Educational Programming Environments for Enhancing Conceptual Design in the Object-Oriented Paradigm: A Systematic Mapping Study

Abstract: Teaching and learning programming, and especially Object-Oriented Programming (OOP), is a complicated and challenging task. Students have to comprehend various OOP concepts and utilize them for designing object-oriented programs. Various types of educational programming environments, such as microworlds and educational games, have been devised for supporting novices mainly in comprehending OOP concepts. However, such environments do not usually support students in the conceptual design of object-oriented programs of a considerable length and complexity. In this paper, we focus on a systematic mapping study (SMS) of educational programming environments for enhancing the conceptual design in OOP, which relies on modularity, abstraction and encapsulation. The research questions investigate the intended learning outcomes, the empirical evidence on the effectiveness, and the teaching / learning technologies used by educational programming environments for enhancing the conceptual design in OOP. The findings can support instructors in selecting appropriate tools for their courses and researchers in the field of educational programming environments for OOP.

Running head: Educational Programming Environments for Conceptual Design in OOP

Alexandros Tsichouridis
Department of Applied Informatics, School of Information Sciences, University of Macedonia, 156 Egnatia Street, GR-54636, Thessaloniki, Greece
atsichouridis@gmail.com

Stelios Xinogalos [Corresponding author]
Department of Applied Informatics, School of Information Sciences, University of Macedonia, 156 Egnatia Street, GR-54636, Thessaloniki, Greece
Tel.: 00302310891895
stelios@uom.edu.gr
https://orcid.org/0000-0002-9148-7779

Apostolos Ampatzoglou
Department of Applied Informatics, School of Information Sciences, University of Macedonia, 156 Egnatia Street, GR-54636, Thessaloniki, Greece
a.ampatzoglou@uom.edu.gr
https://orcid.org/0000-0002-5764-7302

Keywords: object-oriented design, object-oriented programming, teaching, education, tools

Alexandros Tsichouridis is a BSc and MSc holder of the Department of Applied informatics, University of Macedonia, Greece. He is currently working as a senior software engineer at Deloitte. His main research interests throughout the Sc and MSc thesis include programming teaching, programming educational platforms and software quality improvement through the education progress.

Stelios Xinogalos is a Professor at the Department of Applied Informatics, University of Macedonia, Greece. He is a member of the Software and Data Engineering Lab and the Educational Technology Research Group of the Department of Applied Informatics, University of Macedonia. His research in-

terests include Programming Environments and Techniques, Object-oriented Design and Programming, Didactics of Programming, Computer Science Education, Educational Technology, and Serious Games. He has published more than 120 research papers in international journals, conferences and books.

Dr. Apostolos Ampatzoglou is an Associate Professor in the Department of Computer Science of the University of Macedonia, where he carries out research and teaching in the area of software engineering. In the period 2013-2016 he was an Assistant Professor at the Department of Computer Science in the University of Groningen (Netherlands). He holds a BSc in Information Systems (2003), a MSc in Computer Systems (2005) and a PhD in Software Engineering from the Aristotle University of Thessaloniki (2012). His current research interests are focused on technical debt, reverse engineering, software maintainability, software quality management, open source software engineering and software design. He has published more than 120 articles in international journals and conferences.

Introduction

Teaching and learning programming, and especially Object-Oriented Programming (OOP), is a complicated and challenging task. An important aspect in this learning process is to introduce students to the rationale of object-oriented design (Xinogalos, 2015), which relies on three pillars: modularity, abstraction, and encapsulation (Vliet, 2008). In this paper, we focus on the conceptual design, i.e., the process of abstracting from the problem specification and defining the necessary classes that represent the domain of the problem; defining the relationships between the classes; as well as the properties and functions for modeling each entity from the problem domain through classes.

However, while attempting to apply appropriate conceptual design, students face various difficulties (Thomasson et al. 2006; Xinogalos, 2015; Xinogalos, 2016). One of the most important difficulties that students face in their first programming steps is to properly design a solution for a specific problem (Piteira and Costa, 2013; Tan et al., 2009). This problem is derived from the deficiency of skills necessary for solving algorithmic problems by novices, consequently leading to difficulty in decomposing the problem to smaller ones, so as to reach a solution. Moreover, many students find it difficult to understand certain complex programming concepts, such as:

- Pointers and references (Piteira and Costa, 2013). Pointers and references are usually introduced in entry level courses and low-level programming languages. It is important for students to understand such features since higher level programming languages and systems depend on them.
- Data structures and Algorithms (Moraes and Teixeira, 2019; Xinogalos, 2016). One of the most essential things for CS students to learn is at least the basic data structures and algorithms. It is important also to know about the benefits of using them and their storage and computational efficiency. The underlying learning difficulties are connected to the complexity of their logic and execution steps. Teaching data structures and algorithms demands learning material that can reveal their execution details in an understandable and meaningful way (Moraes and Teixeira, 2019).
- OOP concepts. Students face various difficulties and have several misconceptions even for basic OOP concepts. Specifically, students find it difficult to distinguish between classes and objects (Ragonis and Ben-Ari, 2005; Xinogalos, 2005); objects are considered to be mere wrappers of variables (Carter and Fowler, 1998) or database records without behavior (Holland et al., 1997).
- Handling classes. Several difficulties have been recorded in the literature regarding the use of classes, such as (Xinogalos, 2015): difficulty in comprehending that a class models an entity in the program domain (Eckerdal and Thuné, 2005); a class is viewed as a collection of objects instead of an abstraction (Ragonis and Ben-Ari, 2005); difficulty in writing programs with multiple classes (Carter and Fowler, 1998) or composed classes (Ragonis and Ben-Ari, 2005).
- Design patterns (Denegri et al. 2008; Azimullah et al. 2020). Another important topic that is tightly

related to OOP is the learning difficulties that students face with design patterns. Their abstract nature in conjunction with the complexity of OOP languages syntax makes it difficult for students to achieve a solid understanding of their benefits and use cases.

- Switching between programming techniques (Xinogalos, 2016). An example is the switching from procedural to object-oriented programming during the first years of a CS program. The OOP features are often hard to understand for novices. Concepts like inheritance, encapsulation and polymorphism demand a lot of effort from the students to gain an understanding of them. This confuses students who must change their programming mindset, which has huge disparity in some other programming techniques.

Therefore, the teaching of such concepts requires a lot of effort and time from the instructors (***problem-1: difficulties in achieving learning outcomes***). Another crucial factor that affects the comprehension of the conceptual design process is the insufficient or inadequate learning materials and teaching methods (***problem-2: lack of teaching / learning technologies***). In current state-of-practice, materials that are used to teach programming features (like books and static visualizations) are not enough for learning programming, due to its dynamic nature (Cheah, 2020). Consequently, it is hard and demands a lot of effort from the instructor to find or create educational material that fits a programming course. Also, it is difficult to cover the knowledge level of each student, by providing him/her with personalized educational material. Moreover, educational material can also contain a lot of problems or code smells as Fehnker and de Man (2019) concluded in their research. Some abstract concepts like data structures, algorithms and OOP demand pictures, visualizations or slides that help the students understand the way they work (Yang et al. 2018; Moraes and Taxeira, 2019). These materials are also important for motivating students to learn such abstract concepts (Yi Ding et al. 2014). The complexity that algorithms usually have makes the creation of such materials significantly complicated (Moraes and Teixeira, 2019).

In recent years, a promising solution to alleviate the aforementioned problems is the use of educational tools in OOP courses. Various types of educational programming environments have been proposed, including: programming microworlds (Maliarakis et al., 2012; Xinogalos and Satratzemi, 2004), flowchart-based programming environments (Xinogalos, 2013), serious games (Abbasi et al., 2017; Maliarakis et al., 2012), computer-supported collaborative tools (Silva et al., 2020), and distributed pair programming tools (Satratzemi et al., 2023). Such tools are expected to enhance the experience of the students with the course, enable them to perform additional self-studying, and support them in dealing with their difficulties and misconceptions. In this study, our focus is on educational programming environments that have as an additional goal to help students experiment with the design process, which is a trial-and-error process, rather than a strictly engineering one. In that sense, such tools provide the students the opportunity to make multiple conceptual design attempts, until they reach the final solution of optimal quality. The emergence of such a trend has led to the development of various educational tools, which subsequently has led to a need to synthesize existing literature in a systematic manner so as to better understand the domain. To this end, in this paper, we perform a systematic mapping study (SMS), aiming to investigate: (a) the learning objectives of educational tools (e.g., generic problem solving, application of the programming paradigm, mastering the language)—related to ***problem-1***; (b) the learning and teaching technologies used to support the development of the educational tools–related to ***problem-2***; and (c) evaluate the level of empirical evidence of the proposed education tools; to explore their usefulness in practice. Based on this high-level goal, we have set the following research questions (RQs):

RQ$_1$: What are the intended learning outcomes of educational tools for OO programming?

RQ$_2$: What is the empirical evidence on the effectiveness of educational tools for OO programming?

RQ3: Which teaching / learning technologies are used by the educational tools for OO programming?

The results of the study provide several implications that are expected to be of interest both for educators and researchers. For instance, tools that are targeted to different learning outcomes (RQ1) and consequently fall in different classes of the proposed categorization are fitting for different courses: usually tools targeted to generic problem solving issues would be fitting for introductory programming courses; tools targeted to the application of a programming technique would be fitting for intermediate programming courses or entry level analysis and design courses; and tools targeted to mastering the language would be fitting for advanced programming, e.g., software engineering courses.

The rest of the article is organized as follows. In the next section related work is presented and the differences with our systematic mapping study are highlighted, followed by an analysis of the review method. It continues with a presentation and a discussion of the results of the study. The last sections present threats to validity and the final conclusions.

Related Work

Xinogalos and Satratzemi (2004) in a review of teaching approaches and educational tools for introducing novices to programming identified six types of educational programming environments, namely programming microworlds, environments based on compilers with improved diagnostic capabilities, syntax editors (structure editors, iconic programming languages), program animators, systems that use algorithm and program animation, and program auralization tools. The authors conclude that programming microworlds support novices in dealing with most of the general difficulties that accompany the introduction to programming. Moreover, educational programming environments should incorporate a structure editor for avoiding focusing on the syntax, informative error messages that use physical language, as well as program animation and explanatory visualization for comprehending the semantics of the programming language and debugging.

Xinogalos (2013) reviewed educational programming environments based on the technology of structure editing and more specifically flowchart-based programming environments. Eleven environments were recorded in the literature with all of them supporting the imperative-procedural programming technique and just two the object-oriented. The majority of the environments support automatic source code generation in various programming languages or some sort of pseudocode. Novel aspects recorded are the support for collaborative activities, integration of a tutoring system, usage in mobile devices, and design as a web-based application and integration in a Learning Management System.

Malliarakis et al. (2012) performed a review of educational programming environments, programming microworlds and serious games for learning OOP. In their review the authors present representative examples of tools falling in the aforementioned categories, such as BlueJ, Alice, and Robocode, and propose a list of features that any tool used for learning OOP should fulfill. The features proposed are the following: using a physical/familiar metaphor, the GUI is object-oriented, visualization of concepts, object support, class support, interaction/experimentation, the editor supports program development, the compiler allows interaction, highly informative error messages, user friendly debugger, and simplicity. The authors conclude that educational programming environments and microworlds are valuable tools with rich features, but fall short in motivating students to be more active learners. Serious games on the other hand provide motivation in carrying out the tasks they include and are highly interactive, but do not cover all the OOP concepts.

Abbasi et al. (2017) carried out a systematic literature review of ways that serious games are used for learning OOP and teaching approaches applied in this context. The systematic literature review includ-

ed 15 studies published from 2015 to 2016. The results suggest that learning OOP can be accomplished through playing games, creating games, or utilizing game related tools, while playing games is the most common and effective approach followed by the utilization of game related tools. The teaching approaches recorded include objects first, concepts first, GUI first and code first, with the most common one being game first.

Souza et al. (2016) systematically reviewed 49 studies with the aim of investigating what assessment tools have been developed for programming assignments and what their main characteristics are. The tools reviewed were classified by assessment type, approach and specialty. Most of the tools aim at supporting instructors by automating the assessment of assignments or the students by providing them with immediate feedback to improve the quality of their code.

Silva et al. (2020) in their systematic literature review studied computer-supported collaborative learning in programming education. Twenty-seven studies published since 2015 were included in this review to study what collaborative resources are used, which resources are most effective, what has been measured, how collaboration is structured and measured. The resources were classified in nine distinct categories, with two of them referring to programming-oriented resources, including collaborative programming editors and support for pair programming (PP), as well as motivational resources including gamification.

Satratzemi et al. (2023) in a systematic literature review of 57 studies on distributed pair programming (DPP) in higher education investigated, among other issues, DPP tools and their assessment. In 54 out of the 57 studies included in the review, the tools used for applying DPP were identified. More than forty tools were identified and classified in two main categories, namely screen sharing applications and collaborative work support tools. The former category includes video conferencing tools, remote desktop sharing systems and video conferencing tools with desktop sharing and remote desktop control features. The latter category includes synchronous source code editors, Eclipse plugins with DPP support and integrated development environments (IDEs) with DPP support. Eclipse plugins and IDEs with DPP support provide features like awareness-floor control, collaboration awareness and gesturing features that result in enhanced collaboration of students. Logging capabilities for recording students' actions along with the use of learning analytics are considered important both for promoting students' experience and achievements and advancing research in the field.

The studies briefly presented in this section review various types of educational programming environments that aim mainly at supporting novices in dealing with the difficulties faced during their introduction to programming. Programming microworlds, structure editors and flow-chart based programming environments help students concentrate on comprehending programming concepts rather than the syntax of the underlying programming language (Xinogalos and Satratzemi, 2004; Xinogalos 2013). Serious games, on the other hand, aim mainly at motivating students in highly interactive environments (Malliarakis et al., 2012; Abbasi et al., 2017). Some environments and/or teaching approaches for programming exploit the strengths of computer-supported collaborative learning for applying pair programming (Silva et al., 2020) or distributed pair programming (Satratzemi et al., 2023). Both pair programming and distributed pair programming can potentially lead to better quality code. Finally, automatic assessment tools for programming assignments can assist students in improving the quality of their code through immediate feedback (Souza et al., 2016). Although, all the aforementioned types of programming environments can ultimately lead to better quality code through a deeper comprehension of OOP concepts, our SMS aimed at reviewing educational programming environments that focus on supporting students in better object-oriented design. This requires a good comprehension of OOP concepts, as well as higher order thinking skills for abstracting from the problem specification and defining

the necessary classes that represent the domain of the problem; defining the relationships between the classes; as well as the properties and functions for modeling each entity from the problem domain through classes.

Review Method

In this section we present the protocol for designing our systematic mapping study. The study has been designed and reported based on the guidelines of Petersen et al. 2015.

Research Questions

RQ1: What are the intended learning outcomes of educational tools for OO programming?
This research question aims to provide an overview of the intended learning outcomes of the proposed tools. There is an effort to categorize the tools by identifying their common learning outcomes. This categorization gives insights about the main learning outcomes of the tools introduced in the studied papers, enabling the easier identification of gaps in the research state-of-the art, as well as educators in picking the most appropriate tool, based on their needs.

RQ2: What is the empirical evidence on the effectiveness of educational tools for OO programming?

By answering this research question, we investigate the methods used to evaluate the effectiveness of the proposed tools. We relate participants of the experiments to their evaluation context and then to their learning outcomes. Also, the evaluation context is investigated for each intended learning outcome. Additionally, the results of each evaluation are analyzed to determine whether the tools have achieved their goals. The answer to this research question can give information about the methodologies used to evaluate educational tools and assist researchers with further research on the evaluation methodologies or selecting the methodology to use with their case. Furthermore, the effects of the evaluated tools can be identified from the results of each study to help researchers and instructors with what results or issues to expect from using similar tools. Finally, educators are made aware of which tools have been tested and how, so that the selection of educational tools is as informed as possible.

RQ3: Which teaching / learning technologies are used by the educational tools for OO programming?
The answer to this research question will present the main technologies used by the discovered tools. We also investigate what technologies are mostly used for helping with specific learning outcomes, linking this with the results of the first research question. Researchers and educators can use the results of this RQ to find out which are the most used technologies that are currently used or assist them on selecting the most appropriate technologies to use, based on their educational purpose.

Searching and Filtering Strategy

The set strategy aims at identifying a wide range of articles that introduce or use educational programming tools for supporting students in problem-solving and software design, with a focus on code quality, using the OOP approach. The literature search was conducted using Scopus, applying the following search string:

“TITLE-ABS-KEY ((oop OR “object-oriented” OR “object oriented programming”) AND (education OR educational OR teaching OR learning) AND (tool OR environment)) AND (PUBYEAR > 2007)”.

The application of the search string returned a set of 1,417 papers. Given the large number of returned

papers, we consider our goal of having a broad search string as fulfilled, delegating the responsibility of proper paper selection to the next step of the SMS process.

The selection of the papers was performed, by assessing their title and abstract. Articles referring to the evaluation or the development of tools for supporting students in problem-solving and software design using the OOP approach and their integration in the educational process either directly or indirectly were selected. Such papers must also focus on the quality and design aspects of object-oriented programming. The inclusion and exclusion criteria of the papers are presented in Table 1, while the list of studies selected through the aforementioned methodology is presented in Table 2. Upon the application of IC/EC, we have retained 14 papers. After snowballing, 4 additional papers have been included in our analysis, leading to 18 studies in total. Information about the tools such as tool names mentioned in the studies and a short description can be found in tabular form at Appendix A.

Table 1. Inclusion and exclusion criteria

Inclusion criteria	Exclusion criteria
Studies from 2008 to 2020	Studies older than 2008
Developed or existing tools/solutions for helping students improve their code and software design quality using the OOP approach.	Tools/solutions supporting students in non-OOP approaches. Tools/solutions focusing mainly on supporting students in comprehending fundamental OOP concepts (such as programming microworlds, educational games for programming) and not on designing OO programs of a considerable complexity. Tools/solutions targeted to experienced and/or professional programmers without a clear application to educational settings. Research mentioned on MOOCs. Strategies or proposed methodologies not related with a proposed or existing tool.
Tools/solutions that were ideally used and evaluated by students in an educational setting, or at least were evaluated by instructors.	Papers describing solely a proposal/design of a tool.

Table 2. Studies by Phase

Query	# of studies	# of selected studies	# of tools	References (selected studies)
Search query	1417	14	12	Alonso and Py 2009; Alonso et al. 2008; Dominique et al. 2013; Silva and Dorça, 2019; Yang et al. 2015; Yang et al. 2018; Azimullah et al. 2020; Vallejos et al. 2018; Ardimento et al. 2020; Blau and Moss 2015; Herout and Brada 2015; Mirmotahari et al. 2019; Yan et al. 2020; Zaw et al. 2018
Snowballing		4	4	Hashiura et al. 2010; Fehnker and de Man, 2019; de Andrade Gomes et al. 2017; Dietrich and Kemp, 2008
Total		18	16	

Data Collection

Upon study selection we recorded various data points for each primary study. The recorded variables can be organized into 4 categories, as presented in Table 3. The first category contains demographic data, such as the title of the study, the authors, and the year of publication. The second category contains the purpose of the study and the contributing field. The third category aims at the tool or the solution that is introduced or described in each study. The final category refers to details about the evaluation of the tool or solution of each study. The evaluation should contain properties such as the evaluation method, the origin and the size of the sample that participated, and the conclusions about the effectiveness of the proposed tool or solution.

Table 3. Data Collection Overview

Category	Properties
Demographics	● Title ● Authors ● Year
Purpose/Contributing Field	● purpose of the research ● contributing field
Tool/Solution	● In the case of a tool proposal its architecture, used technologies, and its functionality were recorded. ● In the case of a solution based on existing tools their integration and parameterization were recorded.
Evaluation	● Evaluation method ● Sample ● Number of participants ● Conclusions

The final set of recorded variables are described in Table 4. The utilization of specific variables during this phase will help at answering the research questions, referring to each property using the variable symbol (V1, V2, etc.). Variables V1 to V3 are intended for demographic purposes, and variables V4 to V9 help in answering the research questions.

Table 4. Recorded Variables

Code	Name	Description
V1	Title	The title of the paper
V2	Author	The list of the authors of the paper
V3	Year	The publication year of the paper
V4	Learning Outcome Category	The intended learning outcome of the proposed tool (code quality, OOP concepts, OOP design)
V5	Evaluation	If an evaluation has been conducted or planned. The values are between "yes", "no", and "planned".
V6	Context of Evaluation	The context where the evaluation of the study was conducted. The values are among exams, assignments, and experiments.
V7	Evaluation Participants	The numeric value of the participants in the evaluation of the tool.
V8	Evaluation outcome	The main outcomes and conclusions of the evaluation. They are organized based on the effects of the tool at the learning process, the conclusions about the utilization of the tool, and any limitations of the tool or negative results that occurred.
V9	Learning Technology Categories	The learning technology categories are based on the IEEE Transactions on Learning Technologies (TLT) taxonomy

For [V4], the learning outcome category is defined by classification using keywording. Keywords and the terms used to define the learning outcome are present in the study abstract, keywords, and full text. The learning technology categories for [V9] is determined based on the description, the specifications, and the purpose of the tool that each study presents. Most of the tools fall into multiple categories. The learning technologies are based on the TLT Taxonomy[1] categories and subcategories. The evaluation method data leveraged for answering the corresponding research question consists of the information included in the variables [V5], [V6], and [V7]. Evaluation data are not complete for each study due to missing or incomplete data. Many studies did not mention all the details about their evaluation or had inaccurate values such as averages or value ranges. In inaccurate cases the average is used for the measurements.

[1] https://ieee-edusociety.org/about/tlt-taxonomy-page

Data Analysis

The collected variables are used to answer the research questions. Variables [V1] – [V3] are collected for documenting and identifying the papers. Variable [V5] is used for the selection of the papers as the only acceptable values are "yes" or "planned". Variables [V6] – [V9] are leveraged for answering the research questions. For RQ_1 we investigate the intended learning outcomes for each proposed tool. The learning outcome is measured using terms that describe the purpose of each research. These terms were mainly found in the abstract, keywords and in the title of the paper. Because of the variety of terms and keywords found, a method for processing this data was necessary. To merge the found terms and keywords into general ones the Open Card Sorting (Spencer, 2009) is leveraged. The following steps were followed: (a) keywords related to intended learning outcomes were collected from title, keywords, and abstract, for each study, (b) the discovered keywords were reviewed and candidates for merging were found, (c) the names of the final categories were formed. For RQ_2, the results of the evaluation variables collection are reported. Then the relation of the participants and the study evaluation context with the learning outcome comes from each study by performing cross-tabulation between the corresponding variables. For measuring the evaluation outcome [V8] of the educational tools, a similar methodology with the learning outcomes extraction was followed based on the Open Card Sorting (Spencer, 2009): (a) the effects were recorded form the abstract and the results of the papers; afterwards, (b) the effects were reviewed, and possible categories were identified; and finally, (c) the effect categories were formed. For RQ_3 we present the defined learning technology categories based on the TLT taxonomy, for each proposed tool. Furthermore, the learning technology categories and subcategories relation with the learning outcome category of each tool is investigated. The date analysis methods used each RQ are presented in Table 5.

Table 5. Research question data analysis techniques

Research Question	Used Variables	Analysis Method
RQ_1	[V4]	Frequency Tables
RQ_2	[V4], [V5], [V6], [V7], [V8]	Frequency Tables and Cross Tabulation
RQ_3	[V4], [V9]	Frequency Tables and Cross Tabulation

Results

In this section the results of the analysis are presented. Specifically, we present demographics data, and then we present the results for each research question. In Figure 1 the frequencies of the studies publication years are presented. The years are grouped in periods, due to the small number of selected studies. From the results of this grouping, it can be observed that during the latest period, 2017 to 2020, more tools are developed than the other periods, 2008 to 2016.

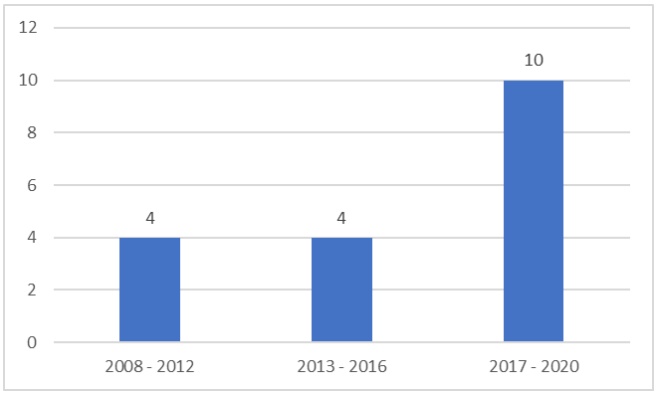

Figure 1. Paper Intensity Evolution

Learning Outcomes (RQ_1)

In this section, we present the findings related to the learning outcomes of the identified tools. The learning outcomes are classified into 3 different categories. For the identification of the categories, the Open Card Sorting process was applied (Spencer, 2009):

- **Code quality** refers to tools that aim on quality aspects of code related with the style and the design.
- **OOP Concepts**. This category includes some tools aimed at the understanding of specific object-oriented concepts such as inheritance, polymorphism, class and properties, encapsulation etc.
- **OOP Design**. This category contains tools that help with the understanding of design concepts of the object-oriented paradigm. In this category tools related to the teaching of design patterns, which are strictly related with the OOP design quality, are also included.

Following the usual reporting of SMSs, in Table 6, we present the frequencies of the studies in each learning outcomes' category. Based on Table 6, we can observe that most of the studies refer to the OOP design followed by the ones that refer to the code quality. Only 3 of the studies fall into the category that refers to OOP concepts. There is a clear trend in design and quality aspects and less on fundamental OOP concepts for improving the overall quality of code. This result was not surprising, since the studies reviewed aim at enhancing the conceptual design of OO programs. This means that higher order thinking, analysis, synthesis and design skills are required, while the comprehension of fundamental of OOP concepts is at some degree taken for granted.

Table 6. Mapping of Primary Studies to Learning Outcomes

Intended Learning Outcome	Studies	Studies
OOP Design	8	Ardimento et al. 2020; Alonso and Py 2009; Alonso et al. 2008; Dominique et al. 2013; Silva and Dorça, 2019; Fehnker and de Man, 2019; Azimullah et al. 2020; Dietrich and Kemp, 2008
Code Quality	7	Blau and Moss, 2015; Herout and Brada, 2015; Mirmotahari et al. 2019; Yan et al. 2020; Hashiura et al. 2010; de Andrade Gomes et al. 2017; Vallejos et al. 2018

Intended Learning Outcome	Studies	Studies
OOP Concepts	3	Zaw et al. 2018; Yang et al. 2015; Yang et al. 2018

Empirical Evidence (RQ_2)

In this section, we present the evaluation results conducted in the included studies. The existence or the planning of an evaluation of the proposed tool was also a condition to include a study to this research. From the selected studies one described an evaluation that is planned to take place (Yang et al. 2015) and the rest of the studies (Ardimento et al. 2020; Alonso and Py 2009; Alonso et al. 2008; Dominique et al. 2013; Silva and Dorça, 2019; Fehnker and de Man, 2019; Azimullah et al. 2020; Dietrich and Kemp, 2008; Blau and Moss, 2015; Herout and Brada, 2015; Mirmotahari et al. 2019; Yan et al. 2020; Hashiura et al. 2010; de Andrade Gomes et al. 2017; Vallejos et al. 2018; Zaw et al. 2018; Yang et al. 2018) presented a completed evaluation. From the evaluation data, the most accurate and complete were the study evaluation context [V6], the number of participants [V7], and the evaluation outcome [V8]. The context of evaluation refers to the context where the evaluation of each proposed tool took place. The most common contexts for evaluation were experiments, assignments, and exams. An experiment is a procedure that is dedicated to the evaluation of a tool and involves students in most of the analyzed studies. Assignments are used to test the tools in the context of course assignments that are either optional or mandatory (are part of the final grade). In just one case, the tool was used as part of the final exams of a university course. In Figure 2 the frequencies of each context is presented. Most of the studies (10 in total) had their evaluation conducted as a dedicated experiment for their proposed tool. In 4 studies, the tools were used to help students to complete their assignments for their evaluation. In just one case the tool is tested during the final exams of the CS course.

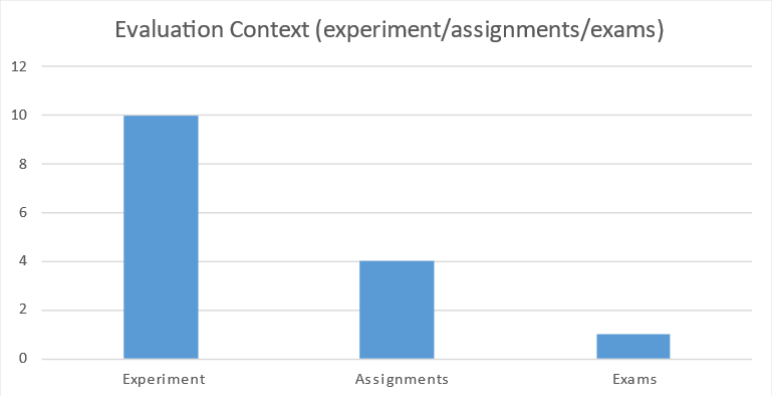

Figure 2. Evaluation Context

For each study, the number of participants was collected in case it was provided. It refers to the total number of people, in most cases students, that participated in the evaluation of the tools. There were many issues regarding the determination of the exact number of participants. In some studies, (Herout and Brada, 2015; Blau and Moss, 2015), the number of participants was an average or a value range. In these cases, the average or the mean value are considered as the numeric value of the participants. Also, in some cases (Alonso et al. 2008; Yang et al. 2018) there are multiple evaluations with different participants, where the total participants were considered. To investigate the relationship between the intended learning outcome category and the evaluation data collected from the studies, cross-tabulations are performed between variables [V4], [V6], and [V7]. The average evaluation participants

per intended learning outcome category are presented in Table 7. Tools related to code quality have the most participants in their evaluation, 164 participants, followed by the tools aiming on teaching specific OOP concepts, 86 participants. The fewer participants are observed in the evaluation of tools related to OOP design.

Table 7. Participants per intended learning outcome (RQ_1)

Learning Outcome (RQ1)	AVG	Min	Max
Code Quality	163.5	15	528
OOP Concepts	86	10	162
OOP Design	20.5	12	38

In Table 8, we present the average number of participants per study evaluation context [V6]. The most participants are observed during final exams, where most of the students already participate to pass the course. The assignments are on average conducted with 83 participants, and finally experiment with on average approximately 42 students. That indicates the importance of exams and assignments to the students by rewarding them with extra credit, where in the experiments it is more difficult to find participants if this does not affect the final grade.

Table 8. Average Participants per study evaluation context

Study Evaluation context	#Participants	Min Participants	Max Participants
Exams	528	528	528
Assignments	83	10	300
Experiment	41.625	12	162

In Figure 3, we present the results of the cross-tabulation between the study evaluation context [V6] and the intended learning outcomes [V4]. We observed that experiments were used more for tools aiming on learning OOP design, whereas assignments and exams were leveraged for evaluating tools aiming on the code quality. For all the intended learning outcome categories, the dominating evaluation context is the experiment.

Finally, the outcomes of the evaluation of each tool give information and feedback about its effectiveness after it has been used. The results give insights about the effects of the tools in the learning process of the students, conclusions and tool use cases, and limitations of the tools. We recorded information about the evaluation outcomes in variable [V8] using the Open Card Sorting methodology (Spencer et al., 2009). (a) We analyzed the results and the conclusions of the evaluation and found the main points for each study. (b) Then reviewed the main points extracted from the studies to find common results and conclusions, to merge them into more general ones. (c) Finally, we defined the main evaluation outcome categories for the given studies. Each study had one or more conclusions and therefore they can be included in one or more of the evaluation outcome categories.

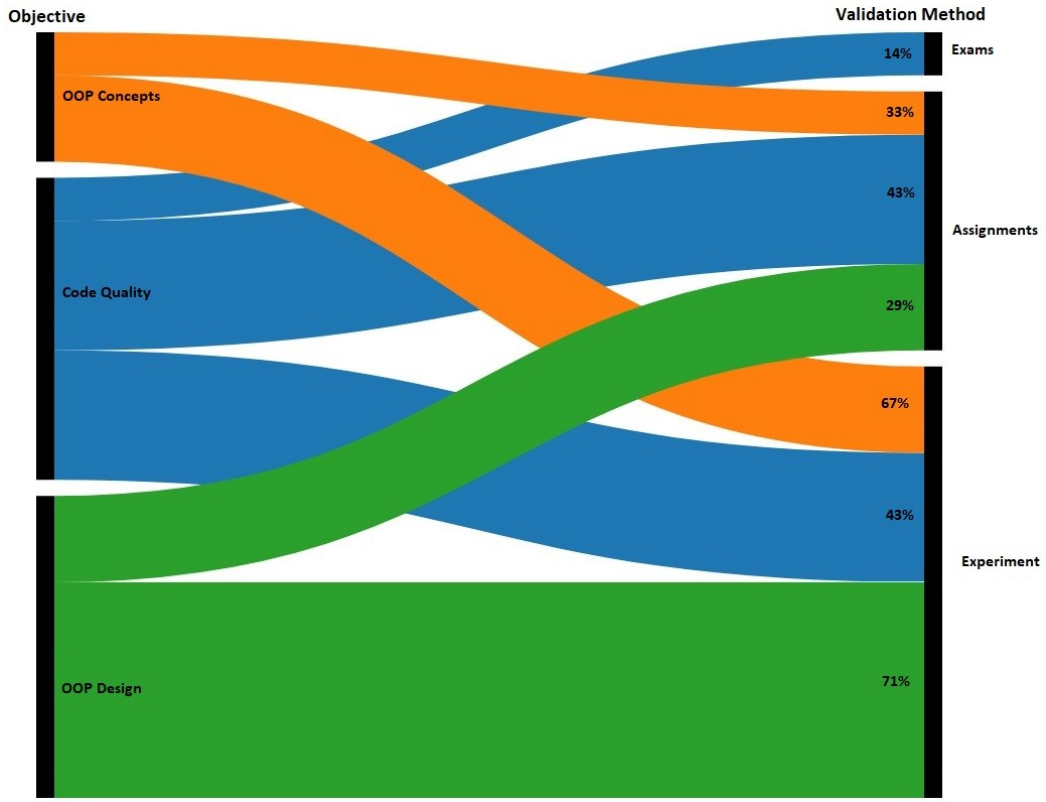

Figure 3. Alluvial diagram of study evaluation context of intended learning categories

Table 9 presents the results of the evaluations related to the effects of the tools in the learning process. Some of the tools had direct effects which were reflected in the students' performance. It was observed that the usage of some tools helped students correct mistakes in their code (Alonso and Py 2009; Alonso et al. 2008; Dominique et al. 2013) and improve it and its quality (Ardimento et al. 2020; Blau and Moss 2015; Zaw et al. 2018). In the case of Azimullah et al. (2020), they concluded that the students understood and learned how to use design patterns by using their proposed tool. In other cases, the tools had indirect effects at the learning process. In studies (Yan et al. 2020; Zaw et al. 2018; de Andrade Gomes et al. 2017) the tools helped the learners understand what code quality is. Yang et al. (2018) came to the conclusion that the students understood object-oriented programming concepts and had better understanding of the program execution, by using their proposed tool. Based on the conclusions of Yan et al. (2020), the students had their programming skill improved with the assistance of their tool. There is also a case (Ardimento et al. 2020) where students that used the proposed tool scored higher grades.

Table 9. Frequencies of the effects of the tools

Effects	#Studies	Studies
Correct mistakes	3	Alonso and Py 2009; Alonso et al. 2008; Dominique et al. 2013
Understand code quality	3	Yan et al. 2020; Zaw et al. 2018; de Andrade Gomes et al. 2017
Improve code quality	3	Ardimento et al. 2020; Blau and Moss 2015; Zaw et al. 2018
Higher grade	1	Ardimento et al. 2020
Improve programming skill	1	Yan et al. 2020
Better understanding of program execution	1	Yang et al. 2018
Understand and use design patterns	1	Azimullah et al. 2020
Understand OOP concepts	1	Yang et al. 2018

In Table 10, the results of the evaluations related to the conclusions about the tools and their utilization can be observed. Students in the study by Yang et al. (2018) were satisfied with the tool and in studies (Yang et al. 2018; Azimullah et al. 2020) were comfortable with using the tools. Students who were using the tool by Herout and Brada (2015) found out its usefulness while they were using it. In some cases, the proposed tool helped on the discovery of OOP errors while the instructors were using it (Vallejos et al. 2018), and on checking of the quality of learning materials (Fehnker and de Man, 2019).

Table 10. Frequencies of conclusions and tool use cases

Tool Conclusions and Utilization	#Studies	Papers
Comfortable with using the tool	2	Yang et al. 2018; Azimullah et al. 2020
Found the tool useful	1	Herout and Brada 2015
Satisfied with the tool	1	Yang et al. 2018
Help on OOP errors discovery	1	Vallejos et al. 2018
Help on check the quality of learning materials	1	Fehnker and de Man, 2019

Except for the positive results about the effectiveness of the tools and the observations of the tools, there were also limitations discovered. These results are presented in Table 11. The main issue of the tools is about the effectiveness of the automatic validation applied to students' solutions. In two studies (Alonso and Py 2009; Alonso et al. 2008) the tool utilized did not take into account the correctness of alternative solutions of the students. Vallejos et al. (2018) came to the conclusion that it is impossible for the proposed tool to detect some specific errors. All these outcomes come to an agreement with Herout and Brada (2015) who mention that fully automatic validation is not a solution to all the problems. Mirmotahari et al. (2019) on the other hand, conclude that automated feedback is helpful, but the time required is the same as before due to the time required to prepare the required criteria and review instructions. Also, in the study by Hashiura et al. (2010) it is concluded that just reviews are not enough for the improvement of code quality.

Table 11. Frequencies of tool limitations

Tool limitations	Papers
Automated feedback helps, but the time required is almost the same as before	Mirmotahari et al. 2019
Students can find also alternative solutions unknown to the tool	Alonso and Py 2009
The tool does not take into account the correctness of the students' solution diagram.	Alonso et al. 2008
Some errors are impossible to be found by the tool	Vallejos et al. 2018
Reviews are not enough for code quality improvement	Hashiura et al. 2010

Teaching / Learning Technologies (RQ_3)

In this section, we present the classification of the studies based on their teaching / learning technologies. For the analysis, the learning technology categories variable [V9] is mainly used to indicate the technologies utilized in the proposed tools of the studies. The results of the identification of the learning technologies are combined with the intended learning outcome to investigate potential relations between them. The frequencies of learning technology categories are presented in the Table 12. It is important to notice that a tool of a study can be associated with one or more learning technology categories. As can be observed, most of the studies are classified as part of the subcategory "2.8 E-Learning Tools, Self-Assessment Technologies" (12 papers) and "4.1 Adaptive and Intelligent Educational Systems, Intelligent Tutoring Systems" (12 papers). These categories are followed by "1.4 Learning environments, Virtual Labs" (9 papers) and "2.7 E-Learning Tools, Automatic Assessment Tools" (8 papers). Based on these leading categories, a trend on assessment technologies and smart virtual learning environments comes to surface.

Table 12. Mapping of Learning Technology

Learning Technology	#Studies	Papers
2.8 E-Learning Tools, Self-Assessment Technologies	12	Ardimento et al. 2020; Blau and Moss 2015; Herout and Brada 2015; Mirmotahari et al. 2019; Yan et al. 2020; Zaw et al. 2018; Dominique et al. 2013; Hashiura et al. 2010; Silva and Dorça, 2019; de Andrade Gomes et al. 2017; Azimullah et al. 2020; Dietrich and Kemp, 2008
4.1 Adaptive and Intelligent Educational Systems, Intelligent Tutoring Systems	12	Ardimento et al. 2020; Alonso and Py 2009; Alonso et al. 2008; Dominique et al. 2013; Hashiura et al. 2010; Silva and Dorça, 2019; Fehnker and de Man, 2019; de Andrade Gomes et al. 2017; Yang et al. 2015; Yang et al. 2018; Azimullah et al. 2020; Dietrich and Kemp, 2008
1.4 Learning Environments, Virtual Labs	9	Ardimento et al. 2020; Yan et al. 2020; Alonso and Py 2009; Alonso et al. 2008; Dominique et al. 2013; de Andrade Gomes et al. 2017; Yang et al. 2015; Yang et al. 2018; Dietrich and Kemp, 2008

Learning Technology	#Studies	Papers
2.7 E-Learning Tools, Automatic Assessment Tools	8	Ardimento et al. 2020; Blau and Moss 2015; Herout and Brada 2015; Mirmotahari et al. 2019; Yan et al. 2020; Hashiura et al. 2010; Fehnker and de Man, 2019; Vallejos et al. 2018
5.4 Standards and Interoperability, Web Services	7	Ardimento et al. 2020; Yan et al. 2020; Zaw et al. 2018; Hashiura et al. 2010; de Andrade Gomes et al. 2017; Yang et al. 2015; Yang et al. 2018
5.5 Standards and Interoperability, Authoring Tools	7	Alonso and Py 2009; Alonso et al. 2008; Dominique et al. 2013; de Andrade Gomes et al. 2017; Yang et al. 2015; Yang et al. 2018; Azimullah et al. 2020
2.6 E-Learning Tools, Homework Support Systems	6	Ardimento et al. 2020; Blau and Moss 2015; Yan et al. 2020; Alonso and Py 2009; Alonso et al. 2008; Dominique et al. 2013
1.2 Learning Environments, Learning via Discovery	4	Alonso and Py 2009; Alonso et al. 2008; Dominique et al. 2013; de Andrade Gomes et al. 2017
1.5 Learning Environments, Educational Simulations	3	Yang et al. 2015; Yang et al. 2018; Azimullah et al. 2020
4.3 Adaptive and Intelligent Educational Systems, Personalized E-Learning	2	Yan et al. 2020; Hashiura et al. 2010
2.1 E-Learning Tools, Web Lectures and Notes	1	Dietrich and Kemp, 2008
2.5 E-Learning Tools, Instructor Interfaces	1	Hashiura et al. 2010
4.2 Adaptive and Intelligent Educational Systems, Adaptive Hypermedia	1	Fehnker and de Man, 2019
5.3 Standards and Interoperability, Ontologies	1	Fehnker and de Man, 2019

In Table 13 we focus on the relation between the learning technology categories [V9] and the intended learning outcomes [V4] of the studied tools. The data is presented as learning technology category frequencies per intended learning outcome. In Figure 4 there is also a visual representation of the relation of [V9] and [V4] through a treemap. As can be observed, the leading technology category differs for each learning outcome. In the first category of intended learning outcomes, namely “code quality”, the most popular TLT utilized in the corresponding tools are “2.7 E-Learning Tools, Automatic Assessment Tools” and “2.8 E-Learning Tools, Self-Assessment Technologies”. This observation reveals a

trend of using e-learning and assessment technologies for tools proposed to help students directly with their code quality. For the "OOP Concepts" learning outcomes category, the leading TLT category is "5.4 Standards and Interoperability, Web Services", followed by "1.4 Learning Environments, Virtual Labs", "1.5 Learning Environments, Educational Simulations", "4.1 Adaptive and Intelligent Educational Systems, Intelligent Tutoring Systems", and "5.5 Standards and Interoperability, Authoring Tools". Based on these categories we can assume that learning environments, educational tools and web services are mainly used to improve object-oriented concepts teaching and learning process. For the last category of intended learning outcomes, namely "OOP Design", the main learning technology categories are "4.1 Adaptive and Intelligent Educational Systems, Intelligent Tutoring Systems", "1.4 Learning Environments, Virtual Labs", "2.8 E-Learning Tools, Self-Assessment Technologies", "2.6 E-Learning Tools, Homework Support Systems", and "5.5 Standards and Interoperability, Authoring Tools". In the case of object-oriented design, educational environments and labs that support students with tutoring, self-assessment, and homework are most used among the studied tools.

Table 13. Crosstabulation of TLTs and Learning Outcome

Learning Outcomes	TLT Category	# papers	Papers
1. Code Quality	2.7 E-Learning Tools, Automatic Assessment Tools	6	Blau and Moss 2015; Herout and Brada 2015; Mirmotahari et al. 2019; Yan et al. 2020; Hashiura et al. 2010; Vallejos et al. 2018
	2.8 E-Learning Tools, Self-Assessment Technologies	6	Blau and Moss 2015; Herout and Brada 2015; Mirmotahari et al. 2019; Yan et al. 2020; Hashiura et al. 2010; de Andrade Gomes et al. 2017
	5.4 Standards and Interoperability, Web Services	3	Yan et al. 2020; Hashiura et al. 2010; de Andrade Gomes et al. 2017
	1.4 Learning Environments, Virtual Labs	2	Yan et al. 2020; de Andrade Gomes et al. 2017
	2.6 E-Learning Tools, Homework Support Systems	2	Blau and Moss 2015; Yan et al. 2020
	4.1 Adaptive and Intelligent Educational Systems, Intelligent Tutoring Systems	2	Hashiura et al. 2010; de Andrade Gomes et al. 2017
	4.3 Adaptive and Intelligent Educational Systems, Personalized E-Learning	2	Yan et al. 2020; Hashiura et al. 2010
	1.2 Learning Environments,	1	de Andrade Gomes et al. 2017

Learning Outcomes	TLT Category	# papers	Papers
	Learning via Discovery		
	2.5 E-Learning Tools, Instructor Interfaces	1	Hashiura et al. 2010
	5.5 Standards and Interoperability, Authoring Tools	1	de Andrade Gomes et al. 2017
2. OOP Concepts	5.4 Standards and Interoperability, Web Services	3	Zaw et al. 2018; Yang et al. 2015; Yang et al. 2018
	1.4 Learning Environments, Virtual Labs	2	Yang et al. 2015; Yang et al. 2018
	1.5 Learning Environments, Educational Simulations	2	Yang et al. 2015; Yang et al. 2018
	4.1 Adaptive and Intelligent Educational Systems, Intelligent Tutoring Systems	2	Yang et al. 2015; Yang et al. 2018
	5.5 Standards and Interoperability, Authoring Tools	2	Yang et al. 2015; Yang et al. 2018
	2.8 E-Learning Tools, Self-Assessment Technologies	1	Zaw et al. 2018
3. OOP Design	4.1 Adaptive and Intelligent Educational Systems, Intelligent Tutoring Systems	8	Ardimento et al. 2020; Alonso and Py 2009; Alonso et al. 2008; Dominique et al. 2013; Silva and Dorça, 2019; Fehnker and de Man, 2019; Azimullah et al. 2020; Dietrich and Kemp, 2008
	1.4 Learning Environments, Virtual Labs	5	Ardimento et al. 2020; Alonso and Py 2009; Alonso et al. 2008; Dominique et al. 2013; Dietrich and Kemp, 2008
	2.8 E-Learning Tools, Self-Assessment Technologies	5	Ardimento et al. 2020; Dominique et al. 2013; Silva and Dorça, 2019; Azimullah et al. 2020; Dietrich and Kemp, 2008
	2.6 E-Learning Tools, Homework Support Systems	4	Ardimento et al. 2020; Alonso and Py 2009; Alonso et al. 2008; Dominique et al. 2013

Learning Outcomes	TLT Category	# papers	Papers
	5.5 Standards and Interoperability, Authoring Tools	4	Alonso and Py 2009; Alonso et al. 2008; Dominique et al. 2013; Azimullah et al. 2020
	1.2 Learning Environments, Learning via Discovery	3	Alonso and Py 2009; Alonso et al. 2008; Dominique et al. 2013
	2.7 E-Learning Tools, Automatic Assessment Tools	2	Ardimento et al. 2020; Fehnker and de Man, 2019
	1.5 Learning Environments, Educational Simulations	1	Azimullah et al. 2020
	2.1 E-Learning Tools, Web Lectures and Notes	1	Dietrich and Kemp, 2008
	4.2 Adaptive and Intelligent Educational Systems, Adaptive Hypermedia	1	Fehnker and de Man, 2019
	5.3 Standards and Interoperability, Ontologies	1	Fehnker and de Man, 2019
	5.4 Standards and Interoperability, Web Services	1	Ardimento et al. 2020

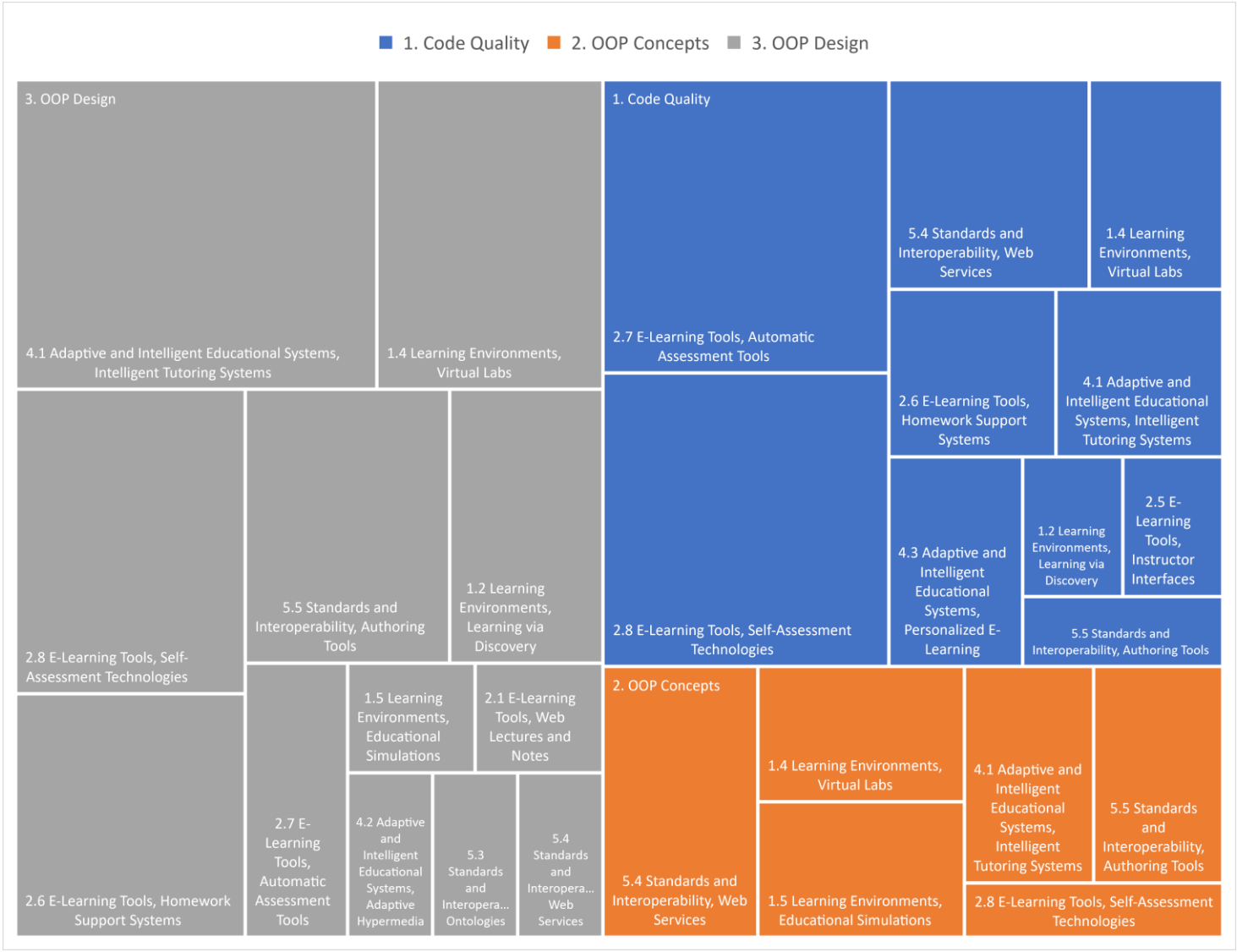

Figure 4. Treemap of top TLTs per learning outcome category

Discussion

In this section we discuss the results of the research presented in this paper. They are also compared to the findings of some of the related work studies. The results of this research align with the conclusions of Souza et al. (2016) regarding the variety of the tools for automatic assessment and the immediate feedback to help students to improve their code quality. Also, the design as web applications and the usage of web technologies by educational tools is concluded by Xinogalos (2013).

During the examination of the papers and their year of publication there were observations about the research trend per year. In recent years there is an increasing interest on programming teaching tools, and more specifically OOP teaching tools. This fact could reveal an increasing necessity for efficient teaching of OOP concepts and design. This could be a result of the increasing demand for IT knowledge in the industry.

Most of the analyzed tools aim at OOP design and quality aspects rather than specific OOP concepts. A possible explanation of this distinction could be the difficulty and complexity behind the understanding

and implementation of an efficient and well-structured Object-Oriented domain model. Design and quality aspects of OOP are also highly related. Design affects the overall quality of the software to a higher level.

A very important aspect of an educational tool is its effectiveness in the educational process. One of the main goals of this research is the review of the evaluation process of the discovered tools. Regarding the evaluation context of the evaluations, the most participants were recorded during exams. This happens because it is mandatory for students to pass the exams for each course. Assignments are a good option for testing a tool as they are tested in near real teaching conditions. Students are also obliged to complete them as they are part of a course, and they usually affect its final grade. A downside of using assignments for the evaluation is the possibility of students trying to cheat or receive help in completing them, leading to less significant results. Most of the tools were tested during a dedicated experiment. In this case, the participants were fewer than the cases where exams or assignments were used. The context and the methodology can be strictly defined and adapted to the research needs. The appropriate participants sample can be explicitly defined and selected. Participating in an experiment is not usually mandatory, which could cause problems in finding willing participants. When taking part in an experiment there is also the danger of participant bias, since the participants have in mind that they are part of the experiment and that could also affect the evaluation results.

Some of the tools evaluation results refer to the students' perspective. Usefulness and usability were very important aspects during the evaluation of the educational tools. Researchers' ought to think about them during the implementation and evaluation of their educational tools. Moreover, a lot of limitations of the tools were noticed, which are mainly referred to inability of the tools evaluating alternative solutions provided by the students. Creativity and freedom of the students are limited in these cases. In the other hand, there are cases where mistakes made by students are impossible to be discovered by tool, making them effective only on certain cases.

A very noticeable insight into the developed educational tools are the technologies used by them and their architecture design. Tools are developed mainly as web applications, desktop applications, and plugins for existing tools or software. There is a big focus on automatic assessment and tutoring students in problem-solving during classes or homework. The usage of dynamic visualizations and simulations is also useful when static content in not sufficient. Tools aiming on improving code quality, have as their main goal to help students understand and correct their quality-related mistakes. They are mainly developed as web services or virtual lab environments. For teaching specific OOP concepts, the developed tools are educational simulations in most cases. They are designed as web services or virtual lab environments. Simulations can assist students in understanding complex object-oriented concepts in a user-friendly manner. Object-oriented design teaching is approached by developing tutoring systems and virtual labs. These tools can help students by tutoring them in implementation of OO design during assignments solving.

Threats to Validity

In this section threats to validity are presented, following the guidelines of Ampatzoglou et al. (2019). More specifically, the study selection validity, the data validity and the research validity are described in the aforementioned order.

Study Selection Validity

Study selection validity refers to the early stages of the research, the searching of the studies and their filtering. The identification of the studies is based on automated search in Scopus. A broad search string is used, which included keywords and synonyms related to object-oriented programming learning and teaching tools or environments. The inclusion of studies published after 2007 is also part of the search string. Studies that used different terminology than the one used in this research might have been excluded unintentionally. Additionally, some articles have been discovered through snowballing. Studies and articles published in grey literature are excluded, since we focus on empirical evidence, which are almost never published in gray literature. Our study also is not suffering from missing non-English papers and papers published in a limited number of journals, since a large number of venues is used for the search of the papers. Finally, we had access to every DL we are interested in, as our institution provided access to them.

Data Validity

The data validity is mainly affected by data extraction bias. The data is extracted and recorded by the first author. To mitigate the possibility of subjectivity during this process, the other 2 authors reviewed the extracted data and re-validated them. Afterwards, all authors discussed together the results of the extraction and resolved any conflicts that took place.

Another possible issue of the data validity is the publication bias. It can be either (a) bias caused because of a closed and small circle the primary studies are published at or (b) the tendency of publishing positive results and not negative ones (Ampatzoglou et al. 2019). The first type of publication bias is not present in this study because the broad search string was applied in Scopus and the studies originated from a large group of researchers. Regarding the second type of publication bias, there are some cases where it can be identified. There is a tendency to emphasize more the positive effects of the educational tools in the teaching and learning procedure, where negative or neutral effects are omitted or barely mentioned. Some limitations of the evaluated tools mentioned were also recorded and included in the study results.

There are additional threats that can affect the data validity of this study. The sample size of the research is not very large. After the recording, 18 studies were finally selected from the initial 1417 discovered by the search string. Lack of relationships is not a possible threat for this study, because it is not intended to find any relationship among the recorded data, but only to classify the data. Low quality of primary studies is a potential threat because, based on the SMS guidelines (Petersen et al. 2008), no type of quality assessment is performed, as there was no explicit quality related research question. The selection of the variable to be extracted was not a threat because they were discussed, and any conflicts existed at the beginning of the research were resolved before the data was extracted and recorded. There were no issues regarding the statistical analysis, as there was no hypothesis testing. Only basic statistical analysis and cross tabulations were calculated. Finally, the researchers bias in data interpretation and analysis was mitigated by discussing the clustering for the intended learning outcomes of the studies and the evaluation context. Some of the results explanations are based on the viewpoints and personal opinions and experiences of the authors, as they understand them.

Research Validity

The main threats related to the research validity are the research method bias and repeatability. Regarding the first threat, the majority of the authors of this study are very familiar with the process of conducting secondary studies, as they have participated in a very large number of secondary studies as reviewers and coauthors. Therefore, the threat concerning the research method bias is minimal. The

second threat, repeatability, can be ensured as replication and reliability are enabled due to the detailed review process followed in this study. The review procedures and all decisions made in specific cases are recorded and described in this manuscript. Multiple authors were involved in every phase of this research to reduce any potential bias. Finally, all the data extracted is publicly available to allow the validation and the comparison of the research results.

Three research questions were defined through discussion between the 3 authors. These research questions are accurately and holistically mapped to the goal of the study, as it was described in the introduction. Therefore, there was no research question selection bias. Furthermore, the research method selected for this study is adequate for the goal and no deviations from the guidelines were made.

Conclusion

Teaching object-oriented programming is a very important and demanding procedure. A lot of effort is required to efficiently teach the related concepts and the conceptual design. Two very crucial issues that affect the teaching process are the complexity of specific programming concepts and the lack of sufficient learning tools and materials. This mapping study aims on providing insights about OOP educational tools that assist students with understanding and learning the conceptual design. More specifically, we investigate (a) the main learning outcomes of educational tools, (b) the empirical evidence on the effectiveness of these tools, and (c) the main technologies used. For this research, 18 total articles were discovered and analyzed in order to give answers to the research questions.

The results of this research showed that there is an increasing interest in OOP teaching tools. The last years there was a significant increase in the number of studies introducing OOP educational tools. There is a lot of attention paid also in the program design and quality regarding the goals and the learning outcomes of the tools. Regarding the evaluation of the tools, course assignments and experiments are widely used. A combination of them, experiments during course assignments, could be proved more useful and effective. There is a lot of focus at the usefulness and the usability of the tools during the evaluation process. Additionally, several defects of the tools were pointed out, mainly referring to limited solutions for a problem and the inability to validate alternative ones. The review of the discovered papers showed that tools are mainly developed as web applications or software plugins. Their main goal includes automatic assessment or tutoring functionalities during assignments solving.

Appendix A

Tools Info Table

Study	Tool Name	Tool Info
(Ardimento et al., 2020)	Student Profiling Tool (SPT)	A cloud-based tool based on Eclipse Che Platform. It helps students monitor common Object-oriented paradigm violations by providing feedback as reports about their mistakes, in order to improve their OOP knowledge and skills.
(Blau and Moss, 2015)	FrenchPress	An eclipse plugin that provides students feedback about their Java programs. It delivers explanatory massages to the students about common novice Object-Oriented idiom mistakes, in vocabulary appropriate to their current knowledge level.
(Herout and Brada, 2015)	Undefined	The proposed tool uses "duck tests" to validate the students' assignments implementation quality. The tool is used by the students to help them verify their assignment solutions.
(Mirmotahari et al., 2019)	Undefined	A digital assessment tool, which generates and provides qualitative formative feedback to students in order to help them improve the quality of their programs and assist them in the learning process.
(Yan et al., 2020)	ProgEdu	ProgEdu is an automated programming assessment system (APAS) that validates the quality of the code submitted by students, especially if the code follows Java programming language conventions. Students can interact with the environment by submitting their assignments multiple times and receive immediate feedback. It also provides insights to the instructors about the students' learning performance.
(Zaw et al., 2018)	JPLAS	A web-based learning assistant system that aims to help students understand OOP concepts by implementing source code given the necessary information. The tool evaluates the code following the informative test code approach. This test code provides the necessary information about the expected contents of the code such as names of classes, methods or properties, access modifiers, data types and arguments.
(Alonso and Py, 2009; Alonso et al., 2008; Dominique et al., 2013)	Diagram	Diagram is an interactive learning environment, developed as a UML editor application, for object-oriented modeling. It gives the ability to create the UML diagram for a given problem and match terms of the problem description text with the elements of the UML diagram.
(Hashiura et al., 2010)	Undefined	The proposed tool analyzes students' exercises code quality among a rule set and provides feedback to them. In order to achieve it, SonarQube is used with specific rules applied to analyze the code of the students.
(Silva and Dorça, 2019)	Undefined	An Eclipse plugin is proposed which helps students and teachers recognize software design problems and learn object-oriented programming.
(Fehnker and de Man, 2019)	Undefined	This study introduces a tool that provides feedback about code smells in PROCESSING, a language based in Java. Its purpose is to support teaching by providing insights about program design quality to novice students.

(de Andrade Gomes et al., 2017)	SMaRT	SMaRT is an Eclipse plugin that provides insights about the quality of the students' code. It leverages SonarQube to gather metrics about the quality of the code and present them in a friendly report view.
(Yang et al., 2015)	JavlinaCode	JavlinaCode is a web-based interactive educational programming environment which is designed to help teach OOP in Java.
(Yang et al., 2018)	JaguarCode	The introduced tool is a web-based programming environment which helps students understand the static structure and dynamic behavior of Java programs. It provides UML diagrams and dynamic execution trace features.
(Azimullah et al., 2020)	Bounce project	Bounce Project is a tool that uses a combination of real-world metaphors and programming coding exercises to teach design patterns step by step. Students receive real time feedback about their exercise completion status.
(Dietrich and Kemp, 2008)	DPLab	DPLab is an Eclipse IDE plugin that assists students to learn design patterns. The plugin provides guides and exercises about several design patterns which can be accessed through Eclipse.
(Vallejos et al., 2018)	Soploon	The proposed tool is an Eclipse plugin which analyzes students' code and detects automatically novice programmer errors.

References

Abbasi, S., Kazi, H., & Khowaja, K. (2017, November). A systematic review of learning object oriented programming through serious games and programming approaches. In *2017 4th IEEE International Conference on Engineering Technologies and Applied Sciences (ICETAS)* (pp. 1-6). IEEE.

Alonso, M., & Py, D. (2009). An evaluation of pedagogical feedbacks in DIAGRAM, a learning environment for object-oriented modeling. Frontiers in Artificial Intelligence and Applications, 200(1), 653–655. Scopus. https://doi.org/10.3233/978-1-60750-028-5-653

Alonso, M., Py, D., & Lemeunier, T. (2008). A Learning Environment for Object-Oriented Modeling, Supporting Metacognitive Regulations. 2008 Eighth IEEE International Conference on Advanced Learning Technologies, 69–73. https://doi.org/10.1109/ICALT.2008.124

Ampatzoglou, A., Bibi, S., Avgeriou, P., Verbeek, M., & Chatzigeorgiou, A. (2019). Identifying, categorizing and mitigating threats to validity in software engineering secondary studies. Information and Software Technology, 106, 201–230. https://doi.org/10.1016/j.infsof.2018.10.006

Ardimento, P., Bernardi, M. L., & Cimitile, M. (2020). Software Analytics to Support Students in Object-Oriented Programming Tasks: An Empirical Study. IEEE Access, 8, 132171–132187. Scopus. https://doi.org/10.1109/ACCESS.2020.3010172

Azimullah, Z., An, Y. S., & Denny, P. (2020). Evaluating an Interactive Tool for Teaching Design Patterns. Proceedings of the Twenty-Second Australasian Computing Education Conference, 167–176. https://doi.org/10.1145/3373165.3373184

Blau, H., & Moss, J. E. B. (2015). FrenchPress gives students automated feedback on Java program flaws. 2015-June, 15–20. Scopus. https://doi.org/10.1145/2729094.2742622

Carter, J., & Fowler, A. (1998). Object oriented students?. *ACM SIGCSE Bulletin*, *30*(3), 271.

Cheah, C. S. (2020). Factors Contributing to the Difficulties in Teaching and Learning of Computer Programming: A Literature Review. Contemporary Educational Technology, 12(2), ep272. https://doi.org/10.30935/cedtech/8247

de Andrade Gomes, P. H., Garcia, R. E., Spadon, G., Eler, D. M., Olivete, C., & Messias Correia, R. C. (2017). Teaching software quality via source code inspection tool. 2017 IEEE Frontiers in Education Conference (FIE), 1–8. https://doi.org/10.1109/FIE.2017.8190658

Denegri, E., Frontera, G., Gavilanes, A., & Martín, P. J. (2008). A tool for teaching interactions between design patterns. Proceedings of the 13th Annual Conference on Innovation and Technology in Computer Science Education, 371. https://doi.org/10.1145/1384271.1384413

Dietrich, J., & Kemp, E. (2008). Tool Support for Teaching Design Patterns. 19th Australian Conference on Software Engineering (Aswec 2008), 200–208. https://doi.org/10.1109/ASWEC.2008.4483208

Dominique, P. Y., Auxepaules, L., & Alonso, M. (2013). Diagram, a learning environment for initiation to object-oriented modeling with UML class diagrams. Journal of Interactive Learning Research, 24(4), 425–446. Scopus.

Eckerdal, A., & Thuné, M. (2005). Novice Java programmers' conceptions of" object" and" class", and variation theory. *ACM SIGCSE Bulletin*, *37*(3), 89-93.

Fehnker, A., & de Man, R. (2019). Detecting and Addressing Design Smells in Novice Processing Programs. In B. M. McLaren, R. Reilly, S. Zvacek, & J. Uhomoibhi (Eds.), Computer Supported Education (pp. 507–531). Springer International Publishing. https://doi.org/10.1007/978-3-030-21151-6_24

Hashiura, H., Matsuura, S., & Komiya, S. (2010). A tool for diagnosing the quality of java program and a method for its effective utilization in education. 276–282. Scopus.

Herout, P., & Brada, P. (2015). Duck testing enhancements for automated validation of student programmes: How to automatically test the quality of implementation of students' programmes. 1, 228–234. Scopus. https://doi.org/10.5220/0005412902280234

Holland, S., Griffiths, R., & Woodman, M. (1997). Avoiding object misconceptions. In *Proceedings of the twenty-eighth SIGCSE technical symposium on Computer science education* (pp. 131-134).

Malliarakis, C., Satratzemi, M. & Xinogalos, S. (2012). Towards the Constructive Incorporation of Serious Games Within Object Oriented Programming. *Proceedings of the 6th European Conference on Games Based Learning (ECGBL 2012)*, 4-5 October, Cork, Ireland, 301-308.

Mirmotahari, O., Berg, Y., Gjessing, S., Fremstad, E., & Damsa, C. (2019). A Case-Study of Automated Feedback Assessment. 2019 IEEE Global Engineering Education Conference (EDUCON), 1190–1197. https://doi.org/10.1109/EDUCON.2019.8725249

Moraes, P., & Teixeira, L. (2019). Willow: A Tool for Interactive Programming Visualization to Help in the Data Structures and Algorithms Teaching-Learning Process. Proceedings of the XXXIII Brazilian Symposium on Software Engineering, 553–558. https://doi.org/10.1145/3350768.3351303

Ragonis, N., and Ben-Ari. M. (2005). A long-term investigation of the comprehension of OOP concepts by novices. *Int. J. Comput. Sci. Educ., 15*(3), 203–221.

Petersen, K., Vakkalanka, S., & Kuzniarz, L. (2015). Guidelines for conducting systematic mapping studies in software engineering: An update. Information and Software Technology, 64, 1–18. https://doi.org/10.1016/j.infsof.2015.03.007

Piteira, M., & Costa, C. (2013). Learning computer programming: Study of difficulties in learning programming. Proceedings of the 2013 International Conference on Information Systems and Design of Communication, 75–80. https://doi.org/10.1145/2503859.2503871

Satratzemi, M, Xinogalos, S., Tsompanoudi, D. (2023). Distributed Pair Programming in Higher Education: A Systematic Literature Review. *Journal of Educational Computing Research*, Volume *61*, Issue 3, pp. 546–577.

Silva, L., Mendes, A. J., & Gomes, A. (2020, April). Computer-supported collaborative learning in programming education: A systematic literature review. In *2020 IEEE Global Engineering Education Conference (EDUCON)* (pp. 1086-1095). IEEE.

Silva, V. J. S., & Dorça, F. A. (2019). An Automatic and Intelligent Approach for Supporting Teaching and Learning of Software Engineering Considering Design Smells in Object-Oriented Programming. 2019 IEEE 19th International Conference on Advanced Learning Technologies (ICALT), 2161-377X, 321–323. https://doi.org/10.1109/ICALT.2019.00100

Souza, D. M., Felizardo, K. R., & Barbosa, E. F. (2016, April). A systematic literature review of assessment tools for programming assignments. In *2016 IEEE 29th International Conference on Software Engineering Education and Training (CSEET)* (pp. 147-156). IEEE.

Spencer, D. (2009). Card Sorting: Designing Usable Categories (1st ed.). Rosenfeld Media.

Tan, P.-H., Ting, C.-Y., & Ling, S.-W. (2009). Learning Difficulties in Programming Courses: Undergraduates' Perspective and Perception. 2009 International Conference on Computer Technology and Development, 1, 42–46. https://doi.org/10.1109/ICCTD.2009.188

Thomasson, B., Ratcliffe, M., & Thomas, L. (2006). Identifying novice difficulties in object oriented design. *ACM SIGCSE Bulletin*, *38*(3), 28-32.

Vallejos, S., Berdun, L. S., Armentano, M. G., Soria, Á., & Teyseyre, A. R. (2018). Soploon: A virtual assistant to help teachers to detect object-oriented errors in students' source codes. Computer Applications in Engineering Education, 26(5), 1279–1292. https://doi.org/10.1002/cae.22021

Vliet, H. V. (2008). Software Engineering: Principles and Practices, John Wiley.

Xinogalos, S. & Satratzemi, M. (2004). Introducing Novices to Programming: a review of Teaching Approaches and Educational Tools. *Proceedings of the 2nd International Conference on Education and Information Systems, Technologies and Applications (EISTA 2004),* Orlando, Florida, USA, July 21-25, Vol. 2, 60-65.

Xinogalos, S. (2013). Using Flowchart-based Programming Environments for Simplifying Programming and Software Engineering Processes. In *Proceedings of 4th IEEE EDUCON Conference*, Berlin, Germany, 13-15 March 2013, IEEE Press, 1313-1322.

Xinogalos, S. (2015). Object Oriented Design and Programming: an Investigation of Novices' Conceptions on Objects and Classes. *ACM Transactions on Computing Education*, Vol. *15*, Issue 3, Article 13 (September 2015), 21 pages.

Xinogalos, S. (2016). Designing and deploying programming courses: Strategies, tools, difficulties and pedagogy. Education and Information Technologies, 21(3), 559–588. Scopus. https://doi.org/10.1007/s10639-014-9341-9

Yan, Y.-X., Wu, J.-P., Nguyen, B.-A., & Chen, H.-M. (2020). The Impact of Iterative Assessment System on Programming Learning Behavior. Proceedings of the 2020 9th International Conference on Educational and Information Technology, 89–94. https://doi.org/10.1145/3383923.3383939

Yang, J., Lee, Y., & Chang, K. H. (2018). Evaluations of JaguarCode: A web-based object-oriented programming environment with static and dynamic visualization. Journal of Systems and Software, 145, 147–163. https://doi.org/10.1016/j.jss.2018.07.037

Yang, J., Lee, Y., Hicks, D., & Chang, K. H. (2015). Enhancing object-oriented programming education using static and dynamic visualization. 2015 IEEE Frontiers in Education Conference (FIE), 1–5. https://doi.org/10.1109/FIE.2015.7344152

Yi Ding, Yongmin Hang, Gang Wan, & Shuiyan He. (2014). Application of software visualization in programming teaching. 2014 9th International Conference on Computer Science Education, 803–806. https://doi.org/10.1109/ICCSE.2014.6926573

Zaw, K. K., Funabiki, N., Mon, E. E., & Kao, W.-C. (2018). An Informative Test Code Approach for Studying Three Object-Oriented Programming Concepts by Code Writing Problem in Java Programming Learning Assistant System. 2018 IEEE 7th Global Conference on Consumer Electronics (GCCE), 629–633. https://doi.org/10.1109/GCCE.2018.8574687