The selection process left 32 titles on the IEEE, 16 titles on the ACM-DL, and 463 titles on the GS (see Fig. 2).

Full titles and abstracts and other identities are filtered for relevance, language used, and DOI. The screening results released 296 titles on the GS that were not relevant, 4 titles on the IEEE and 38 titles on the GS that were not written in English, and 15 titles on the GS that did not have a DOI. The screening process left 28 titles on the IEEE, 16 titles on the ACM-DL, and 114 titles on the GS.

Furthermore, the selection is again carried out in the introduction and conclusion sections to see the relevance of the topics discussed. The results of the screening resulted in 2

IEEE articles and 40 articles for which the pdf document was not available or could not be downloaded. In addition, the selection process issued 3 IEEE articles and 12 GS articles which were not primary studies. So, the selection process left 22 IEEE articles, 14 ACM-DL articles, and 44 GS articles.

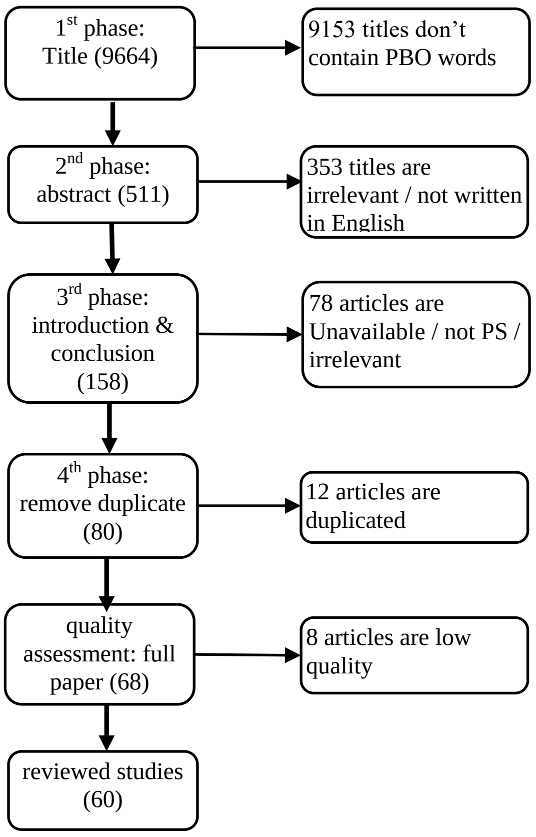
1st phase: Title (9664)
9153 titles don't contain PBO words
2nd phase: abstract (511)
353 titles are irrelevant / not written in English
3rd phase: introduction & conclusion (158)
78 articles are Unavailable / not PS / irrelevant
4th phase: remove duplicate (80)
12 articles are duplicated
quality assessment: full paper (68)
8 articles are low quality
reviewed studies (60)

Fig. 2. Selection process.

The selection process was continued to remove duplicated articles. The selection process issued 1 article in the ACM-DL that had the same title and documents in the IEEE. In addition, the screening process issued 12 articles on the GS that had the same title and documents in the IEEE and ACM-DL. The final selection process left 22 IEEE articles, 13 ACM-DL articles, and 33 GS articles. So that the final total of articles that will be assessed for quality is 68 articles.

Article quality assessment excluded 8 articles that are low quality (see Table III). The final quality assessment process leaves articles that meet a minimum score of 20 IEEE articles, 12 ACM-DL articles, and 28 GS articles. So that the final total of articles reviewed amounted to 60 articles.

TABLE III: ARTICLE QUALITY ASSESSMENT SCORE

No.	Source	Q1	Q2	Quality
1	[1]	Y	Y	High
2	[2]	Y	Y	High
3	[3]	P	Y	Medium
4	[4]	Y	Y	High
5	[8]	Y	Y	High
6	[9]	Y	Y	High
7	[10]	Y	Y	High
8	[13]	P	P	Low
9	[11]	Y	P	Medium
10	[14]	Y	P	Medium
11	[15]	Y	Y	High
12	[16]	P	P	Low
13	[17]	P	Y	Medium
14	[18]	Y	Y	High
15	[19]	P	P	Low
16	[20]	Y	Y	High
17	[21]	Y	Y	High
18	[22]	Y	Y	High
19	[23]	P	Y	Medium
20	[24]	Y	Y	High
21	[25]	Y	Y	High
22	[26]	Y	Y	High
23	[27]	Y	Y	High
24	[28]	Y	Y	High
25	[29]	Y	P	Medium
26	[30]	Y	Y	High
27	[31]	P	P	Low
28	[32]	Y	P	Medium
29	[33]	Y	Y	High
30	[34]	Y	Y	High
31	[35]	Y	Y	High
32	[36]	P	Y	Medium
33	[37]	Y	Y	High
34	[38]	Y	Y	High
35	[39]	Y	Y	High
36	[40]	P	Y	Medium
37	[41]	P	P	Low
38	[42]	Y	Y	High
39	[43]	Y	P	Medium
40	[44]	P	Y	Medium
41	[45]	Y	P	Medium
42	[46]	Y	Y	High
43	[47]	P	Y	Medium
44	[48]	Y	P	Medium
45	[49]	Y	P	Medium
46	[50]	P	Y	Medium
47	[51]	P	P	Low
48	[52]	Y	Y	High
49	[53]	Y	Y	High
50	[54]	Y	Y	High
51	[55]	P	P	Low
52	[56]	Y	Y	High
53	[57]	Y	Y	High
54	[58]	P	Y	Medium
55	[59]	Y	Y	High
56	[60]	P	Y	Medium
57	[61]	Y	Y	High
58	[62]	Y	Y	High
59	[63]	P	Y	Medium
60	[64]	Y	P	Medium
61	[65]	Y	Y	High
62	[66]	Y	Y	High
63	[67]	P	Y	Medium
64	[68]	Y	Y	High
65	[69]	P	P	Low
66	[70]	Y	Y	High
67	[71]	Y	Y	High
68	[72]	P	Y	Medium

III. Results

A. Problem Finding

The problems found were divided into 3 main aspects, as done by Kebritchi and Lipschuetz [73]. The three categories are problems related to students, problems related to content and technology, and problems related to lecturers (see Fig. 3).

1) Problems related to students

Regarding students, the literature review reveals that these aspects are categorized into problems of Cost, Background and Student Learning Experiences, Learning Motivation, Level of Comprehension, and Problem-Solving Skills (see Table IV).

TABLE IV: Problems Related to Students

Problem	Source
Cost (S1)	[4]
Background and Student Learning Experiences (S2)	[1, 14, 15, 17, 18, 20–22]
Learning Motivation (S3)	[11, 23–26]
Level of Comprehension (S4)	[27–30, 32–34]
Problem Solving Skills (S5)	[35–37]

Cost. For most students, learning to program is a very expensive task and can slow down the learning process. The consequences of this delay increase the difficulty of understanding advanced programming concepts [4, 38].

Background and Student Learning Experiences. Most students from different backgrounds find it difficult to learn and master the OOP concept [1]. A programming paradigm that is difficult to break [14, 20]. For first-year students, it is difficult to conceptualize the way of thinking and related information needed to be successful in their studies [17]. For novice programmers, learning the concept of OOP is often daunting due to its abstract nature [18]. Of course, they find it difficult to understand and apply large-scale program structures [15], when they have to struggle with aspects of the computer science curriculum [21]. For novice programmers, both basic logic and simple OOP programs are challenging tasks [22].

Learning Motivation. Student enthusiasm often drops dramatically in the second year due to difficulties in learning programming [24, 25, 39]. The motivation of some students does not arise because of their passive learning style and lack of understanding of previous programming classes [11]. This has an impact on the non-fulfillment of industry requirements in the programming field. Even though there is a need and hope for a prospective workforce from the programming field of education [23].

Level of Comprehension. Students who are learning OOP for the first time at the end of the lecture only master the basic material and lack understanding of software engineering ideas [27–30], [34]. They are required to build their mental models to overcome misperceptions [32]. Students' active learning abilities, self-management, and self-discipline are not yet strong [33].

Problem-Solving Skills. There are many symptoms at the K-12 education level in applying the concept of problem-solving [35]. Several competencies, especially digital skills, are seen as crucial for the development of personal abilities [36]. Students cannot solve complex problems, so they are replaced with basic concepts to build higher cognitive understanding [37].

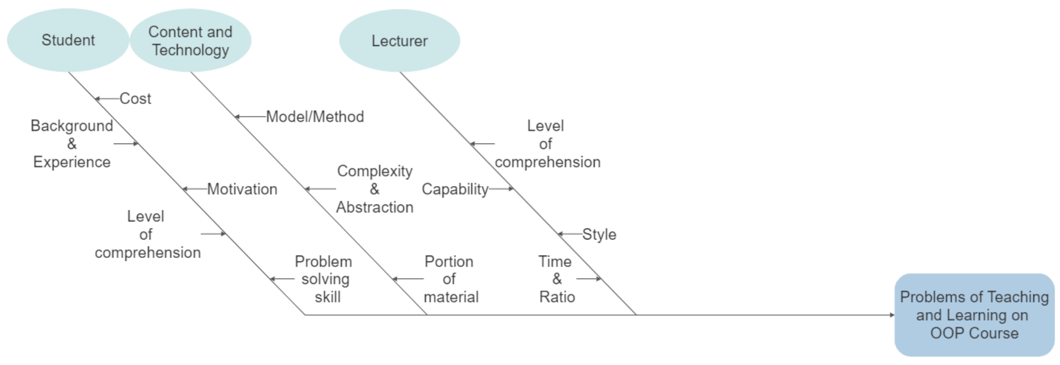

Fig. 3. Problem diagram in OOP learning.

2) Problems related to content and technology

Related to content and technology, this aspect was categorized into Learning Models or Methods, Complexity and Abstraction of Material, and the Portion of Material. See Table V.

TABLE V: Problems Related to Content and Technology

Problem	Source
Learning Model/Method (C1)	[3, 8, 9, 28, 35, 36, 40, 42–49]
Material Complexity and Abstraction (C2)	[10, 18, 20, 22, 24, 34, 37, 46, 50, 52–54, 56–62]
Material Portion (C3)	[28, 25, 33, 61, 63]

Learning Models or Methods. Traditional learning methods do not adequately assist students in understanding the OOP paradigm [8, 9, 49] and fail to relate computational concepts to various student interests [35]. For weak students, the single instructor method is very difficult to arouse the creativity of each student [42]. Learning models based on dependence in closed classrooms trigger cognitive conflicts when students have to undergo a transition from a procedural paradigm to an object-oriented paradigm [48]. Since learning to program is a substantial cognitive challenge, today's MOOC (Massive Open Online Course) runs the risk of over-tensioning students, leaving them frustrated before starting their studies [40]. There is no uniform method for teaching OOP, so lecturers usually have to experiment a lot to find more effective ways to introduce object-oriented concepts and techniques [46]. The latest learning methods do not determine the correlation between performance and programming skills, which is the essence of learning quality-oriented programming languages [28]. Upper-level students sometimes need to correct inaccurate models from their previous work [47] with knowledge of rules that do not meet the quality attributes [3]. There is often a difference between the methodology and focus of programming instruction on the one hand, and the development of programming languages, development tools, and development methodologies on the other [44]. Several innovative ideas should be introduced from time to time [43], one of which is exploring game-based learning in the classroom [45].

Material Complexity and Abstraction. For most students, programming is a very difficult task [59, 61], because it requires high metacognitive skills such as abstraction, deep comprehension, tenacity, and skills to complete the stages of problem-solving and programming [10]. One of the challenges faced by students is not understanding the concept of OOP [24]. OOP requires a high understanding of abstract concepts, as well as the use of these concepts with advanced programming methods [54]. OOP is difficult for students to learn, especially novice students, because both basic concepts (objects and classes) and fundamental concepts (encapsulation, inheritance, and polymorphism) are abstract [18, 50, 56, 62], is difficult to describe [15, 20, 52], and goes beyond just understanding definitions [58]. Some of these difficulties include the fact that OOP translates real-world objects into object-oriented code [57], sometimes lecturers do not have enough experience with the OOP concept on how to teach the abstraction of the concept to students [53], and many applications are designed to make objects interact with each other, which is different from the procedural programming style [22]. Of course, the transition from a procedural programming paradigm to an object-oriented paradigm is a challenge in itself [37]. Due to its complexity and abstraction, the OOP concept needs innovative solutions, which can serve as a means to study the concepts [60]. There is no uniform method for teaching OOP, so lecturers have to experiment a lot to find better and more effective ways to introduce OOP [46]. In other words, teaching OOP concepts and other advanced topics always has a high level of complexity so there is a potential for many students to fail this course [34].

Material Portion. Programming courses have a very large portion, so if students experience problems in this series of courses, students will likely fail [25, 28]. The OOP course contains a lot of material with high difficulty, but only a few hours are available for the course [33]. Students often have difficulty learning how to program in an object-oriented style because object-oriented languages require programmers to be familiar with a large number of non-trivial concepts, to write even the simplest programs [61, 63].

3) *Problems related to lecturer*

The literature review also reveals that problems related to lecturers are categorized into Level of Comprehension, Lecturer Capabilities, Teaching Styles, and Availability of Time & the Lecturer-Student Ratio, see Table VI.

TABLE VI: PROBLEMS RELATED TO A LECTURER

Problem	Source
Level of Comprehension (L1)	[2, 64]
Lecturer Capability (L2)	[49, 53, 65]
Teaching Style (L3)	[32, 36, 62]
Availability of Time & Lecturer-Student Ratio (L4)	[10, 48, 65-67]

Level of Comprehension. Although OOP is considered suitable for large and complex software constructions, lecturers find teaching OOP to be difficult and therefore students also find it difficult to understand it [2]. In massive courses, lecturers cannot follow the practice part well, therefore automation of the knowledge transfer process to students must be used to direct students in the right direction [64].

Lecturer Capability. Non-computer science lecturers find it difficult to find effective ways to teach programming and are not at all trained to become computer science lecturers [65]. Sometimes instructors may not have enough experience in the field to teach often abstract problems convincingly [53]. There is no information available about which features lecturers deem necessary to improve the learning process [49].

Teaching Style. Lecturers usually only use laptops and write code practically so that students are more bored and less enthusiastic about studying OOP courses [62]. Lecturers need to be aware of student perceptions to teach more effectively and provide appropriate interventions in the future [32]. Digital skills are becoming increasingly important in modern society and lecturers are required to adapt to these skills in teaching [36].

Availability of Time & Lecturer-Student Ratio. The transition between paradigms (procedural - Object Oriented) leads to cognitive conflicts, which take more time to resolve [48]. Limited time, limited resources, unbalanced faculty-student ratio, and student diversity, all of which make it difficult for a lecturer to design lessons that meet specific learning needs [10, 65]. Scoring takes a lot of time and effort to manually print hundreds of student source codes so the burden on lecturers becomes even greater [66], even though lecturers have to examine many programs in one period and provide assessments with different criteria [67].

B. *Finding Proposed Solutions*

The literature review also reveals the solutions offered by researchers. The solutions are categorized into 7 categories: Learning Models, Learning Strategies, Learning Methods, Tools or Technology, Learning Environments and Platforms, Assessment Techniques, and Instructional Designs, see Table VII and Fig. 4.

Learning Model. Wong and Yatim [24] proposed the development of game-based learning referring to a game-based framework that can be used as a learning medium in OOP courses. Zaw and Funabiki *et al.* [56] developed a web-based JAVA Programming Learning Assistance System (JPLAS) that facilitates students writing coding assignments. Ribeiro and Bittencourt [48] tried to use project-based learning (PBL). Krugel and Hubwieser [40] developed a MOOC with the name “LPBO” which presents an introduction to the concepts of computational thinking and friendly object-oriented concepts before entering the programming section. Bouali and Nygren *et al.* [2] developed a Virtual Reality (VR) based learning game to support OOP concept learning. Xie and Wu *et al.* [33] designed a blended learning model by combining paired classes, assignment assignments, SPOC, and online evaluation. Boudia and Bengueddach *et al.* [10] demonstrated the impact and effectiveness of collaborative strategies on the learning process. Jusuf and Ibrahim *et al.* [25] implemented a hybrid learning model and compared student interactions in the classroom with the traditional model. Seng and Mohamad Yatim *et al.* [9] adopted a game-based learning model to increase student understanding.

Strategy and Approach. Niculescu and Şerban *et al.* [3] proposed a cyclic learning approach to introduce OOP and analyze its impact. You *et al.* [14] and Batur [29] tried to find an applicable design approach to identify students' conceptions and mental models concerning the used eIDE. Batur and Brinda [39] developed a design approach to identify students' mental conceptions and models about certain digital games. Tanielu and Akau'ola *et al.* [18] developed a systematic approach to creating interactive learning activities by combining analogies and visualizations without students feeling distracted or overwhelmed by the technicalities inherent in textual codes. Dlamini and Leung [65] Developed an Adaptive Pedagogical Model (APM) which can improve teaching strategies through machine learning to assess learning preferences for different types of students. Ardiana and Loekito [11], Çubukçu and Wang *et al.* [68] used the concept of gamification to increase student motivation during class sessions. Wong and Hayati *et al.* [8] developed a mobile game that is by the game-based learning design approach model for students to learn object-oriented programming paradigms. Muyan-Özçelik [23] developed a cross-platform mobile programming approach to introduce two important software engineering topics, namely OOP concepts and design patterns. Gabaruk and Logofatu *et al.* [46] applied a java and OOP teaching approach using the Children Board Games (CBG). Passerini and Lombardi [61] developed a procrastination approach that introduces the concepts of class and instantiation after a few weeks. Lokare *and* Jadhav *et al.* [42] applied for a logic development program in the OOP (C++) course. Tylman [43] discussed the striking similarities between influential philosophical concepts of the past and approaches currently used in certain fields of computer science and tried to gained into Plato's foresee about OOP. Kanaki and Kalogiannakis [57] suggested determining the right starting point for teaching basic object-oriented concepts and marking the appropriate educational tools.

TABLE VII: SOLUTION FINDING

Category	Source
Learning Model (SL1)	[2, 9, 10, 24, 25, 33, 40, 48, 56]
Strategy and Approach (SL2)	[3, 8, 11, 14, 18, 23, 29, 39, 42, 43, 46, 57, 61, 65, 68]
Learning Method (SL3)	[26–28, 32, 44, 54]
Tools or Technology (SL4)	[1, 4, 15, 20–22, 30, 34, 36–38, 52, 53, 59, 60, 62, 64, 70]
Environment and Platform (SL5)	[45, 49, 50, 58, 71]
Assessment Technique (SL6)	[17, 63, 66, 67]
Instructional Design (SL7)	[35, 47, 72]

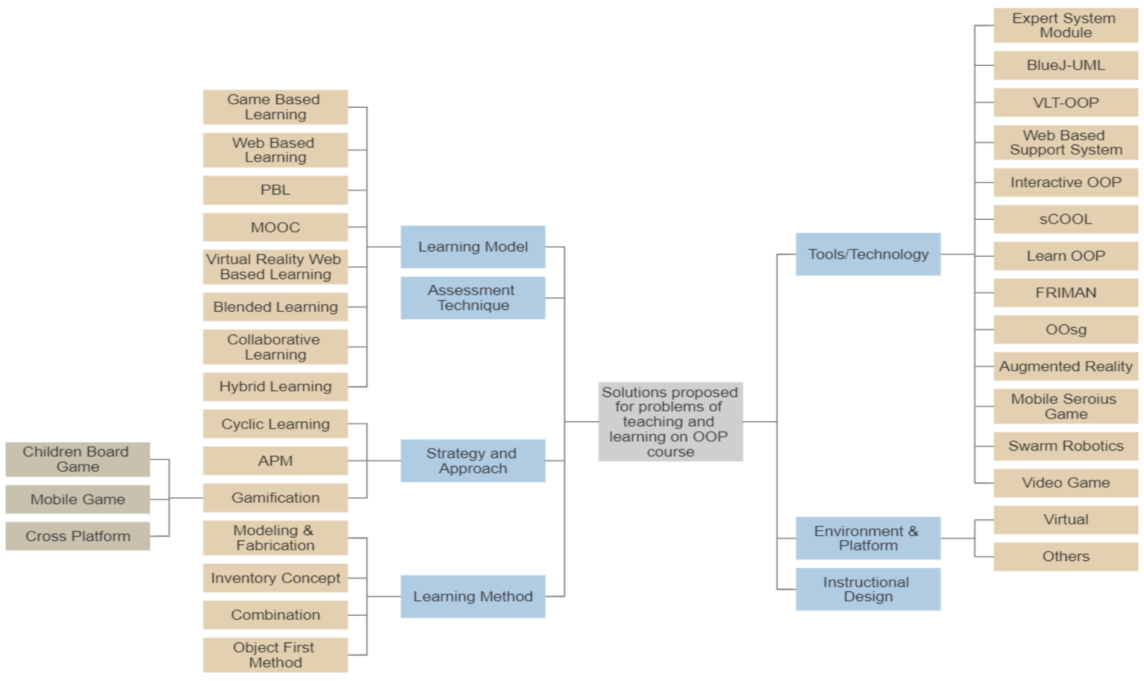

Fig. 4. Various solutions to OOP course learning problems.

Learning Methods. Oliveira and Bonacin [54] Developed innovative methods for planning and implementing OOP learning activities with the support of digital modeling and fabrication based on instructional design concepts and organizational semiotics. Julie *et al.* [32] developed a dedicated Inventory Concept (CI) method for OOP. Xie *et al.* [27] Developed teaching methods by combining SPOC, Flipped Classroom, and project-based approaches. Sun *et al.* [28] proposed a quantitative assessment model of student success in programming skills. Fojtik [44] using the

Object-First Methodology. Huri and Orcid *et al.* [26] implemented Collaborative Learning by using Face-To-Face Groups.

Tools or Technology. Silva and Dora [4] developed an expert system module to enhance existing tools and assist students and lecturers in their assignments. Keung and Xiao *et al.* [1] developed BlueJ-UML which extends and enhances the previous BlueJ platform. Su and Hsu [52] designed and developed a Web-based Visualized OOP Learning Tool (VLT-OOP) to facilitate the learning of OOP concepts. Asano and Kagawa [15] developed a web-based support system for OOP exercises aimed at helping students understand OOP design concepts and techniques. Kucera *et al.* [60] Develop interactive educational applications (Interactive OOP). Mosquera and Steinmaurer *et al.* [36] Designing Cool as a mobile game to learn and practice code in a fun way. Torchiano and Bruno [64] Designing an automated programming assignment infrastructure. Ahmed and Amarif [59] Developed an interactive tool called Learn OOP which includes an animated visual model that demonstrates the role of objects in a Java program. Sedlacek *et al.* [22] created the FRIMAN project to simplify the process of learning the OOP paradigm in the Java language. Abbasi and Kazi *et al.* [37] developed a Serious Game (SG) prototype called OOsg to study OOP. Abidin and Zawawi [62] Combining OOP with Augmented Reality (AR). Holmstedt and Mengiste [53] Develop and maintain a system (codebase) for instructors as a tool to support teaching. Shmallo and Ragonis [70] developed a diagnostic tool that is presented in Java and can be translated into other OOP languages. Lotfi *et al.* [20] developed a mobile serious game (MSG) to teach object programming concepts in a fun and easy way. Stovold and Powell [21] used swarm robotics and firefly synchronization algorithm. Lopez *et al.* [34] developed a video game project to teach OOP concepts and other advanced topics.

Environments and Platforms. Thurner [58] develops a virtual learning environment that allows students new to object orientation to represent the qualities of their model, and quickly detect incorrect modeling decisions. Olier *et al.* [50] designed and validated a learning environment to learn object-oriented programming concepts. Passerini and Lombardi *et al.* [71] developed Wollok, which includes an educational language and a dedicated integrated development environment (IDE) designed to study OOP by integrating a pedagogical approach and an industrial environment. Kelter *et al.* [49] Investigating lecturers' perspectives on popular learning and programming environments that are used in secondary computer science education. Thongmak [45] designs a gamification platform that students use to play simple card games and contest their program involving card games with other people.

Assessment Techniques. Grobbelaar [17] develops and tests a special software instrument to influence changes in students' abstract thinking abilities. Barkmin and Kramer *et al.* [63] invented a computer-based instrument and conducted the first study with 42 students. Tateishi and Inoue [66] Developed a system that automatically grades GUI programming exercises submitted by students. Inoue [67] Developed a method for testing and assessing student programs with a graphical user interface written in JavaFX.

Instructional Design. Santos *et al.* [47] apply the concept of bridging for the paradigm shift from functional to object-oriented programming. Zhu and Zha [72] integrates educational concepts and core elements of OBE into the teaching design of OOP courses. Rahman [35] designed a 2-week block course curriculum that teaches the basics of OOP, via App Inventor 2 (AI2) and Java, to students.

IV. Discussion

We have presented the findings of the problem from the literature and grouped the findings into 3 aspects, namely those related to students, related content and technology, and related lecturers. We have also grouped the findings of solutions to problems into 7 categories. The tables below show the frequency of studies that raised issued the problems and solutions.

TABLE VIII: Frequency of Problem Related to Student, Content & Technology, and Lecturer

	Problem	Frequency
Frequency of Problem Related to Student	S1	1 - 4.00%
	S2	**8 – 32.00%**
	S3	6 – 24.00%
	S4	7 – 28.00%
	S5	3 – 12.00%
Frequency of Problem Related to Content & Technology	C1	**15 – 38.46%**
	C2	**19 – 48.72%**
	C3	5 – 12.82%
Frequency of Problem Related to Lecturer	L1	2 – 15.38%
	L2	3 – 23.08%
	L3	3 – 23.08%
	L4	**5 – 38.46%**

TABLE IX: Frequency of Solutions

Solution	Frequency
SL1	9 – 16.98%
SL2	**15 – 25.00%**
SL3	6 – 10.00%
SL4	**18 – 30.00%**
SL5	5 – 8.33%
SL6	4 – 6.67%
SL7	3 – 5.00%

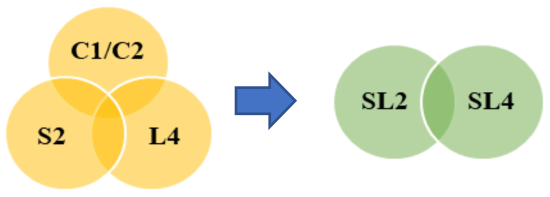

Fig. 5. Relation of the core problems and proposed solutions for teaching and learning on PBO course.

Judging from the frequency of the problems between 2017-2021, it can be said that the trend of discussion by researchers is the problem of Material Complexity and Abstraction (C2) in the OOP course (see Table VIII). As it is known, the way OOP works is by translating real-world objects into code by referring to the properties and characteristics of the object. In other words, anyone who will use OOP as a programming paradigm must first understand the nature and characteristics of the object that will be translated into code. For novice programmers, the difficulty is higher because of the abstract nature of OOP which is difficult to describe and more than just a definition. Meanwhile, for those who are already familiar with the procedural paradigm, the process of transitioning to an object-oriented paradigm is a challenge in itself where they have to change their programming habits and adapt to new habits. The key is that lecturers are required to improve their experience with OOP so that the learning objectives of the course can be achieved properly.

In addition to Material Complexity and Abstraction, another issue that has become a trend for researchers to discuss is the applied Learning Models or Methods (C1). One of the models highlighted is the traditional learning model which does not adequately help students understand the object-oriented paradigm and fails to relate computational concepts to various student interests. This model also difficult to arouse students' creativity and often triggers cognitive conflicts when the process of transitioning from a procedural paradigm to an object-oriented paradigm. In addition, some literature issues the ineffectiveness of models using technology assistance and proposes some improvements. Some models can overcome time constraints but fail to overcome technological problems and vice versa. The models that have been applied also do not fully adopt the needs of the world of work and industry in the field of information technology.

In the student aspect, the trend of discussion is the problem of Background and Student Learning Experiences (S2). The researchers made the low experience of students in the field of programming skills the cause of the low learning achievement. Some of the impacts they found included students finding it difficult to learn and master the concept of OOP at both fundamental levels; students finding it difficult to conceptualize a way of thinking using an object-oriented paradigm; students having difficulty understanding and applying large-scale program structures, and some students feel afraid because of the abstract nature of OOP. Therefore, any solution taken in the future must pay attention to this aspect, see Table VIII.

In the aspect of lecturers, the discussion trend is the issue of Availability of Time & Lecturer-Student Ratio (L4). As it is known that OOP is an abstract program and is very complex, it takes sufficient time for the materials to be fully conveyed. The ratio between lecturers and students that is not ideal automatically takes more time because the lecturer has to control more students both during the learning process and the process of recording grades, see Table VIII.

In general, the above problems have been given a solution by the researchers (see Table VIIII). Most of the literature proposes the development or application of technology (SL4) and learning models (SL2). Some of the proposed technologies are designed and developed in the form of videos, offline or online applications, robotics, websites, and even involving the field of artificial intelligence. Some of the technologies applied can support solving some specific problems. Many learning model developments adopt technology, for example, game-based learning models, web-based learning models, and hybrid learning models. These learning models were also developed, of course, to overcome certain problems. Fig. 5 shows the relation of 4 core problems of teaching and learning in OOP courses that can be solved by 2 main solutions.

The distinction and relation between Models, Approaches, Strategies, Methods, and Techniques of Learning

The learning approach can be interpreted as a starting point or point of view on the learning process, which refers to the view of the occurrence of a process that is still very general, in which it accommodates, inspires, strengthens, and underlies learning methods with a certain theoretical scope. Judging from the approach, there are two types of learning approaches, namely: student-centered approach and teacher-centered approach.

The learning approach that has been determined, is then derived into a learning strategy. Kemp and Rodriguez [74] suggests that a learning strategy is a learning activity that must be done by lecturers and students so that learning objectives can be achieved effectively and efficiently. Sanjaya [75] states that the learning strategy contains the meaning of planning. This means that the strategy is still conceptual about the decisions to be taken in the implementation of learning.

Learning strategies are still conceptual and to implement them, certain learning methods are used. In other words, the strategy is “a plan of operation achieving something” while the method is “a way of achieving something” [75]. So, the learning method can be interpreted as a method used to implement plans that have been prepared in the form of real and practical activities to achieve learning objectives.

Furthermore, the learning method is translated into techniques and learning styles. Thus, learning techniques can be interpreted as the way someone does in implements a specific method.

If the approaches, strategies, methods, and learning techniques have been strung together into a unified whole, what is called a learning model is formed. So, the learning model is a form of learning that is illustrated from beginning to end and is presented specifically by the lecturer. In other words, the learning model is a wrapper or frame from the application of an approach, method, and learning technique, see Fig. 6.

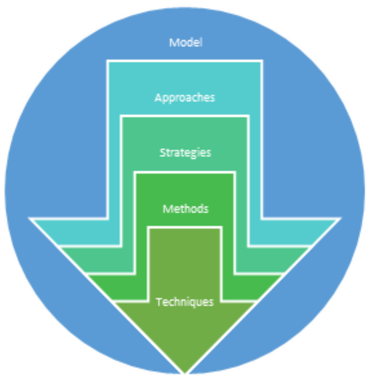

Fig. 6. Relation of model, approaches, strategies, methods and techniques of learning.

V. Conclusion

The research questions that have been raised at the beginning of the study have led this research to find problems in the learning process of Object-Oriented Programming courses. The problems found are divided into 3 main aspects, namely those related to students, related content and technology, and related lecturers. From the many problems found in these three aspects, it is known that the core problems of the current OOP course learning process are the Material Complexity and Abstraction; the Learning Models or Methods; Background, and Student Learning Experiences; and Availability of Time & Lecturer-Student Ratio.

In addition, these research questions have also led this research to find concrete solutions offered as answers to the problems found. There are 7 categories of solutions offered by previous researchers. Of the seven categories, it is known that the majority of researchers offer solutions in the form of developing or implementing technology or learning media. In addition, the majority of researchers also offer the development of learning models.

Future research is expected to contribute to solving problems related to content and technology, especially problems of complexity and abstraction of material as well as the application of learning models in OOP courses. In addition, it is also expected to contribute to solving problems related to students and lecturers, especially the problem of availability of lecturers' time and the background and learning experiences of students. Furthermore, these studies can be focused on solutions by developing learning models accompanied by the application of appropriate learning tools or technology.

Conflict of Interest

The authors declare no conflict of interest.

Author Contributions

Efan conducted the research and wrote the paper; Krismadinata and Jalius Jama verified the manuscript before it was be submitted; Rudi Mulya enhanced the manuscript language before it was be submitted; all authors had approved the final version.

Funding

This work was supported in part by the Institut Teknologi Pagar Alam, Indonesia

References

[1] J. Keung, Y. Xiao, Q. Mi and V. C. S. Lee, “BlueJ-UML: Learning object-oriented programming paradigm using interactive programming environment,” in *Proc. 2018 International Symposium on Educational Technology (ISET)*, 2018, pp. 47-51.

[2] N. Bouali, E. Nygren, S. S. Oyelere, J. Suhonen, and V. Cavalli-Sforza, “Imikode: A VR game to introduce OOP concepts,” in *Proc. 19th Koli Calling International Conference on Computing Education Research (Koli Calling '19)*, 2019, pp. 1-2.

[3] V. Niculescu, C. Şerban and A. Vescan, “Does cyclic learning have positive impact on teaching object-oriented programming?” in *Proc. 2019 IEEE Frontiers in Education Conference (FIE)*, 2019, pp. 1-9.

[4] V. Silva and F. A. Dora, “An automatic and intelligent approach for supporting teaching and learning of software engineering considering design smells in object-oriented programming,” in *Proc. 2019 IEEE 19th International Conference on Advanced Learning Technologies (ICALT)*, 2019, pp. 321–323, vol. 2161–377X.

[5] E. Lotfi and B. Mohammed, “Teaching object oriented programming concepts through a mobile serious game,” in *Proc. 3rd International Conference on Smart City Applications (SCA '18)*, 2018, pp. 1-6.

[6] A. M. Reyna *et al.*, “Object-oriented programming as an alternative to industrial control,” in *Proc. 9th International Conference on Electrical Engineering, Computing Science and Automatic Control (CCE)*, 2012, pp. 1–7.

[7] S. Butler, M. Wermelinger, Y. Yu, and H. Sharp, “Mining java class naming conventions,” in *Proc. 2011 27th IEEE International Conference on Software Maintenance (ICSM)*, 2011, pp. 93-102.

[8] Y. S. Wong, I. M. Hayati, M. Yatim and T. W. Hoe, “A propriety game based learning mobile game to learn object-oriented programming — Odyssey of Phoenix,” in *Proc. 2017 IEEE 6th International Conference on Teaching, Assessment, and Learning for Engineering (TALE)*, 2017, pp. 426-431.

[9] W. Yoke Seng, M. H. Mohamad Yatim, and T. Wee Hoe, “Learning object-oriented programming paradigm via game-based learning game — Pilot study,” *The International journal of Multimedia & Its Applications*, vol. 10, no. 06, pp. 181–197, Dec. 2018.

[10] C. Boudia, A. Bengueddach, and H. Haffaf, “Collaborative strategy for teaching and learning object-oriented programming course: A case study at Mostafa Stambouli Mascara University, Algeria,” *Informatica (Slovenia)*, vol. 43, no. 1, pp. 129–144, 2019.

[11] D. P. Y. Ardiana and L. H. Loekito, “Gamification design to improve student motivation on learning object-oriented programming,” *Journal of Physics: Conference Series*, vol. 1516, no. 1, pp. 1-8, 2020.

[12] B. Kitchenham and S. Charters, *Guidelines for performing Systematic Literature Reviews in Software Engineering*, UK: Keele University & University of Durham, 2007.

[13] E. Kaila, E. Kurvinen, E. Lokkila, and M. J. Laakso, “Redesigning an object-oriented programming course,” *ACM Transactions on Computing Education*, vol. 16, no. 4, pp. 1–21, 2016.

[14] X. You, C. Xiong, and P. Zhang, “Brief discuss the application of object-oriented in java language programming course,” in *Proc. 2018 3rd International Conference on Automation, Mechanical and Electrical Engineering (AMEE 2018)*, 2018, pp. 544-548.

[15] Y. Asano and K. Kagawa, “Development of a web-based support system for object oriented programming exercises with graphics programming,” in Proc. *2019 18th International Conference on Information Technology Based Higher Education and Training (ITHET)*, 2019, pp. 1–4.

[16] E. A. Kalinga, “Learning software development through modeling using object oriented approach with unified modeling language: A case of an online interview system,” *Journal of Learning for Development*, vol. 8, no. 1, pp. 74–92, 2021.

[17] L. Grobbelaar, “The effects of a software artefact designed to stimulate abstract thinking ability on the academic performance in object oriented programming of first year information technology students,” in *Proc. 2018 International Conference on Intelligent and Innovative Computing Applications (ICONIC)*, 2018, pp. 1–8.

[18] T. Tanielu, R. Akau’ola, E. Varoy, and N. Giacaman, “Combining analogies and virtual reality for active and visual object-oriented programming,” in *Proc. ACM Conference on Global Computing Education (CompEd '19)*, 2019, pp. 92–98.

[19] C. C. Lin, Z. C. Liu, C. L. Chang, and Y. W. Lin, “A genetic algorithm-based personalized remedial learning system for learning object-oriented concepts of java,” *IEEE Transactions on Education*, vol. 62, no. 4, pp. 237–245, Nov. 2019.

[20] E. Lotfi, O. Bakkali Yedri, and M. Bouhorma, “Towards a mobile serious game for learning object oriented programming paradigms,” *Innovations in Smart Cities Applications*, Switzerland: Springer, 2019, pp. 450–462.

[21] J. Stovold and S. Powell, “Teaching object-oriented programming in secondary schools using swarm robotics,” *Educational Robotics in the Context of the Maker Movement*, Switzerland: Springer, 2020, vol. 946, pp. 201–204.

[22] P. Sedlacek, M. Kvet, and M. Vaclavkova, “Development of FRIMAN: Supporting tool for object oriented programming teaching,” *Open Computer Science*, vol. 11, pp. 90–98, 2021.

[23] P. Muyan-Öz çelik, “A hands-on cross-platform mobile programming approach to teaching OOP concepts and design patterns,” in *Proc. 2017 IEEE/ACM 1st International Workshop on Software Engineering Curricula for Millennials (SECM)*, 2017, pp. 33-39.

[24] Y. S. Wong and M. H. M. Yatim, “A propriety multiplatform game-based learning game to learn object-oriented programming,” in

Proc. 2018 7th International Congress on Advanced Applied Informatics (IIAI-AAI), 2018, pp. 278–283.

[25] H. Jusuf, N. Ibrahim, and A. Suparman, "Developing a hybrid learning strategy for students' engagement in object-oriented programming course," *Universal Journal of Educational Research*, vol. 7, no. 9 A, pp. 78–87, 2019.

[26] M. Huri, B. Orcid, B. English, and S. T. Orcid, "Face-to-face collaborative learning groups," *Online Journal of Distance Education*, no. July, pp. 0–3, 2019.

[27] S. Xie, L. Fan, and W. Wu, "Teaching research of object oriented programming course based on SPOC and project-driven," in *Proc. 2nd International Conference on Digital Technology in Education (ICDTE 2018)*, 2018, pp. 53–57.

[28] Q. Sun, J. Wu, and K. Liu, "How are students' programming skills developed: an empirical study in an object-oriented course," in *Proc. ACM Turing Celebration Conference (ACM TURC '19)*, 2019, pp. 1–6.

[29] F. Batur, "How does an educational IDE influence students' conceptions of object-oriented programming?: Design of a Ph.D. research project to explore secondary school students' conceptions of OOP," in *Proc. 14th Workshop in Primary and Secondary Computing Education (WiPSCE'19)*, 2019, pp. 1–2.

[30] M. Amarif and S. Ahmed, "The effect of visualizing role of variable in object oriented programming understanding," *Computer Science & Information Technology (CS & IT)*, vol.9, no. 2, pp. 11–20, 2019.

[31] Q. Sun, J. Wu, and K. Liu, "Toward understanding students' learning performance in an object-oriented programming course: The perspective of program quality," *IEEE Access*, vol. 8, pp. 37505–37517, 2020.

[32] H. Julie, D. Bruno, H. Patrick and L. Tony, "Object-oriented programming: Diagnosis understanding by identifying and describing novice perceptions," in *Proc. 2020 IEEE Frontiers in Education Conference (FIE)*, 2020, pp. 1-5.

[33] S. Xie, W. Wu, C. Hu, and L. Fan, "Reform of object oriented programming based on task driven and blended teaching," in *Proc. 2020 the 4th International Conference on Education and E-Learning (ICEEL 2020)*, 2020, pp. 38-41.

[34] M. A. López, E. V. Duarte, and E. C. Gutiérrez, "Experience in teaching object-oriented programming and advanced topics of programming through the development of a video game project," *CEUR Workshop Proc*, vol. 2747, pp. 109–118, 2020.

[35] F. Rahman, "From app inventor to Java: Introducing object-oriented programming to middle school students through experiential learning," *ASEE Annual Conference and Exposition, Conference Proceedings*, vol. 2018-June, 2018.

[36] C. K. Mosquera, A. Steinmaurer, C. Eckhardt, and C. Guetl, "Immersively learning object oriented programming concepts with sCool," in *Proc. 6th International Conference of the Immersive Learning Research Network, iLRN 2020*, 2020 no. June, pp. 124–13.

[37] S. Abbasi, H. Kazi, A. W. Kazi, K. Khowaja, and A. Baloch, "Gauge object oriented programming in student's learning performance, normalized learning gains and perceived motivation with serious games," *Information (Switzerland)*, vol. 12, no. 3, pp. 1–21, 2021.

[38] M. D. B. Castro, G.M. Tumibay, "A literature review : efficacy of online learning courses for higher education institution using meta-analysis," *Educ Inf Technol*, vol 26, pp. 1367–1385, 2021.

[39] F. Batur and T. Brinda, "Students' conceptions of object-oriented programming in the context of game designing in computing education," in *Proc. the 52nd ACM Technical Symposium on Computer Science Education*, 2021, p. 1290.

[40] J. Krugel and P. Hubwieser, "Computational thinking as springboard for learning object-oriented programming in an interactive MOOC," *IEEE Global Engineering Education Conference,* no. November, pp. 1709–1712, 2017.

[41] A. M. Brito Jr. and A. A. D. Medeiros, "A motivating approach to introduce object-oriented programming to engineering students," *The International Journal of Electrical Engineering & Education*, pp. 1-10, 2019.

[42] V. T. Lokare, P. M. Jadhav, and S. S. Patil, "An integrated approach for teaching object oriented programming (C++) course," *Journal of Engineering Education Transformations*, vol. 31, no. 3, pp. 17–23, 2018.

[43] W. Tylman, "Computer science and philosophy: Did plato foresee object-oriented programming?" *Found Sci*, vol. 23, no. 1, pp. 159–172, 2018.

[44] R. Fojtík, "Teaching of object-oriented programming," in *Proc. the 12th International Scientific Conference on Distance Learning in Applied Informatics.*, 2018, pp. 273–282.

[45] M. Thongmak, "Creating gameful experience in the object-oriented programming classroom: A case study," *Online Journal of Applied Knowledge Management*, vol. 6, no. 1, pp. 30-53, 2018.

[46] J. Gabaruk, D. Logofătu, D. Groskreutz, and C. Andersson, "On teaching java and object oriented programming by using children board games," in *Proc. 2019 IEEE Global Engineering Education Conference (EDUCON)*, 2019, pp. 601–606.

[47] I. Moreno Santos, M. Hauswirth, and N. Nystrom, "Experiences in bridging from functional to object-oriented programming," in *Proc. 2019 ACM SIGPLAN Symposium on SPLASH-E (SPLASH-E 2019)*, 2019, pp. 36–40.

[48] A. L. Ribeiro and R. A. Bittencourt, "A PBL-based, integrated learning experience of object-oriented programming, data structures and software design," in *Proc. 2018 IEEE Frontiers in Education Conference (FIE)*, 2018, pp. 1–9.

[49] R. Kelter, M. Kramer, and T. Brinda, "Teachers' perspectives on learning and programming environments for secondary education," in *Proc. Open Conference on Computers in Education (OCCE)*, no. July, 2019, pp. 47-55.

[50] A. J. Olier, A. A. Gómez, and M. F. Caro, "Design and implementation of a teaching tool for introduction to object-oriented programming," *IEEE Latin America Transactions*, vol. 15, no. 1, pp. 97–102, 2017.

[51] X. Zhang, J. D. Crabtree, M. G. Terwilliger, and T. T. Redman, "Assessing students' object-oriented programming skills with java: the 'department-employee' project," *Journal of Computer Information Systems*, vol. 60, no. 3, pp. 274–286, May 2020.

[52] J.-M. Su and F.-Y. Hsu, "Building a visualized learning tool to facilitate the concept learning of object-oriented programming," in *Proc. 2017 6th IIAI International Congress on Advanced Applied Informatics (IIAI-AAI)*, 2017, pp. 516-520.

[53] V. Holmstedt and S. A. Mengiste, "Zirr — Why and how practice impacts confidence in introductory object oriented programming courses," in *Proc. Fifth International Conference Modelling and Development of Intelligent Systems*, 2017, pp. 29–42.

[54] G. Oliveira and R. Bonacin, "A method for teaching object-oriented programming with digital modeling," in *Proc. 2018 IEEE 18th International Conference on Advanced Learning Technologies (ICALT)*, 2018, pp. 233–237.

[55] J. Udvaros, "Teaching object oriented programming by visual devices," in *Proc. eLearning and Software for Education Conference*, 2019, pp. 407–413.

[56] K. K. Zaw, N. Funabiki, E. E. Mon, and W.-C. Kao, "An informative test code approach for studying three object-oriented programming concepts by code writing problem in java programming learning assistant system," in *Proc. 2018 IEEE 7th Global Conference on Consumer Electronics (GCCE)*, 2018, pp. 629–633.

[57] K. Kanaki and M. Kalogiannakis, "Introducing fundamental object-oriented programming concepts in preschool education within the context of physical science courses," *Educ Inf Technol (Dordr)*, vol. 23, 2018.

[58] V. Thurner, "Fostering the comprehension of the object-oriented programming paradigm by a virtual lab exercise," in *Proc. 2019 5th Experiment International Conference (exp.at'19)*, 2019, pp. 137–142.

[59] S. Ahmed and M. Amarif, "An interactive animation tool for java object oriented programming understanding," *International Journal of Programming Languages and Applications*, vol. 9, no. 3, pp. 1–17, 2019.

[60] E. Kucera, O. Haffner, and R. Leskovsky, "Multimedia application for object-oriented programming education developed by unity engine," in *Proc. 30th International Conference on Cybernetics and Informatics, K and I 2020*, no. March, 2020.

[61] N. Passerini and C. Lombardi, "Postponing the concept of class when introducing OOP," in *Proc. 2020 ACM Conference on Innovation and Technology in Computer Science Education (ITiCSE '20)*, 2020, pp. 152–158.

[62] Z. Z. Abidin and M. A. A. Zawawi, "OOP-AR: Learn object oriented programming using augmented reality," *International Journal of Multimedia and Recent Innovation*, vol. 2, no. 1, pp. 60–75, 2020.

[63] M. Barkmin, M. Kramer, D. A. Tobinski, and T. Brinda, "Code structure difficulty in OOP: An exploration study regarding basic cognitive processes," in *Proc. 17th Koli Calling International Conference on Computing Education Research*, 2017, pp. 185-186.

[64] M. Torchiano and G. Bruno, "Integrating software engineering key practices into an OOP massive in-classroom course: An experience report," in *Proc. International Conference on Software Engineering*, no. June, pp. 64–71, 2018.

[65] M. Dlamini and W. Leung, "Enhancing object-oriented programming pedagogy with an adaptive intelligent tutoring system," in *Proc. 47th*

Annual Conference of the Southern African Computer Lecturers' Association, SACLA 2018, 2019, pp. 269–284.

[66] Y. Tateishi and U. Inoue, "GUI static testing for object-oriented programming exercises," in *Proc. 2019 20th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD)*, 2019, pp. 280–285.

[67] U. Inoue, "GUI Testing for introductory object-oriented programming exercises," *Computational Science/ Intelligence & Applied Informatics*, vol. 787, pp. 1–13, 2019.

[68] Ç. Çubukçu, B. Wang, L. Goodman *et al*., "Gamification for assessment of object oriented programming," in *Proc. ICICTE 2017*, 2017, pp. 226–237.

[69] Y. Wu, "Object-oriented programming course reform using python language in the background of artificial intelligence," in *Proc. 019 3rd International Conference on Education, Management Science and Economics (ICEMSE 2019)*, 2019, pp. 93-96.

[70] R. Shmallo and N. Ragonis, "Understanding the 'this' reference in object oriented programming: Misconceptions, conceptions, and teaching recommendations," *Educ Inf Technol (Dordr)*, vol. 26, no. 1, pp. 733–762, 2021.

[71] N. Passerini, C. Lombardi, J. Fernandes, P. Tesone, and F. Dodino, "Wollok: Language + IDE for a gentle and industry-aware introduction to OOP," in *Proc. 12th Latin American Conference on Learning Objects and Technologies (LACLO 2017)*, 2017, pp. 1–4.

[72] Q. ZHU and Y. ZHA, "Teaching reform of object-oriented programming course based on OBE," in *Proc. 2018 3rd International Conference on Education, Management and Systems Engineering (EMSE 2018)*, 2018.

[73] M. Kebritchi, A. Lipschuetz, and L. Santiague, "Issues and challenges for teaching successful online courses in higher education," *Journal of Educational Technology Systems*, vol. 46, no. 1, pp. 4–29, 2017.

[74] J. E. Kemp and L. Rodriguez, "The basics of instructional design," *J Contin Educ Nurs*, vol. 23, no. 6, 1992.

[75] W. Sanjaya, *Strategi pembelajaran berorientasi standar proses pendidikan*, 1st ed. Jakarta: Prenada Media, 2011.

Efan received the S.Kom. degree from Bina Darma University, Palembang, Indonesia, in 2007 and the M.Kom. degree from the University of Putera Indonesia YPTK, Padang, Indonesia, in 2013. Since 2017, He has continued doctoral college in Universitas Negeri Padang, Indonesia.

He is currently a lecturer of the Department of Informatics Engineering, Sekolah Tinggi Teknologi Pagaralam (now reformed to Institut Teknologi Pagar Alam), Indonesia.

His research interests include TVET, information system, artificial intelligence, and database.

Krismadinata was born in Padang Indonesia. He received the B.Eng. degree from Universitas Andalas, Padang, Indonesia, in 2000 and the M.Eng. degree from the Institute of Technology Bandung, Indonesia, in 2004 and the Ph.D. degree from the University of Malaya, Kuala Lumpur, Malaysia, in 2012, and post-doctoral at UMPEDAC within 2012–2014.

He was one of awardees ASEAN-INDIA Research Training Fellowship in 2019. He is currently with the Department of Electrical Engineering, Universitas Negeri Padang, where he is also a professional engineer and the director of Center for Energy and Power Electronics Research Universitas Negeri Padang. Dr. Kris has international patents and also as reviewer for many international journals in electrical engineering and energy fields. He is actively involved in consulting on renewable energy projects.

His research interests are power electronics, control system and renewable energy.

Jalius Jama is a professor in Universitas Negeri Padang. He received engineer (Ir.) degree form Institut Keguruan dan Ilmu Pendidikan Yogyakarta, Indonesia and M.Ed. degree from Sam Houston State University, USA. Then he received Ph.D. degree from The OHIO State University, USA. He is currently a lecturer of the Department of Electrical Engineering, Universitas Negeri Padang, Indonesia.

Rudi Mulya was born in Padang. He received B.Eng. degree from Department of Electrical Engineering Universitas Andalas Padang in 2008. Then He received M.Kom. from Universitas Putera Indonesia YPTK Padang, Indonesia in 2016. In 2022 he received the Ph.D. degree on Technical and Vocational Education Training program from Universitas Negeri Padang, Indonesia.

He is currently a lecturer of the Department of Electrical Engineering, Universitas Negeri Padang, Indonesia. His research interests include electrical engineering, informatic technology, vocational technology, education.