Education and Information Technologies (2022) 27:7205–7239
https://doi.org/10.1007/s10639-022-10929-5

Ranking of problems and solutions in the teaching and learning of object-oriented programming

Luz E. Gutiérrez[1] · Carlos A. Guerrero[2] · Héctor A. López-Ospina[3]

Received: 4 August 2021 / Accepted: 31 January 2022 / Published online: 8 February 2022

Abstract

This study describes the most relevant problems and solutions found in the literature on teaching and learning of object-oriented programming (OOP). The identification of the problem was based on tertiary studies from the IEEE Xplore, Scopus, ACM Digital Library and Science Direct repositories. The problems and solutions identified were ranked through the multi-criteria decision methods DEMATEL and TOPSIS in order to determine the best solutions to the problems found and to apply these results in the academic context. The main contribution of this study was the categorization of OOP problems and solutions, as well as the proposal of strategies to improve the problem. Among the most relevant problems it was found: 1) difficulty in understanding, teaching and implementing object-orientation, 2) difficulties related to understanding classes and 3) difficulty in understanding object-oriented relationships. After doing the multicriteria analysis, it was found that the most important solutions to face the problems found in the teaching of OOP were: 1) use of active learning techniques and intrinsic rewards and 2) emphasize on basic programming concepts and introduce the object-oriented paradigm at an early point in the curriculum. As a conclusion, it was evidenced that there is coherence between the literary guarantee that gives support to the problems and solutions in the teaching of OOP presented in this study and the approaches that experts in the area of development highlight as relevant when they identify weaknesses in the process.

Keywords Object-oriented programming · Teaching-Learning · Multi-criteria · DEMATEL · TOPSIS

✉ Luz E. Gutiérrez
egluz@uninorte.edu.co

Extended author information available on the last page of the article

1 Introduction

For Martins et al. (2018) programming is the basis of a professional on systems. For such reason, it is required that programming courses are based on a model that allows putting into practice all the proposed theoretical approach. According to Popat and Starkey (2019) programming skills are 21st-century competences every person should enhance.

Martins et al. (2018) state that a learning process in the programming area should allow the student to identify a real problem and transform it into a sequence of activities that will finally be translated into a language. According to Qian et al. (2020) the teacher is the one in charge of guiding the students in this process of problem transformation, making the complexity implicit in the programming decrease and motivate them to continue. In addition, Dorn et al. (2018) affirm that although the teachers are facilitators, knowing the difficulties of a programming teaching process can allow them to implement pedagogical strategies that help the students in their training process.

Now, for Azmi et al. (2016) facilitating the teaching of algorithm design and programming is an activity that not only requires technical knowledge from the teacher, but also skills to motivate the students to overcome the obstacles that arise in their training. According to Martins et al. (2018) a non-motivated student is likely to increase the dropout statistics.

Ismail et al. (2018) state that the current teaching processes are not the most adequate, as is the case of "Teaching based solely on referring to the books seen to fail to attract the students' interest in learning", the authors state that "every educator should practice effective teaching methods to produce optimum outcomes. The success of a student lies in the way of teaching. Thus, it is important for teachers to study appropriate teaching methods that suit with the targeted students." Therefore, according with Draz et al. (2016) and Sarkar et al. (2016) traditional methods can create resistance in the student that will eventually be transformed into fear of programming.

In the work by Yang et al. (2015) and Qian and Lehman (2017) concepts such as variables, cycles and conditional structures are challenges for a student in training. However, in programming the most critical points are in abstract thinking and object-oriented programming (Hadar, 2013; Jordine et al., 2015; Krpan et al., 2015). A wrong training process leads the students to take a reactive attitude and to develop the idea that they do not have the ability or competences to continue in software development (Dorn et al., 2018).

More recent research work suggests that a deep knowledge of the teaching programming problems could allow the establishment of processes based on emerging methodologies such as the case of video games (Guerrero et al., 2020). These new teaching processes seek that the student generates a commitment and motivation to address the topics of study (Piteira et al., 2017).

Several paradigms such as structural programming, object-oriented programming, aspect-oriented programming, and reactive programming are identified in the programming area. This article will focus primarily on the object-oriented paradigm OOP.

Therefore, it is important to explore what factors affect the teaching and learning process of OOP within the context of every student and their environment. Thus, the following research question arises: What solutions could be prioritized in the resolution of problems in the teaching and learning of OOP?

The second section of this article describes the methods and materials used to identify and prioritize the OOP problem. The results section presents the most important findings related to the problem. The discussion section relates the observations of experts involved in the research and analysis of the results. Finally, the conclusions consolidate the contributions of the study.

2 Material and methods

2.1 Procedure for identifying OOP problems and solutions

A systematic literature review was carried out in order to identify the problems and solutions present in the teaching-learning process of object-oriented programming This review took as a reference the guidelines of the protocol proposed by Kitchenham et al. (2007). The PICOC strategy (Kitchenham & Charters, 2007) used in this research is presented below.

2.1.1 PICOC

According to Kitchenham and Charters (2007) the use of 5 criteria is suggested to define the research questions that will guide the search for the studies which will be part of the research: population, intervention, comparison, results, and context. These criteria are generally used in medicine and can be applied in the systems area. Population refers to the people affected by the intervention. The interventions which are usually a comparison between two or more alternative treatments. The outcomes are the clinical and economic factors that will be used to compare the interventions. The comparison refers to what the intervention is being compared with. The context refers to what is the context in which the intervention is delivered. The definition of each concept in the framework of this research is presented below:

Population: Corresponds to the literature related to topics that address the problems and solutions in the teaching-learning processes of object-oriented programming.
Intervention: The search string displayed in each one of the repositories made it possible to delimit the work to be done and established the field of intervention of the research.
Comparison: This concept was used in the present investigation when comparing the problems found.
Results: Identification of problems and solutions in the teaching-learning process of object-oriented programming.
Context: It is made up of the works that have as their foundation the study of object-oriented programming.

2.1.2 SLR Research Questions

The research questions proposed for carrying out the systematic literature review, which are supported by the PICOC criteria, are presented below.

Q1: What are the problems related to the learning and teaching of object-oriented programming?
Q2: What are the solutions to the problems found in the teaching and learning of object-oriented programming?

2.1.3 Search process

Once the research questions were defined and the keywords were identified, the generic search string was established to obtain the primary sources of the study. Fig. 1 presents the defined query string. This string is intended to identify tertiary papers that focus their studies towards teaching, learning or object-oriented paradigm skills. The "*" is used as a catch-all symbol to replace any combination of the words learn and teach, for example, it would apply learning and teaching.

The work carried out by Brereton et al. (2007) and Kitchenham and Charters (2007) was established as a reference for the selection of the search repositories. The selected repositories were IEEE Xplore, Scopus, ACM Digital Library and Science Direct.

3009 non-duplicate studies were found with the execution of the search string. For the Science Direct repository, an automatic filtering of the identified papers was performed with the VOSviewer tool (Van Eck & Waltman, 2010). The data obtained with the processing tool allowed to generate a term co-occurrence map that defines the most important topics for the present study in the Science Direct repository. The topics classified by the VOSviewer tool are shown in Fig. 2.

Due to the limited number of papers, automatic processing was not performed with the VOSviewer tool in the IEEE, Scopus and ACM repositories. The automatic processing performed with VOSviewer significantly reduced the number of pre-selected studies, identifying 945 relevant investigations for the present study.

2.1.4 Selection and exclusion criteria

The following selection criteria were applied to the title and abstract metadata of the 945 articles preselected in the previous stage.

((object oriented programming) OR (object oriented paradigm)) AND (((survey) OR (map) OR (review)) AND ((learn) OR (teach*) OR (skill)))*

Fig. 1 Search string

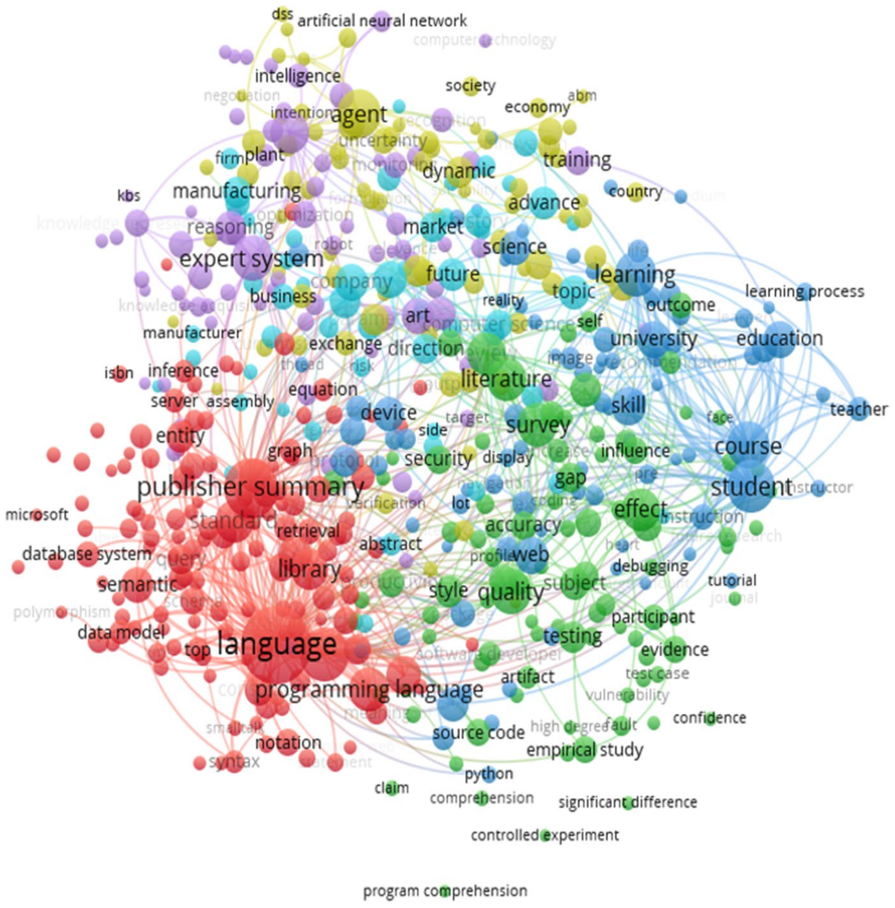

Fig. 2 Science Direct terms co-occurrence network map

SC1: Studies that address problems in the teaching-learning of object-oriented programming.
SC2: Studies that reference bibliography where the problems of object-oriented programming are identified.

After applying the selection criteria 87 papers remained. Then, exclusion criteria were applied to these 87 papers. As a result of this process 56 studies remained which formed the conceptual basis of the present investigation. The applied exclusion criteria are shown below:

EC1: Incomplete studies that do not present the details of the research.
EC2: Articles that do not allow access to their information.

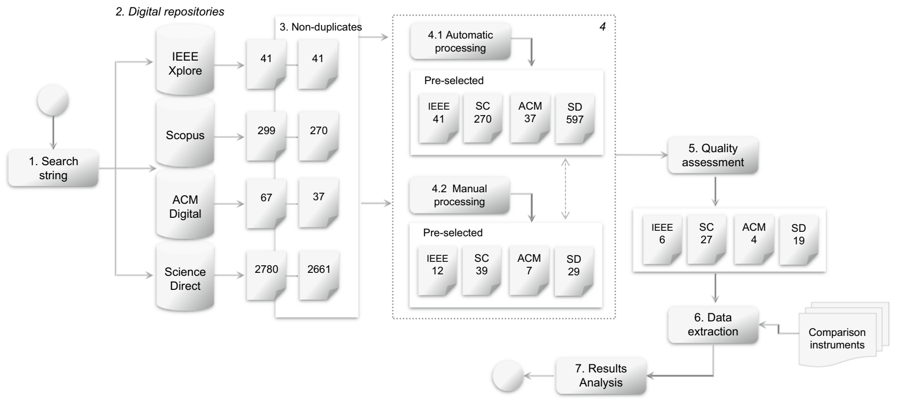

Fig. 3 Search and selection process

The complete process from definition of the search string, selection of repositories and selection and exclusion criteria application, is presented in Fig. 3 and Table 1.

2.2 Identified problems

A comparison matrix was made after analyzing the 56 selected studies. It allowed the identification of 14 problems related to the teaching and learning process of object-oriented programming. Each of these problems is described below.

Difficulty in understanding *object* and its dynamic nature (D01). This problem is referred as the students' conception of the term object as a simple record of a database. The students do not understand the aspect of behavior and variation as a function of the object's state (Hadar, 2013; Jordine et al., 2015; Karahasanović et al., 2007; Moons & De Backer, 2009; Moussa et al., 2016; Olsson & Mozelius, 2015; Rajashekharaiah et al., 2016; Sajaniemi et al., 2007; Sanders et al., 2008; Sheetz, 2002; Sheetz et al., 1997; Sien & Chong, 2011; Tegarden & Sheetz, 2001; Thomasson et al., 2006; Xinogalos, 2015; Yang et al., 2018).

Table 1 Results of the selection process

	IEEE	Scopus	ACM	Science	Total
Initial	41	299	67	2780	3187
Non-duplicates	41	270	37	2661	3009
Automatic processing	41	270	37	597	945
Manual processing	12	39	7	29	87
Quality assessment	6	27	4	19	56

Difficulties related to understanding *classes* (D02). This difficulty is described as the complexity presented by the students when assimilating the static nature and depth of classes. It is challenging for them to understand the hierarchy and the identification of correct classes. The students even refer to the difficulty in distinguishing between class and object. They generally assimilate class as a collection of objects, rather than an abstraction (Benander et al., 2004; Biddle & Tempero, 1998; Gorschek et al., 2010; Hadar, 2013; Hubwieser & Mühling, 2011; Karahasanović et al., 2007; Lewis et al., 2004; Moons & De Backer, 2009; Moussa et al., 2016; Musil & Richta, 2017; Nelson et al., 1997; Rajashekharaiah et al., 2016; Sajaniemi et al., 2007; Sanders et al., 2008; Sheetz, 2002; Sheetz et al., 1997, Sien, 2011, Sien & Chong, 2011, Tegarden & Sheetz, 2001, Thomasson et al., 2006; Xinogalos, 2015; J. Yang et al., 2018).
Difficulty in understanding the concept of *method* (D03). In this case it is referred as the complexity presented when assimilating the concept of method, there is no clarity on how to make the method calls. The students do not know how to determine the number of methods needed or what labels or names to assign to them. They do not understand how to reuse methods or their proper placement (Berges et al., 2012; Gorschek et al., 2010; Hubwieser & Mühling, 2011; Karahasanović et al., 2007; Moons & De Backer, 2009; Moussa et al., 2016; Olsson & Mozelius, 2015; Sajaniemi et al., 2007; Sanders et al., 2008; Sheetz et al., 1997; Tegarden & Sheetz, 2001).
Difficulty in understanding, teaching and implementing object-orientation (D04). This problem is specified as the challenge of performing object-oriented analysis, design, and programming. The students present difficulties when adopting the object-oriented paradigm, because their initial formative process is generally based on purely structural programming. The modular nature of the object-oriented paradigm is conceived as a challenge for educators, since in this process it is common for students to assimilate erroneous conceptions and to present problems in understanding and implementing object-oriented standards (Abbasi et al., 2017; Anniroot & de Villiers, 2012; Arif, 2000; Barr et al., 1999; Benander et al., 2004; Black et al., 2013; Cetin, 2013; Dale, 2006; Fedorowicz & Villeneuve, 1999; García Perez-Schofield et al., 2008; Hadar, 2013; Hosanee & Panchoo, 2015; Hubwieser & Mühling, 2011; Hundley, 2008; Jordine et al., 2015; Tahat, 2014; Karahasanović et al., 2007; Kunkle & Allen, 2016; Lewis et al., 2004; Mazaitis, 1993; Moussa et al., 2016; Nelson et al., 1997; Pei et al., 2010; Rajashekharaiah et al., 2016; Sajaniemi et al., 2007; Sanders et al., 2008; Seng & Yatim, 2014; Sheetz, 2002; Sheetz et al., 1997; Sien, 2011; Sien & Chong, 2011; Streib & Soma, 2010; Tan et al., 2014; Tegarden & Sheetz, 2001; Thomasson et al., 2006; Turner et al., 2010; Xinogalos, 2015; J. Yang et al., 2018; Zhang et al., 2018).
Difficulty in understanding object-oriented relationships (D05). It refers to the difficulty that the students have when understanding and implementing object-oriented relationships, such as association, dependency, generalization / specialization-inheritance, composition and aggregation. These problems are common due to the learners' lack of experience in relation to the object-oriented programming paradigm. The students generally present difficulties in the

process of modeling these relationships, and consequently in the implementation and application of concepts that are often conceived as complex (Barr et al., 1999; Benander et al., 2004; Berges et al., 2012; Biddle & Tempero, 1998; Dale, 2006; Gorschek et al., 2010; Hadar, 2013; Hosanee & Panchoo, 2015; Hundley, 2008; Karahasanović et al., 2007; Lewis et al., 2004; Moussa et al., 2016; Musil & Richta, 2017; Nelson et al., 1997; Olsson & Mozelius, 2015; Rajashekharaiah et al., 2016; Sheetz, 2002; Sheetz et al., 1997; Sien, 2011; Sien & Chong, 2011; Tegarden & Sheetz, 2001; Thomasson et al., 2006; J. Yang et al., 2018; Zhang et al., 2013).

Difficulty in understanding *polymorphism* and *overload* (D06). In this case it is indicated the high level of complexity the concepts of polymorphism and overload have at the moment of initiating a student into the programming area (Benander et al., 2004; Dale, 2006; Hosanee & Panchoo, 2015; Hundley, 2008; Lewis et al., 2004; Moussa et al., 2016; Rajashekharaiah et al., 2016; Sheetz, 2002; Sheetz et al., 1997; Tegarden & Sheetz, 2001; J. Yang et al., 2018).

Difficulty in understanding *encapsulation* (D07). This problem is related to the assimilation of several misconceptions related to understanding encapsulation, modularity and information hiding (Biddle & Tempero, 1998; Gorschek et al., 2010; Hubwieser & Mühling, 2011; Hundley, 2008; Karahasanović et al., 2007; Lewis et al., 2004; Moussa et al., 2016; Rajashekharaiah et al., 2016; Sanders et al., 2008; Sheetz, 2002; Sheetz et al., 1997; Sien & Chong, 2011; Tegarden & Sheetz, 2001; Turner et al., 2010; Xinogalos, 2015; J. Yang et al., 2018).

Complexity with the programming languages and tools used in the teaching and learning of object-orientation (D08). This problem is specified as the difficulty that students present with the use of debugging, navigation, testing and documentation tools. The change in technologies, paradigms and languages makes the learning process even more difficult (Barr et al., 1999; Benander et al., 2004; Bishop-Clark, 1995; García Perez-Schofield et al., 2008; Jiang et al., 2004; Jordine et al., 2015; Karahasanović et al., 2007, Kiss, 2013, Mazaitis, 1993, Moons & De Backer, 2009, Moons & De Backer, 2013; Nelson et al., 1997; Radenski, 2006; Rajashekharaiah et al., 2016; Sheetz, 2002; Sheetz et al., 1997; Tegarden & Sheetz, 2001; Thomasson et al., 2006; J. Yang et al., 2018; T.-C. Yang et al., 2015; Zainal et al., 2012; X. Zhang et al., 2018).

Difficulty in teaching and understanding general programming topics (D09). This difficulty refers to the challenges that the students face when understanding algorithms and basic programming concepts. Concepts such as variables, parameters, functions, and control structures are often considered difficult topics (Benander et al., 2004; Berges et al., 2012; Biddle & Tempero, 1998; Cetin, 2013; Dale, 2006; Govender, 2009; Hadar, 2013; Hubwieser & Mühling, 2011; Hundley, 2008; Jiang et al., 2004; Jordine et al., 2015; Karahasanović et al., 2007; Kiss, 2013; Krpan et al., 2015; Kunkle & Allen, 2016; Mazaitis, 1993; Moons & De Backer, 2009; Moons & De Backer, 2013; Moussa et al., 2016; Nelson et al., 1997; Olsson & Mozelius, 2015; Radenski, 2006; Sanders et al., 2008; Sheetz, 2002; Sheetz et al., 1997; Sien, 2011; Tan et al., 2014; Tegarden & Sheetz, 2001; Thomasson et al., 2006; T.-C. Yang et al., 2015; Zainal et al., 2012; J. Zhang et al., 2013).

Difficulty in developing abstract thinking (D10). This problem is related to the difficulty when understanding and solving real-world problems. The students frequently face processes where they must coordinate the acquired abstract thinking skills and the assimilated knowledge. This integration of skills and concepts challenges the student and, in many cases, makes the training process difficult (Anniroot & de Villiers, 2012; Biddle & Tempero, 1998; Black et al., 2013; Hadar, 2013; Hundley, 2008; Jordine et al., 2015; Karahasanović et al., 2007; Krpan et al., 2015; Olsson & Mozelius, 2015; Rajashekharaiah et al., 2016; Sheetz, 2002; Sheetz et al., 1997; Sien, 2011; Sien & Chong, 2011; Tegarden & Sheetz, 2001; Thomasson et al., 2006).

Difficulty in understanding software analysis and design (D11). It refers to the inability the students have to represent and design real-world problems. Students find challenges when using analysis and design techniques. They find it difficult to apply design concepts in Unified Modeling Language (UML) and to make use of related techniques and patterns (Anniroot & de Villiers, 2012; Biddle & Tempero, 1998; Bishop-Clark, 1995; Black et al., 2013; Hadar, 2013; Hundley, 2008; Tahat, 2014; Karahasanović et al., 2007; Lewis et al., 2004; Moons & De Backer, 2009; Rajashekharaiah et al., 2016; Sanders et al., 2008; Sheetz, 2002; Sheetz et al., 1997; Sien, 2011; Sien & Chong, 2011; Tegarden & Sheetz, 2001; Thomasson et al., 2006; Turner et al., 2010; J. Yang et al., 2018).

Difficulty in understanding *reuse* (D12). This is a quite recurrent problem. The learners do not understand when and where to reuse and they confuse this concept with the tendency to copy code, generating redundancy and duplication of information (Karahasanović et al., 2007; Sheetz, 2002; Sheetz et al., 1997; Tegarden & Sheetz, 2001).

Difficulty with project administration and management methodologies and techniques (D13). This problem refers to understanding activities that include time and resource restrictions. It is confusing for the learners to know when to stop, advance or finish the project (Karahasanović et al., 2007; Sheetz, 2002; Sheetz et al., 1997; Tegarden & Sheetz, 2001).

Difficulty in software implementation and maintenance (D14). This last problem is related to the difficulty students have in starting the software and in adding, subtracting or modifying the code to be adapted. These challenges demand significant amounts of time and effort, which generally causes apathy and disinterest in the process (Karahasanović et al., 2007; Tegarden & Sheetz, 2001).

2.3 Identified solutions

According to the literature review, six possible solutions to the problems of teaching-learning of object-oriented programming were found. Each of the possible solutions is described below.

Use of tools that support knowledge transfer (S01). This solution is described as an emerging proposal where virtualization, animation, online sessions and more channels that support knowledge transfer are used. Additionally, it is emphasized on the use of game-related tools and more suitable languages for teaching (Abbasi et al., 2017; Govender, 2009; Jordine et al., 2015; Kiss, 2013; Mazaitis, 1993; Moons & De Backer, 2009; Moons & De Backer, 2013; Olsson & Mozelius, 2015; Radenski, 2006; Sajaniemi et al., 2007; Seng & Yatim, 2014; Sheetz et al., 1997; Tan et al., 2014; J. Yang et al., 2018; T.-C. Yang et al., 2015).

Emphasize basic programming concepts and introduce the object-oriented paradigm at an early point in the curriculum (S02). It is considered that introducing the object-oriented paradigm at an early point in the curriculum make the students better understand the associated concepts. In addition, the basic concepts, such as class and object, must be emphasized, because they tend to be confused (Biddle & Tempero, 1998; Hundley, 2008; Mazaitis, 1993; Sanders et al., 2008; Tegarden & Sheetz, 2001).

Make use of UML diagrams, design patterns and simplified methodologies (S03). The use of the unified modeling language helps the students visualize and formulate programming concepts (Hundley, 2008; Jiang et al., 2004; Moons & De Backer, 2013; Sheetz et al., 1997; J. Yang et al., 2018).

Minimize aspects of the problem mastery, while learning object-oriented fundamentals (S04). This solution refers to emphasizing the resolution and mastery of the problem, putting aside the complexity of programming languages or development environments (Tegarden & Sheetz, 2001).

Use of active learning techniques and intrinsic rewards (S05). This solution is referred as the use of active learning techniques that involves peer tutoring, role-play activities, workshops, exemplifications, use of metaphors, and concept mapping (Jordine et al., 2015; Mazaitis, 1993; Moons & De Backer, 2013; Nelson et al., 1997; Sajaniemi et al., 2007; Sanders et al., 2008; Sien, 2011; Thomasson et al., 2006; T.-C. Yang et al., 2015; Zainal et al., 2012).

Change the way of teaching (S06). This solution refers to the change of teaching strategies, adapting the approach to the difficulties, achievements and mistakes of others. Thus, the learning is based on the students' experiences (Govender, 2009; Moons & De Backer, 2013; Tan et al., 2014; Thomasson et al., 2006; T.-C. Yang et al., 2015).

2.4 Prioritization of the identified problems

The DEMATEL (Decision Making Trial and Evaluation Laboratory) multi-criteria method is used for the prioritization and classification of the most relevant OOP problems (Espinosa & Salinas, 2013; Jeong & Ramírez-Gómez, 2018; López-Ospina et al., 2017). This method allows finding the relationships between the problems of this study, as well as their hierarchization depending on the decision-making context. In other words, it is assumed that there is a relationship between the problems. DEMATEL is a method that is considered effective for identifying the key components of the cause-effect chain of a complex system. It seeks to evaluate the interdependent relationships between factors and find the most critical or relevant ones through a visual structural model. This method provided the causal relationship between OOP problems and their importance ranking (Alzahrani et al., 2018; Aldowah et al., 2020). The steps of the DEMATEL method that were carried out in the present study are detailed below.

Table 2 Profile of the experts

Expert	Academic Degrees	Academic Experience	Productive Experience
Expert 1	a. Doctor in Engineering b. Master's Degree in Computer Science c. Systems Engineer	20	17
Expert 2	a. Master's Degree in Information Technology b. Systems and Computer Engineer	5	12
Expert 3	a. Master's Degree in Engineering, Computer Science and Informatics Area b. Systems Engineer	9	8

Table 3 Comparative scale

Scale	Value
No influence	0
Low influence	1
Medium influence	2
High influence	3
Very high influence	4

2.4.1 Step 1: Generation of the direct relationship matrix

The evaluation of the direct relationships of the problems was carried out by three experts in the field of object-oriented programming. The selected profiles were those who met: 1) experience of more than 5 years in the academic environment, 2) experience of more than 5 years in application development in the business sector and 3) professionals from different universities. Table 2 describes the experts' profiles.

The scale defined in Table 3 was used for the evaluation of the problems in order to find the influence relationship of the 14 problems identified in the teaching and learning of OOP. This scale is the one generally used in the applications of the DEMATEL method.

The 14 x 14 direct relationships matrix *A* was generated based on the information recorded by the experts (the problems have been described in section 2.2 Identified problems). Each expert evaluated the influence of each problem against the other ones to define the scale of influence among them. From this process, 3 evaluation matrices emerged, which later were averaged to generate the consolidated initial relationships matrix. Table 4 presents matrix *A* with the averages obtained.

2.4.2 Step 2: Normalization of the direct relationships matrix

The normalized matrix *M* was generated using equations (1) and (2). The objective of the transformation is to have a matrix with a norm less than 1. The results of *M* are presented in Table 5.

Table 4 Initial direct relationships matrix

	D01	D02	D03	D04	D05	D06	D07	D08	D09	D10	D11	D12	D13	D14
D01	0,0	2,7	3,3	4,0	3,0	4,0	4,0	2,3	2,3	2,0	1,3	0,7	0,3	0,3
D02	3,0	0,0	3,7	4,0	3,7	2,7	2,7	2,3	2,0	3,7	3,0	3,0	0,7	1,7
D03	2,0	3,0	0,0	3,3	2,3	4,0	2,7	2,0	2,7	2,0	1,3	3,7	0,3	1,3
D04	3,0	3,3	2,7	0,0	3,7	2,3	2,3	1,7	3,3	2,7	3,3	2,7	1,0	1,0
D05	2,0	3,0	2,0	4,0	0,0	1,7	1,7	2,3	2,3	3,7	3,3	1,3	0,3	0,7
D06	2,3	3,0	4,0	4,0	2,3	0,0	3,0	1,3	1,3	2,0	1,3	2,0	0,3	0,3
D07	1,3	1,0	1,3	3,3	3,0	3,0	0,0	1,0	1,0	2,0	1,7	1,7	0,3	0,3
D08	0,3	0,7	0,7	2,3	2,7	1,0	1,0	0,0	3,3	2,3	2,7	3,3	3,0	3,0
D09	3,3	3,0	3,3	2,7	3,0	2,0	2,0	1,7	0,0	2,3	2,7	2,0	1,7	2,0
D10	2,7	2,7	2,0	2,0	1,7	1,7	1,3	2,3	2,7	0,0	2,7	2,0	2,3	2,3
D11	2,0	3,0	1,7	1,7	3,3	1,0	1,3	2,0	2,7	3,7	0,0	2,0	1,3	1,7
D12	1,3	2,7	2,7	2,3	2,7	2,7	2,0	2,7	3,0	2,3	2,3	0,0	1,7	2,3
D13	0,3	0,3	0,3	0,3	0,0	0,0	0,0	2,3	2,0	2,0	1,7	1,7	0,0	2,0
D14	0,7	0,7	0,7	1,0	0,3	0,0	0,3	3,3	1,3	2,3	2,0	2,3	1,0	0,0

$$k = \min\left(\left(\frac{1}{\max\limits_{1\le i\le n} \sum_{j=1}^{n} \left|a_{ij}\right|}\right), \left(\frac{1}{\max\limits_{1\le j\le n} \sum_{i=1}^{n} \left|a_{ij}\right|}\right)\right) \quad i,j \in \{1,2,3,\ldots,n\} \tag{1}$$

$$M = k * A \tag{2}$$

Table 5 Normalized direct relationships matrix

	D01	D02	D03	D04	D05	D06	D07	D08	D09	D10	D11	D12	D13	D14
D01	0,00	0,08	0,09	0,11	0,09	0,11	0,11	0,07	0,07	0,06	0,04	0,02	0,01	0,01
D02	0,09	0,00	0,11	0,11	0,11	0,08	0,08	0,07	0,06	0,11	0,09	0,09	0,02	0,05
D03	0,06	0,09	0,00	0,09	0,07	0,11	0,08	0,06	0,08	0,06	0,04	0,11	0,01	0,04
D04	0,09	0,09	0,08	0,00	0,11	0,07	0,07	0,05	0,09	0,08	0,09	0,08	0,03	0,03
D05	0,06	0,09	0,06	0,11	0,00	0,05	0,05	0,07	0,07	0,11	0,09	0,04	0,01	0,02
D06	0,07	0,09	0,11	0,11	0,07	0,00	0,09	0,04	0,04	0,06	0,04	0,06	0,01	0,01
D07	0,04	0,03	0,04	0,09	0,09	0,09	0,00	0,03	0,03	0,06	0,05	0,05	0,01	0,01
D08	0,01	0,02	0,02	0,07	0,08	0,03	0,03	0,00	0,09	0,07	0,08	0,09	0,09	0,09
D09	0,09	0,09	0,09	0,08	0,09	0,06	0,06	0,05	0,00	0,07	0,08	0,06	0,05	0,06
D10	0,08	0,08	0,06	0,06	0,05	0,05	0,04	0,07	0,08	0,00	0,08	0,06	0,07	0,07
D11	0,06	0,09	0,05	0,05	0,09	0,03	0,04	0,06	0,08	0,11	0,00	0,06	0,04	0,05
D12	0,04	0,08	0,08	0,07	0,08	0,08	0,06	0,08	0,09	0,07	0,07	0,00	0,05	0,07
D13	0,01	0,01	0,01	0,01	0,00	0,00	0,00	0,07	0,06	0,06	0,05	0,05	0,00	0,06
D14	0,02	0,02	0,02	0,03	0,01	0,00	0,01	0,09	0,04	0,07	0,06	0,07	0,03	0,00

2.4.3 Step 3: Obtaining the total relationship matrix

Subsequently, the total relationship matrix *S* was generated using equation (3). Table 6 presents the data of matrix *S*. It contains the direct and indirect relationships between the problems.

$$S = M\,(I - M)^{-1} \tag{3}$$

2.4.4 Step 4: Determine the cause group and effect group

Based on equations (4), (5) and (6) a vector with the sum of the elements per rows of the matrix *S*, named *D*, was generated; then a vector with the sum of the elements per columns of *S*, named *R*, was generated.

$$S = \left|S_{ij}\right|_{nxn} \qquad i,j \in \{1, 2, \dots, n\} \tag{4}$$

$$D = \sum_{j=1}^{n} S_{ij} \tag{5}$$

$$R = \sum_{i=1}^{n} S_{ij} \tag{6}$$

Table 7 presents the calculation values of D+R and D-R. The positive values of D-R represent causes and the negative values are interpreted as the problems that are effect. A problem that is a Cause is one that originates or initiates the

Table 6 Total relationships matrix

	D01	D02	D03	D04	D05	D06	D07	D08	D09	D10	D11	D12	D13	D14
D01	0,25	0,37	0,38	0,45	0,40	0,37	0,36	0,31	0,35	0,37	0,32	0,29	0,14	0,18
D02	0,37	0,35	0,43	0,50	0,47	0,38	0,36	0,36	0,39	0,47	0,41	0,40	0,18	0,26
D03	0,31	0,38	0,29	0,44	0,38	0,37	0,32	0,31	0,36	0,37	0,32	0,37	0,15	0,22
D04	0,35	0,41	0,38	0,37	0,44	0,35	0,33	0,32	0,40	0,41	0,40	0,36	0,18	0,22
D05	0,29	0,36	0,32	0,43	0,30	0,29	0,28	0,30	0,34	0,40	0,36	0,29	0,14	0,19
D06	0,29	0,35	0,37	0,42	0,36	0,25	0,31	0,27	0,30	0,34	0,30	0,30	0,13	0,17
D07	0,21	0,24	0,24	0,33	0,31	0,27	0,17	0,21	0,23	0,28	0,25	0,24	0,11	0,13
D08	0,20	0,25	0,24	0,32	0,32	0,23	0,21	0,21	0,32	0,32	0,30	0,30	0,20	0,23
D09	0,34	0,38	0,38	0,42	0,40	0,32	0,30	0,31	0,29	0,38	0,36	0,33	0,19	0,24
D10	0,29	0,33	0,31	0,36	0,33	0,28	0,25	0,29	0,33	0,28	0,33	0,30	0,19	0,23
D11	0,28	0,34	0,30	0,35	0,37	0,26	0,25	0,28	0,33	0,38	0,26	0,30	0,16	0,21
D12	0,28	0,36	0,35	0,39	0,38	0,32	0,29	0,32	0,36	0,37	0,34	0,27	0,18	0,24
D13	0,10	0,12	0,12	0,14	0,12	0,10	0,09	0,17	0,17	0,18	0,16	0,16	0,06	0,13
D14	0,14	0,17	0,16	0,19	0,17	0,13	0,13	0,22	0,19	0,22	0,20	0,20	0,11	0,10

Table 7 Comparative D+R / D-R

	D01	D02	D03	D04	D05	D06	D07	D08	D09	D10	D11	D12	D13	D14
D+R	8,26	9,73	8,85	10,04	9,06	8,09	6,87	7,52	9,02	8,87	8,36	8,56	3,94	5,07
D-R	0,84	0,92	0,35	-0,20	-0,44	0,25	-0,45	-0,20	0,27	-0,68	-0,26	0,33	-0,29	-0,44

Table 8 Problems classified per group

ID	Problema	Grupo
D01	Difficulty in understanding object and its dynamic nature	Cause
D02	Difficulties related to understanding classes	Cause
D03	Difficulty in understanding the concept of method	Cause
D04	Difficulty in understanding, teaching and implementing object-orientation	Effect
D05	Difficulty in understanding object-oriented relationships	Effect
D06	Difficulty in understanding polymorphism and overload	Cause
D07	Difficulty in understanding encapsulation	Effect
D08	Complexity with the programming languages and tools used in the teaching and learning of object-orientation	Effect
D09	Difficulty in teaching and understanding general programming topics	Cause
D10	Difficulty in developing abstract thinking	Effect
D11	Difficulty in understanding software analysis and design	Effect
D12	Difficulty in understanding reuse	Cause
D13	Difficulty with project administration and management methodologies and techniques	Effect
D14	Difficulty in software implementation and maintenance	Effect

problem, whereas if a problem is an Effect, it means that it is the consequence of another problem. These results can be seen in Table 8.

2.4.5 Step 5: Set the threshold value and obtain the impact diagram

The threshold value was set at 0.2863 for matrix S based on equation (7).

$$threshold = \frac{\sum_{i=1}^{n} \sum_{j=1}^{n} S_{ij}}{n^2} \tag{7}$$

The diagram in Fig. 4 shows the distribution of the matrix S values. The minimum value is 0.063, 1st quartile is 0.2154, 2nd quartile is 0.3027, 3rd quartile is 0.3586 and the maximum value is 0.5049. As it can be seen in Fig. 4, there are no outliers of the S values. Additionally, the threshold value is less than the median and corresponds to the 41st percentile. It means that 41% of the scores are less than or equal to the threshold. This implies that 80 relationships among problems should

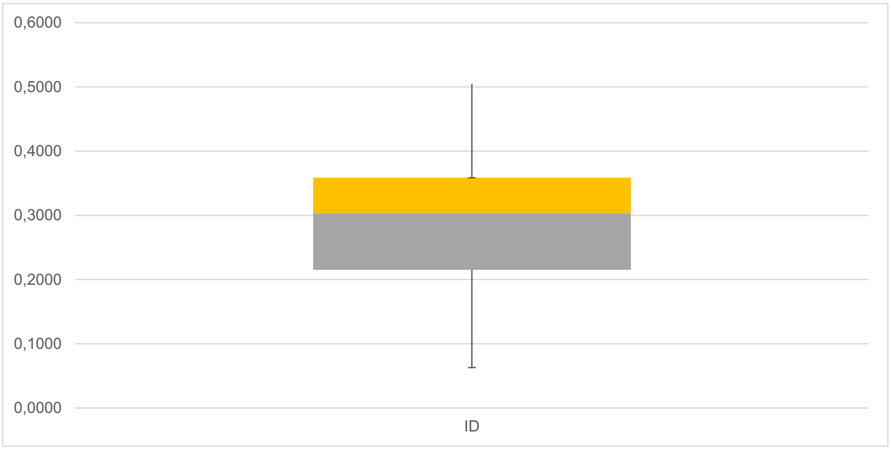

Fig. 4 Box plot of matrix *S*

be analyzed as key in the teaching-learning process. Given this number of relationships, it is proposed to do the aggregate causality analysis of Fig. 5.

Fig. 5 describes the causal diagram is built with the horizontal axis (D + R) called "Prominence" and the vertical axis (D - R) called "Relation". The horizontal axis shows the relative importance of each problem. On the other hand, vertical axis, splits problems into cause or effect. If (D - R) is positive, is a cause problem. Otherwise, it is an effect problem. For that reason, causal diagrams can visualize the causal relationships of problems into a visible structural model. According to the results obtained, D02's problem is the cause factor with the highest importance. On the other hand, D10's problem is the strongest effect factor and has a high weight. The problems classified as cause (D01, D02, D03, D06, D09, D12) have high weighting. However, some effect problems have a low importance.

2.4.6 Step 6: Weighting of problems

In this step, the problems that have the greatest weight in the teaching and learning process of OOP were identified. Equations (8) and (9) were used to weight the problems (Jeong & Ramírez-Gómez, 2018). The result of applying the equations is presented in Table 9 and Table 10.

$$W_i = \sqrt{\left(D_i + R_i\right)^2 + \left(D_i - R_i\right)^2} \tag{8}$$

$$W_i = \frac{W_i}{\sum_{i=1}^{n} W_i} \tag{9}$$

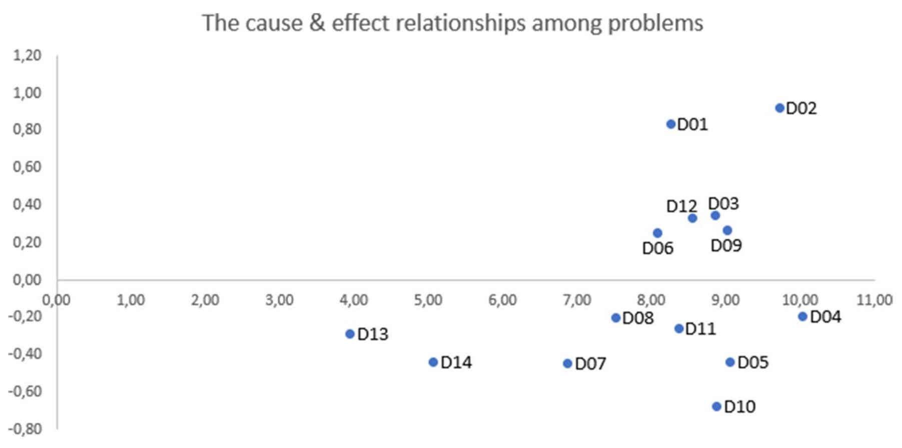

Fig. 5 D+R vs D-R relationship

After applying the weighting and standardization coefficients, the problems with the greatest weight in the teaching and learning process of OOP were identified (see Table 11).

2.5 Selection of solutions strategies

It is important to identify the possible solutions for the problems found in the process of teaching and learning of OOP. There is evidence in the analyzed literature where authors such as Gómez et al. (2020); Zhang et al. (2019) and Yi and Fang (2018) presented significant results when selecting solution strategies through multi-criteria methods such as TOPSIS (Technique for Order Preference by Similarity to Ideal Solution). The TOPSIS method handles the concept of the ideal solution and the anti-ideal solution when choosing decision alternatives. Its premise is based on the fact that a given alternative is located at the shortest distance from a positive ideal solution and at the greatest distance from a negative ideal solution. An ideal solution is defined as an ideal set of levels with respect to all the considered attributes of a given problem, even when the ideal solution is usually impossible or not feasible to obtain (Sun et al., 2016). The

Table 9 Weighting coefficient

D01	D02	D03	D04	D05	D06	D07	D08	D09	D10	D11	D12	D13	D14
8,30	9,78	8,86	10,04	9,07	8,09	6,88	7,53	9,02	8,90	8,37	8,56	3,95	5,09

Table 10 Standardized coefficent

D01	D02	D03	D04	D05	D06	D07	D08	D09	D10	D11	D12	D13	D14
0,07	0,09	0,08	0,09	0,08	0,07	0,06	0,07	0,08	0,08	0,07	0,08	0,04	0,05

Table 11 Ranking of problems

Problem	Value
D04	0,0893
D02	0,0870
D05	0,0806
D09	0,0803
D10	0,0791
D03	0,0788
D12	0,0762
D11	0,0744
D01	0,0739
D06	0,0720
D08	0,0669
D07	0,0612
D14	0,0453
D13	0,0351

steps used in the implementation of the TOPSIS method during the selection process of solution strategies analyzed in this study are listed below.

2.5.1 Step 1: Construction of the decision matrix

The same experts who supported the application process of the DEMATEL method were consulted for the evaluation of the matrix of solutions vs problems with the TOPSIS method. Each expert evaluated it based on the following guidelines: 1) identify how much influence a solution could have on each problem, 2) identify how much it could contribute to solving the problem and 3) how feasible it was to implement that solution. Each expert evaluated the influence of the solutions in each of the 14 problems, with values from 0 to 4, where 0 represents that there is no influence and 4 that there is a total influence of the solution to address the problem. The scale used is found in Table 3. From this process 3 matrices emerge, one for each evaluator. These matrices are averaged and the decision matrix presented in Table 12 is obtained.

2.5.2 Step 2: Normalization of the decision matrix

Subsequently, the decision matrix was normalized based on Equation (10). Table 13 presents the results of the normalized matrix.

$$v_{ij} = \frac{r_{ij}}{\sqrt{\sum_{i=1}^{m}\left(r_{ij}\right)^{2}}}, \forall j = 1, 2, \ldots, n \tag{10}$$

Table 12 Decision matrix

	D01	D02	D03	D04	D05	D06	D07	D08	D09	D10	D11	D12	D13	D14
S01	3,33	3,33	3,67	3,67	2,67	2,33	2,33	3,67	2,33	2,33	3,00	2,67	1,67	1,67
S02	4,00	4,00	3,67	4,00	4,00	3,33	3,33	3,67	3,67	3,33	3,33	3,67	1,67	2,00
S03	1,67	4,00	2,67	2,67	4,00	1,67	2,00	1,33	1,33	3,00	3,67	1,67	0,67	1,00
S04	1,33	1,67	1,33	1,00	1,33	0,67	0,67	2,00	1,33	1,33	1,67	1,33	1,33	1,00
S05	4,00	4,00	4,00	3,67	4,00	3,67	3,67	4,00	3,67	3,33	3,00	3,33	2,33	2,67
S06	2,00	2,00	2,00	2,00	2,00	1,67	1,67	2,33	2,00	2,33	2,00	1,67	1,67	2,00

2.5.3 Step 3: Construction of the weighted normalized decision matrix

The weighting vector was obtained with the DEMATEL method. The weighted normalized decision matrix was calculated by multiplying each W_j by each V_{ij}. Where W is the weight vector for each problem (Table 14), and V is the normalized decision matrix (Table 13). The results of this calculation are presented in Table 15.

2.5.4 Step 4: Determination of the ideal positive and negative solution

The objective of this step was to identify the positive and negative ideal solution, in order to calculate how close the OOP solutions were to the ideal ones. Formulas (11) and (12) were used for this process.

$$\overline{A^{+}} = \left\{\overline{A_1^{+}}, \overline{A_2^{+}}, \dots, \overline{A_n^{+}}\right\} = \left\{\left(\max_i \overline{v_{ij}}, j \in J\right)\left(\min_i \overline{v_{ij}}, j \in J'\right)\right\}; i = 1, 2, \dots, m \tag{11}$$

$$\overline{A^{-}} = \left\{\overline{A_1^{-}}, \overline{A_2^{-}}, \dots, \overline{A_n^{-}}\right\} = \left\{\left(\min_i \overline{v_{ij}}, j \in J\right)\left(\max_i \overline{v_{ij}}, j \in J'\right)\right\}; i = 1, 2, \dots, m \tag{12}$$

The results of Table 16 were obtained after applying formulas (11) and (12). For the case study of this article, the objective was to maximize the values of OOP solutions.

Table 13 Normalized decision matrix

	D01	D02	D03	D04	D05	D06	D07	D08	D09	D10	D11	D12	D13	D14
S01	0,46	0,41	0,49	0,49	0,34	0,39	0,38	0,50	0,37	0,35	0,43	0,43	0,42	0,37
S02	0,56	0,49	0,49	0,54	0,51	0,55	0,55	0,50	0,58	0,50	0,47	0,59	0,42	0,45
S03	0,23	0,49	0,36	0,36	0,51	0,28	0,33	0,18	0,21	0,45	0,52	0,27	0,17	0,22
S04	0,19	0,21	0,18	0,13	0,17	0,11	0,11	0,27	0,21	0,20	0,24	0,21	0,33	0,22
S05	0,56	0,49	0,54	0,49	0,51	0,61	0,60	0,54	0,58	0,50	0,43	0,53	0,58	0,60
S06	0,28	0,25	0,27	0,27	0,26	0,28	0,27	0,32	0,32	0,35	0,28	0,27	0,42	0,45

Table 14 Weight vector

	D01	D02	D03	D04	D05	D06	D07	D08	D09	D10	D11	D12	D13	D14
W	0,074	0,087	0,079	0,089	0,081	0,072	0,061	0,067	0,080	0,079	0,074	0,076	0,035	0,045

Table 15 Weighted normalized decision matrix

	D01	D02	D03	D04	D05	D06	D07	D08	D09	D10	D11	D12	D13	D14
S01	0,034	0,036	0,039	0,044	0,028	0,028	0,023	0,033	0,030	0,028	0,032	0,033	0,015	0,017
S02	0,041	0,043	0,039	0,048	0,041	0,040	0,033	0,033	0,047	0,040	0,035	0,045	0,015	0,020
S03	0,017	0,043	0,028	0,032	0,041	0,020	0,020	0,012	0,017	0,036	0,039	0,020	0,006	0,010
S04	0,014	0,018	0,014	0,012	0,014	0,008	0,007	0,018	0,017	0,016	0,018	0,016	0,012	0,010
S05	0,041	0,043	0,042	0,044	0,041	0,044	0,037	0,036	0,047	0,040	0,032	0,041	0,021	0,027
S06	0,021	0,021	0,021	0,024	0,021	0,020	0,017	0,021	0,025	0,028	0,021	0,020	0,015	0,020

2.5.5 Step 5: Calculation of distance measurements

In this step the positive and negative distance measures were calculated for each solution applying formulas (13) and (14).

$$\overline{d_i^+} = \sqrt{\sum_j \left(\overline{v_{ij}} - \overline{A_j^+}\right)^2} \tag{13}$$

$$\overline{d_i^-} = \sqrt{\sum_j \left(\overline{v_{ij}} - \overline{A_j^-}\right)^2} \tag{14}$$

The results of this arithmetic procedure are presented in Table 17.

2.5.6 Step 6: Calculation of relative proximity

The last step in the TOPSIS method was the calculation of relative proximity, a procedure based on equation (15).

$$\overline{RS_i} = \frac{d_i^-}{d_i^+ + d_i^-};i = 1, 2, \ldots , m \tag{15}$$

Table 18 presents the results of the application of the relative proximity equation. Subsequently, the values were organized from the highest to the lowest weight, in order to identify the solutions that were closest to the solution ideal as described in Table 20.

3 Results

3.1 Influence relationship between the OOP problems

According to the DEMATEL results, Table 19 describes how much one problem affects another and how much it is affected by another. For the case of problem "D02 - Difficulties related to the understanding classes", it influences to a high degree problem "D04 - Difficulty in understanding, teaching and implementing object-orientation". This would be an expected result because, if a student does not handle the concept of classes correctly, it will be reflected when understanding the object-oriented paradigm.

Ten problems affect problem "D04 - Difficulty in understanding, teaching and implementing object orientation", indicating that this problem is the one that receives the most influence from the other factors. The problem that most affects the other problems is “D02 - Difficulties related to understanding classes”, with an occurrence of 11 out of 14.

Table 16 Positive and negative ideal per solution

	D01	D02	D03	D04	D05	D06	D07	D08	D09	D10	D11	D12	D13	D14
A+	0,041	0,043	0,042	0,048	0,041	0,044	0,037	0,036	0,047	0,040	0,039	0,045	0,021	0,027
A-	0,014	0,018	0,014	0,012	0,014	0,008	0,007	0,012	0,017	0,016	0,018	0,016	0,006	0,010

Table 17 Positive and negative ideal of solutions

Solution	d_i^+	d_i^-
S01	0,0391	0,0667
S02	0,0463	0,0994
S03	0,0655	0,0560
S04	0,0993	0,0228
S05	0,0414	0,1021
S06	0,0703	0,0419

Table 18 Relative proximity

	S01	S02	S03	S04	S05	S06
RSi	0,63	0,68	0,46	0,19	0,71	0,37

Table 19 Influence relationship between problems

Problem	Affected the most		Affects it the most	
D01	D04	0,4537	D02	0,3708
D02	D04	0,5049	D04	0,4077
D03	D04	0,4357	D02	0,4296
D04	D05	0,4416	D02	0,5049
D05	D04	0,4276	D02	0,4689
D06	D04	0,4234	D02	0,3811
D07	D04	0,3310	D02	0,3594
D08	D04	0,3233	D02	0,3583
D09	D04	0,4199	D04	0,3996
D10	D04	0,3564	D02	0,4667
D11	D10	0,3800	D02	0,4134
D12	D04	0,3918	D02	0,3972
D13	D10	0,1796	D08	0,1977
D14	D10	0,2220	D02	0,2559

3.2 Towards a generic framework

The construction of the Ranking allowed to define a framework for implementing solutions to problems in the object-oriented programming teaching. As a general contribution of this research, the steps to implement strategies that help in the object-oriented programming teaching-learning process are presented. Fig. 6 describes the stages and activities to be carried out for this purpose.

The validation stage is part of the next phase of this research in which instruments to evaluate the perception of students and teachers after applying the suggested strategies will be built. From this activity a feedback process will emerge for the proper calibration of such instruments and strategies.

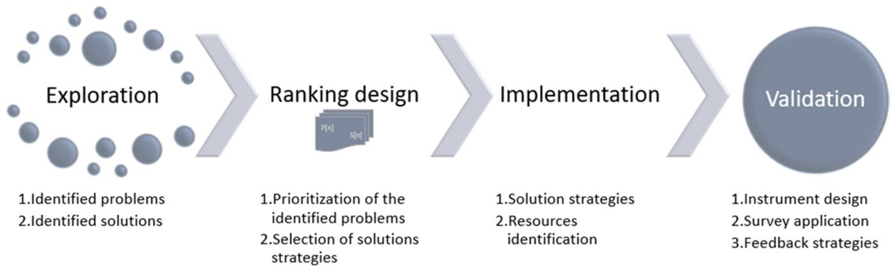

Fig. 6 Generic framework

3.3 Strategies to implement the best solutions

As it can be seen in Table 20, the top ranked solutions is "S05 - Use of active learning techniques and intrinsic rewards", followed by "S02 - Emphasize basic programming concepts and introduce the object-oriented paradigm at an early point in the curriculum", and in the third position "S01 - Use of tools that support knowledge transfer".

The solutions that are closest to the ideal are the best strategies to help minimize the 14 problems in this study (Table 20). Recommendations for the implementation of the solutions ranked in the top 3 positions are provided below.

3.3.1 Strategies for the use of active learning techniques and intrinsic rewards (S05)

For the implementation of the solution "Use of active learning techniques and intrinsic rewards", three strategies are proposed as it can be seen in Table 21.

3.3.2 Strategies to emphasize basic programming concepts and introduce the object-oriented paradigm at an early in the curriculum (S02).

For the implementation of the solution "Emphasize basic programming concepts and introduce the object-oriented paradigm at an early point in the curriculum", four strategies are proposed as it can be seen in Table 22.

3.3.3 Strategies for the use of tools that support knowledge transfer (S01).

For the implementation of the solution "Use of tools that support knowledge transfer", three strategies are proposed as it can be seen in Table 23.

Table 20 Solutions ranking

Position	Ranking	
1	S05	0,71
2	S02	0,68
3	S01	0,63
4	S03	0,46
5	S06	0,37
6	S04	0,19

4 Discussion

The problems and solutions identified in the literature review phase present some suggestions by the panel of experts. For expert 1 problems D11, D13 and D14 are not part of the scope of OOP teaching and learning. Based on this statement, he recommends that these problems should not be taken into account in future research. He also mentions that there are relevant problems that were not explicitly identified in the findings, such as the programming language syntax requirement. By having a diversity of programming languages, each one will have its own statements and typing rules, making the teaching-learning process of OOP difficult. Another factor that influences the teaching of OOP is the pedagogical preparation of teachers. Instructors are usually experts in informatics or computer science; however, teaching methodology is not an area of their expertise.

On the other hand, expert 2 affirms that, although problems D13 and D14 are not directly related to the teaching and learning of OOP, it is important to evaluate them within the framework of a complete process of analysis of this problem. For expert 2, the problems identified group together the OOP universe.

Regarding the level of complexity of each problem, expert 3 states that the problems identified in the literature review are at different levels, it means that there are generic problems and other more specific ones. In this regard, he recommends working with problems D01 to D10.

Regarding the solutions found, Expert 1 considers that appropriate preparation in methodological and pedagogical issues is important in the teaching staff that guides programming topics.

Experts 1 and 2 suggest that ideally solution "S06 - Change the way of teaching" would be the optimal solution for the OOP teaching and learning problems. However, in the practice it is not feasible to implement it, because it implies the construction of individual and personalized teaching activities.

Expert 3 considers that solutions S01 and S05 have a great similarity in their concept, for this reason, they can be considered as the same solution when implementing them.

When analyzing the results of Table 11, it is possible to see that the problem with the greatest weight is D04 “Difficulty in understanding, teaching and implementing object orientation”. According to the decision matrix of the TOPSIS method this problem may be solved by implementing S02 solution. This solution is very important, because the basis of OOP lies in understanding the concept of class and object; and as this research has proved these concepts are often confused.

The second most important problem was D02 “Difficulties related to understanding classes”. This problem, according to the matrix in Table 12, can be addressed from 3 solutions: S02, S03 and S05 where only S03 did not appear in the ranking of solutions provided by the TOPSIS. However, it must be taken into account for this specific problem.

In general, the solutions of the ranking have in common that the change in the teaching and learning methodologies of OOP is prevailing. It is not advisable to maintain the same strategies for lectures and to present merely theoretical concepts.

Table 21 Strategies for S05

Suggested strategy	Implications	Level
1. Solve problems through computer games. The economic cost of using this strategy can be high, however, free-to-use games are now available.	Acquire tools based on gamification for teaching. Curriculum adjustments.	High
2. Perform intelligence challenges. Present problems with several solutions to teach the student that it is possible to solve a situation in different ways.	Add extra work to teachers to review different possible solutions for each assigned project.	Medium
3. Training of teachers in the use of gamification in their academic space. At financial level it does not require a high investment because there are online activities and courses on this topic. What is most important is a change of attitude towards the use of new strategies in class.	Allocate time and resources to train teachers. Acquire tools based on gamification for teaching.	Medium

Table 22 Strategies for S02

Suggested strategy	Implications	Level
1. Carry out curricular updates in systems engineering and related programs, with the objective of introducing the subject of OOP since the first year of the major.	Request changes in the organization of academic programs before the relevant organizations.	Medium
2. Design projects with real programming topics to build skills on the students. These projects must be integrated with other subjects such as databases.	Create project banks so that students can select work according to their preference.	Low
3. Focus the implementation of code in early stages to specific problems. This implies minimizing the complexity in the writing of the problem, in such a way that the student does not feel overwhelmed just by reading the problem statement.	Allow students to solve specific problems that can be reused in different contexts, seeking the student to repeat and apply a solution many times.	Low
4. Use programming environments in early stages, ensuring a training that includes most of the features of the environment, even without coding. The students express fear when using a tool because of their lack of knowledge of it.	Develop early skills in the use of development tools.	Low

Table 23 Strategies for S01

Suggested strategy	Implications	Level
1. Define educational activities and object-oriented programming topics to be learned through the gamified learning environment, supported by the curriculum of each academic program. Individual work is prioritized over group work throughout the course. However, there are activities at the end of the academic space that seek to encourage and harmonize teamwork.	Integrate gamification tools into the teaching process that allow to enhance the use of OOP.	Medium
2. Develop the subjects with workshops and real cases, for this reason the project-based learning method is recommended, intending to make the students propose their own algorithms to provide solutions to real problems with the teacher's guidance.	Prepare additional material to allow students to carry out permanent practices in order to solve each of the proposed problems.	Low
3. Verify and establish rewards for the development of the selected modules of the gamified learning environment.	Propose a reward scheme based on the advance of the students.	Low

It is important to have students involved directly in the process and not continue thinking that they are actors who only receive information. As Ismail et al. (2018) well stated it, “This gamification approach to education is considered as one of the new teaching era that it is capable of improving students’ achievement”.

5 Conclusions

This article presents an analysis of the OOP problem in 3 phases. The first phase of the study sought to identify the problems and solutions of OOP. This process of problem recognition was carried out through a systematic literature review in high-impact repositories. The second phase was the hierarchization of the OOP problems by means of the DEMATEL method; with this method, it was possible to identify the most relevant problems in the teaching-learning process of OOP and the relationship level between these problems. The third phase was the making up of the solutions ranking that was developed through the TOPSIS method, as well as the proposal of implementing strategies for each of the solutions located in the first three positions of the ranking. The contribution of this research can be focused on 3 aspects: 1) List of problems categorized from the greatest to the least relevance. 2) Ranking of solutions and 3) Strategies to implement the best solutions. The conclusions of this work according to the phases of the investigation are listed below.

It was possible to obtain information about the problems, causes and solutions of the teaching-learning process of object-oriented programming from the systematic literature review process carried out. These results show the interest of the academic community in presenting alternatives to implement strategies to improve this process.

According to the results obtained when applying the multicriteria techniques, the problem “D02 - Difficulties related to understanding classes” was identified as a cause in the DEMATEL method and had a high weighting value. This indicates the importance of emphasizing this subject in the classes, in order to generate adequate conceptual bases for programming students.

Based on the TOPSIS method results, it is found that the top ranked solution is "Strategies for the use of active learning techniques and intrinsic rewards". This finding reinforces the need for a change in OOP teaching strategies. For such reason, the use of tools related to computer games and the search for new teaching strategies that motivate students in their formative process can support the learning of object-oriented programming.

It was evidenced that there is coherence between the literary guarantee that supports the problems and solutions in the teaching of OOP presented in this study and the approaches that experts in the development area highlight as relevant when identifying weaknesses in the process.

The use of multi-criteria decision methods made it possible to identify the relationships between the OOP problems, as well as to prioritize the problems and solutions.

Taking into account the current world situation with the Covid-19 pandemic, the application of different teaching-learning strategies in all areas of study is imperative. In the case of OOP, strategies aimed at the use of tools that support knowledge transfer (S01) and the use of active learning techniques and intrinsic rewards (S05) are particularly important. It is because work mediated by technology and strengthening of individual skills is a priority with the current situation, where remote work and virtuality will be the new study scenarios.

6 Future research direction

This study laid the conceptual foundations in the identification of problems and solutions based on literature review. For this reason, it is necessary to carry out an analysis with a representative sample group, where OOP teachers and students of this subject help to identify which, from their experience, are the problems and solutions for the teaching of OOP by means of an evaluation instrument.

At the university where the authors belong, teacher evaluation processes are carried out twice each year, specifically, once for each academic semester. These results can feed the problems database that arise in the teaching of OOP.

The next phase of this research consists of implementing strategies to respond to the solution “Strategies for the use of active learning techniques and intrinsic rewards (S05)”. At this stage, a video game will be developed at the University of origin to teach concepts such as method, polymorphism, and encapsulation.

Acknowledgements This work was carried out with the assistance provided by Universidad Santo Tomás, Universidad del Norte and the program "Convocatoria de Doctorados Nacionales 785 de 2017". The authors would like to thank Universidad Santo Tomás and Universidad del Norte for allowing the articulation of an interdisciplinary team for the research project development. Thanks are extended to the experts who provided their valuable contributions.

Availability of data and materials Not applicable.

Code availability Not applicable.

Authors' contributions **Luz E. Gutiérrez:** Conceptualization, Methodology, Software, Formal analysis, Investigation, Writing - Original Draft, Writing - Review & Editing.

Carlos A. Guerrero: Conceptualization, Methodology, Investigation, Writing - Original Draft, Writing - Review & Editing, Supervision.

Héctor A. López-Ospina: Methodology, Formal analysis, Writing - Review & Editing.

Funding information Universidad Santo Tomás. CAFP 002 SIST.2018-2 and Convocatoria Doctorados Nacionales 785. The funding bodies participated in the design of the study and collection, analysis, and interpretation of data and in writing the manuscript.

Declarations

Conflicts of interest/Competing interests The authors declare that they have no competing interests.

Ethics approval Not applicable.

Consent to participate Not applicable.

Consent for publication Not applicable.

References

Abbasi, S., Kazi, H., & Khowaja, K. (2017). A systematic review of learning object oriented programming through serious games and programming approaches. *2017 4th IEEE International Conference on Engineering Technologies and Applied Sciences (ICETAS)*, *2018-Janua*, 1–6. https://doi.org/10.1109/ICETAS.2017.8277894

Aldowah, H., Al-Samarraie, H., Alzahrani, A. I., & Alalwan, N. (2020). Factors affecting student dropout in MOOCs: a cause and effect decision-making model. *Journal of Computing in Higher Education, 32*(2), 429–454. https://doi.org/10.1007/s12528-019-09241-y

Alzahrani, A. I., Al-Samarraie, H., Eldenfria, A., & Alalwan, N. (2018). A DEMATEL method in identifying design requirements for mobile environments: students' perspectives. *Journal of Computing in Higher Education, 30*(3), 466–488. https://doi.org/10.1007/s12528-018-9176-2

Anniroot, J., & de Villiers, M. R. (2012). A study of Alice: A visual environment for teaching object-oriented programming. *Proceedings of the IADIS International Conference on Information Systems 2012*, *2*, 251–258.

Arif, E. M. (2000). A methodology for teaching object-oriented programming concepts in an advanced programming course. *ACM SIGCSE Bulletin, 32*(2), 30–34. https://doi.org/10.1145/355354.355367

Azmi, S., Iahad, N. A., & Ahmad, N. (2016). Attracting students' engagement in programming courses with gamification. *2016 IEEE Conference on E-Learning, e-Management and e-Services (IC3e)*, 112–115. https://doi.org/10.1109/IC3e.2016.8009050

Barr, M., Holden, S., Phillips, D., & Greening, T. (1999). An exploration of novice programming errors in an object-oriented environment. *Working Group Reports from ITiCSE on Innovation and Technology in Computer Science Education - ITiCSE-WGR '99*, *Part F1291*(4), 42–46. https://doi.org/10.1145/349316.349392

Benander, A., Benander, B., & Sang, J. (2004). Factors related to the difficulty of learning to program in Java—an empirical study of non-novice programmers. *Information and Software Technology, 46*(2), 99–107. https://doi.org/10.1016/S0950-5849(03)00112-5

Berges, M., Mühling, A., & Hubwieser, P. (2012). The gap between knowledge and ability. *Proceedings of the 12th Koli Calling International Conference on Computing Education Research - Koli Calling '12*, 126–134. https://doi.org/10.1145/2401796.2401812

Biddle, R., & Tempero, E. (1998). Teaching programming by teaching principles of reusability. *Information and Software Technology, 40*(4), 203–209. https://doi.org/10.1016/S0950-5849(98)00040-8

Bishop-Clark, C. (1995). Cognitive style, personality, and computer programming. *Computers in Human Behavior, 11*(2), 241–260. https://doi.org/10.1016/0747-5632(94)00034-F

Black, A. P., Bruce, K. B., Homer, M., Noble, J., Ruskin, A., & Yannow, R. (2013). Seeking Grace: A New Object-Oriented Language for Novices. *Proceeding of the 44th ACM Technical Symposium on Computer Science Education - SIGCSE '13*, 129. https://doi.org/10.1145/2445196.2445240

Brereton, P., Kitchenham, B. A., Budgen, D., Turner, M., & Khalil, M. (2007). Lessons from applying the systematic literature review process within the software engineering domain. *Journal of Systems and Software, 80*(4), 571–583. https://doi.org/10.1016/j.jss.2006.07.009

Cetin, I. (2013). Visualization: a tool for enhancing students' concept images of basic object-oriented concepts. *Computer Science Education, 23*(1), 1–23. https://doi.org/10.1080/08993408.2012.760903

Dale, N. B. (2006). Most difficult topics in CS1: Results of an Online Survey of Educators. *ACM SIGCSE Bulletin, 38*(2), 49–53. https://doi.org/10.1145/1138403.1138432

Dorn, N., Berges, M., Capovilla, D., & Hubwieser, P. (2018). Talking at Cross Purposes - Perceived Learning Barriers by Students and Teachers in Programming Education. *Proceedings of the 13th Workshop in Primary and Secondary Computing Education*, 1–4. https://doi.org/10.1145/3265757.3265769

Draz, A., Abdennadher, S., & Abdelrahman, Y. (2016). Kodr: A Customizable Learning Platform for Computer Science Education. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics): Vol. 9891 LNCS* (pp. 579–582). https://doi.org/10.1007/978-3-319-45153-4_67

Espinosa, F. F., & Salinas, G. E. (2013). Selección de Estrategias de Mejoramiento de las Condiciones de Trabajo para la Función Mantenimiento Utilizando la Metodología MCDA Constructivista. *Información Tecnológica, 24*(3), 57–72. https://doi.org/10.4067/S0718-07642013000300008

Fedorowicz, J., & Villeneuve, A. O. (1999). Surveying object technology usage and benefits: A test of conventional wisdom. *Information & Management, 35*(6), 331–344. https://doi.org/10.1016/S0378-7206(98)00098-6

García Perez-Schofield, J. B., García Roselló, E., Ortín Soler, F., & Pérez Cota, M. (2008). Visual Zero: A persistent and interactive object-oriented programming environment. *Journal of Visual Languages & Computing, 19*(3), 380–398. https://doi.org/10.1016/j.jvlc.2007.11.002

Gómez, J. C. O., Tabares-Urrea, N., & Ramírez-Flórez, G. (2020). AHP difuso para la selección de un proveedor 3PL considerando el riesgo operacional. *Revista EIA*, *17*(33), 1–17. https://doi.org/10.24050/reia.v17i33.1329

Gorschek, T., Tempero, E., & Angelis, L. (2010). A large-scale empirical study of practitioners' use of object-oriented concepts. *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering - ICSE '10*, *1*, 115. https://doi.org/10.1145/1806799.1806820

Govender, I. (2009). The learning context: Influence on learning to program. *Computers & Education, 53*(4), 1218–1230. https://doi.org/10.1016/j.compedu.2009.06.005

Guerrero, C. A., Gutiérrez, L. E., & Cuervo, K. D. (2020). Los videojuegos como estrategia para incrementar la motivación y alcance de logros en procesos de aprendizaje. *Memorias La Formación de Ingenieros: Un Compromiso Para El Desarrollo y La Sostenibilidad*, 1–9. https://acofipapers.org/index.php/eiei/article/view/750/755

Hadar, I. (2013). When intuition and logic clash: The case of the object-oriented paradigm. *Science of Computer Programming, 78*(9), 1407–1426. https://doi.org/10.1016/j.scico.2012.10.006

Hosanee, Y., & Panchoo, S. (2015). An enhanced software tool to aid novices in learning Object Oriented Programming (OOP). *2015 International Conference on Computing, Communication and Security (ICCCS)*, 1–7. https://doi.org/10.1109/CCCS.2015.7374197

Hubwieser, P., & Mühling, A. (2011). What students (should) know about object oriented programming. *Proceedings of the Seventh International Workshop on Computing Education Research - ICER '11*, 77. https://doi.org/10.1145/2016911.2016929

Hundley, J. (2008). A review of using design patterns in CS1. *Proceedings of the 46th Annual Southeast Regional Conference on XX - ACM-SE 46*, 30. https://doi.org/10.1145/1593105.1593113

Ismail, M. E., Sa'adan, N., Samsudin, M. A., Hamzah, N., Razali, N., & Mahazir, I. I. (2018). Implementation of The Gamification Concept Using KAHOOT! Among TVET Students: An Observation. *Journal of Physics: Conference Series, 1140*(1), 1–8. https://doi.org/10.1088/1742-6596/1140/1/012013

Jeong, J. S., & Ramírez-Gómez, Á. (2018). Optimizing the location of a biomass plant with a fuzzy-DEcision-MAking Trial and Evaluation Laboratory (F-DEMATEL) and multi-criteria spatial decision assessment for renewable energy management and long-term sustainability. *Journal of Cleaner Production, 182*, 509–520. https://doi.org/10.1016/j.jclepro.2017.12.072

Jiang, K., Maniotes, J., & Kamali, R. (2004). A different approach of teaching introductory visual basic course. *Proceedings of the 5th Conference on Information Technology Education - CITC5 '04*, 219–223. https://doi.org/10.1145/1029533.1029586

Jordine, T., Liang, Y., & Ihler, E. (2015). A Mobile Device Based Serious Gaming Approach for Teaching and Learning Java Programming. *International Journal of Interactive Mobile Technologies (IJIM), 9*(1), 53–59. https://doi.org/10.3991/ijim.v9i1.4380

Karahasanović, A., Levine, A. K., & Thomas, R. (2007). Comprehension strategies and difficulties in maintaining object-oriented systems: An explorative study. *Journal of Systems and Software, 80*(9), 1541–1559. https://doi.org/10.1016/j.jss.2006.10.041

Kiss, G. (2013). Teaching Programming in the Higher Education not for Engineering Students. *Procedia - Social and Behavioral Sciences, 103*, 922–927. https://doi.org/10.1016/j.sbspro.2013.10.414

Kitchenham, B., Brereton, P., Budgen, D., Turner, M., Bailey, J., & Linkman, S. (2007). *Protocol for a Tertiary study of Systematic Literature Reviews and Evidence-based Guidelines in IT and Software Engineering*. https://www.semanticscholar.org/paper/Protocol-for-a-Tertiary-study-of-Systematic-Reviews-Kitchenham-Brereton/bf3910a40028240b8217907388969546389 32e33#paper-header

Kitchenham, B., & Charters, S. (2007). *Guidelines for performing Systematic Literature Reviews in Software Engineering*. https://www.elsevier.com/__data/promis_misc/525444systematicreviewsguide.pdf

Krpan, D., Mladenović, S., & Rosić, M. (2015). Undergraduate Programming Courses, Students' Perception and Success. *Procedia - Social and Behavioral Sciences, 174*, 3868–3872. https://doi.org/10.1016/j.sbspro.2015.01.1126

Kunkle, W. M., & Allen, R. B. (2016). The Impact of Different Teaching Approaches and Languages on Student Learning of Introductory Programming Concepts. *ACM Transactions on Computing Education, 16*(1), 1–26. https://doi.org/10.1145/2785807

Lewis, T. L., Rosson, M. B., & Pérez-Quiñones, M. A. (2004). What do the experts say? Teaching Introductory Design from an Expert's Perspective. *ACM SIGCSE Bulletin, 36*(1), 296–300. https://doi.org/10.1145/1028174.971405

López-Ospina, H., Quezada, L. E., Barros-Castro, R. A., Gonzalez, M. A., & Palominos, P. I. (2017). A method for designing strategy maps using DEMATEL and linear programming. *Management Decision, 55*(8), 1802–1823. https://doi.org/10.1108/MD-08-2016-0597

Martins, V. F., de Almeida Souza Concilio, I., & de Paiva Guimarães, M. (2018). Problem based learning associated to the development of games for programming teaching. *Computer Applications in Engineering Education, 26*(5), 1577–1589. https://doi.org/10.1002/cae.21968

Mazaitis, D. (1993). The object-oriented paradigm in the undergraduate curriculum: a survey of implementations and issues. *ACM SIGCSE Bulletin, 25*(3), 58–64. https://doi.org/10.1145/165408.165432

Moons, J., & De Backer, C. (2009). Rationale Behind the Design of the EduVisor Software Visualization Component. *Electronic Notes in Theoretical Computer Science, 224*(C), 57–65. https://doi.org/10.1016/j.entcs.2008.12.049

Moons, J., & De Backer, C. (2013). The design and pilot evaluation of an interactive learning environment for introductory programming influenced by cognitive load theory and constructivism. *Computers & Education, 60*(1), 368–384. https://doi.org/10.1016/j.compedu.2012.08.009

Moussa, W. E., Almalki, R. M., Alamoudi, M. A., & Allinjawi, A. (2016). Proposing a 3d interactive visualization tool for learning oop concepts. *2016 13th Learning and Technology Conference (L&T)*, 1–7. https://doi.org/10.1109/LT.2016.7562861

Musil, M., & Richta, K. (2017). Contribution to Teaching Programming Based on "Object-First" Style at College of Polytechnics Jihlava. *Advances in Intelligent Systems and Computing*, *511 AISC*, 219–228. https://doi.org/10.1007/978-3-319-46535-7_17

Nelson, H. J., Irwin, G., & Monarchi, D. E. (1997). Journeys up the mountain: Different paths to learning object-oriented programming. *Accounting, Management and Information Technologies, 7*(2), 53–85. https://doi.org/10.1016/S0959-8022(96)00024-0

Olsson, M., & Mozelius, P. (2015). Visualization of concepts and algorithms in programming education - A design theoretic multimodal perspective. *Proceedings of the International Conference on E-Learning, ICEL*, *2015-Janua*(June), 257–264.

Pei, T., Shih, H., Zheng, W., Skelton, G., & Leggette, E. (2010). Integrating Self Regulating Learning With An Object Oriented Programming Course. *2010 Annual Conference & Exposition Proceedings*, 15.770.1-15.770.13. https://doi.org/10.18260/1-2--16444

Piteira, M., Costa, C. J., & Aparicio, M. (2017). CANOE e Fluxo: Determinantes na adoção de curso de programação online gamificado. *RISTI - Revista Ibérica de Sistemas e Tecnologias de Informação*, *25*(25), 34–53. https://doi.org/10.17013/risti.25.34-53

Popat, S., & Starkey, L. (2019). Learning to code or coding to learn? A systematic review. *Computers & Education, 128*(October 2018), 365–376. https://doi.org/10.1016/j.compedu.2018.10.005

Qian, Y., Hambrusch, S., Yadav, A., Gretter, S., & Li, Y. (2020). Teachers' Perceptions of Student Misconceptions in Introductory Programming. *Journal of Educational Computing Research, 58*(2), 364–397. https://doi.org/10.1177/0735633119845413

Qian, Y., & Lehman, J. (2017). Students' Misconceptions and Other Difficulties in Introductory Programming. *ACM Transactions on Computing Education, 18*(1), 1–24. https://doi.org/10.1145/3077618

Radenski, A. (2006). "Python First": A Lab-Based Digital Introduction to Computer Science. *Proceedings of the 11th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education - ITICSE '06*, 197–201. https://doi.org/10.1145/1140124.1140177

Rajashekharaiah, K. M. M., Pawar, M., Patil, M. S., Kulenavar, N., & Joshi, G. H. (2016). Design Thinking Framework to Enhance Object Oriented Design and Problem Analysis Skill in Java Programming Laboratory: An Experience. *2016 IEEE 4th International Conference on MOOCs, Innovation and Technology in Education (MITE)*, *February 2018*, 200–205. https://doi.org/10.1109/MITE.2016.048

Sajaniemi, J., Byckling, P., & Gerdt, P. (2007). Animation Metaphors for Object-Oriented Concepts. *Electronic Notes in Theoretical Computer Science, 178*(1), 15–22. https://doi.org/10.1016/j.entcs.2007.01.037

Sanders, K., Boustedt, J., Eckerdal, A., McCartney, R., Moström, J. E., Thomas, L., & Zander, C. (2008). Student understanding of object-oriented programming as expressed in concept maps. *Proceedings of the 39th SIGCSE Technical Symposium on Computer Science Education - SIGCSE '08*, 332–336. https://doi.org/10.1145/1352135.1352251

Sarkar, S. P., Sarker, B., & Hossain, S. K. A. (2016). Cross platform interactive programming learning environment for kids with edutainment and gamification. *2016 19th International Conference on Computer and Information Technology (ICCIT)*, 218–222. https://doi.org/10.1109/ICCITECHN.2016.7860198

Seng, W. Y., & Yatim, M. H. M. (2014). Computer Game as Learning and Teaching Tool for Object Oriented Programming in Higher Education Institution. *Procedia - Social and Behavioral Sciences, 123*, 215–224. https://doi.org/10.1016/j.sbspro.2014.01.1417

Sheetz, S. D. (2002). Identifying the difficulties of object-oriented development. *Journal of Systems and Software, 64*(1), 23–36. https://doi.org/10.1016/S0164-1212(02)00019-5

Sheetz, S. D., Irwin, G., Tegarden, D. P., Nelson, H. J., & Monarchi, D. E. (1997). Exploring the Difficulties of Learning Object-Oriented Techniques. *Journal of Management Information Systems, 14*(2), 103–131. https://doi.org/10.1080/07421222.1997.11518167

Sien, V. Y. (2011). Implementation of the Concept-Driven Approach in an Object-Oriented Analysis and Design Course. In *2013 12th International Conference on Environment and Electrical Engineering: Vol. 6627 LNCS* (pp. 55–69). IEEE. https://doi.org/10.1007/978-3-642-21210-9_6

Sien, V. Y., & Chong, D. W. K. (2011). Threshold concepts in object-oriented modelling. *Electronic Communications of the EASST*, *52*, 1–11. https://doi.org/10.14279/tuj.eceasst.52.763

Streib, J. T., & Soma, T. (2010). Using contour diagrams and JIVE to illustrate object-oriented semantics in the Java programming language. *Proceedings of the 41st ACM Technical Symposium on Computer Science Education - SIGCSE '10*, 510–514. https://doi.org/10.1145/1734263.1734435

Sun, R., Zhang, B., & Liu, T. (2016). Ranking web service for high quality by applying improved Entropy-TOPSIS method. *2016 17th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD)*, 249–254. https://doi.org/10.1109/SNPD.2016.7515909

Tahat, K. (2014). An Innovative Instructional Method for Teaching Object-Oriented Modelling. *International Arab Journal of Information Technology, 11*(November), 540–549.

Tan, J., Guo, X., Zheng, W., & Zhong, M. (2014). Case-based teaching using the Laboratory Animal System for learning C/C++ programming. *Computers & Education, 77*, 39–49. https://doi.org/10.1016/j.compedu.2014.04.003

Tegarden, D. P., & Sheetz, S. D. (2001). Cognitive activities in OO development. *International Journal of Human-Computer Studies, 54*(6), 779–798. https://doi.org/10.1006/ijhc.1999.0462

Thomasson, B. J., Ratcliffe, M. B., & Thomas, L. A. (2006). Improving the tutoring of software design using case-based reasoning. *Advanced Engineering Informatics, 20*(4), 351–362. https://doi.org/10.1016/j.aei.2006.07.002

Turner, S., Pérez-Quiñones, M. A., Edwards, S., & Chase, J. (2010). Peer Review in CS2: Conceptual Learning. *Proceedings of the 41st ACM Technical Symposium on Computer Science Education - SIGCSE '10*, 331. https://doi.org/10.1145/1734263.1734379

Van Eck, N. J., & Waltman, L. (2010). Software survey: VOSviewer, a computer program for bibliometric mapping. *Scientometrics, 84*(2), 523–538. https://doi.org/10.1007/s11192-009-0146-3

Xinogalos, S. (2015). Object-Oriented Design and Programming. *ACM Transactions on Computing Education, 15*(3), 1–21. https://doi.org/10.1145/2700519

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*(May), 147–163. https://doi.org/10.1016/j.jss.2018.07.037

Yang, T.-C., Chen, S. Y., & Hwang, G.-J. (2015). The influences of a two-tier test strategy on student learning: A lag sequential analysis approach. *Computers & Education, 82*(1), 366–377. https://doi.org/10.1016/j.compedu.2014.11.021

Yi, T., & Fang, C. (2018). A Novel Method of Complexity Metric for Object-Oriented Software. *International Journal of Digital Multimedia Broadcasting, 2018*(6), 1–9. https://doi.org/10.1155/2018/7624768

Zainal, N. F. A., Shahrani, S., Yatim, N. F. M., Rahman, R. A., Rahmat, M., & Latih, R. (2012). Students' Perception and Motivation Towards Programming. *Procedia - Social and Behavioral Sciences, 59*, 277–286. https://doi.org/10.1016/j.sbspro.2012.09.276

Zhang, J., Caldwell, E. R., & Smith, E. (2013). Learning the concept of Java inheritance in a game. *Proceedings of CGAMES'2013 USA*, 212–216. https://doi.org/10.1109/CGames.2013.6632635

Zhang, X., Crabtree, J. D., Terwilliger, M. G., & Redman, T. T. (2018). Assessing Students' Object-Oriented Programming Skills with Java: The "Department-Employee" Project. *Journal of Computer Information Systems, 60*(3), 1–13. https://doi.org/10.1080/08874417.2018.1467243

Zhang, Y., Zhang, F., Zhu, H., & Guo, P. (2019). An Optimization-Evaluation Agricultural Water Planning Approach Based on Interval Linear Fractional Bi-Level Programming and IAHP-TOPSIS. *Water, 11*(5), 1–22. https://doi.org/10.3390/w11051094

Authors and Affiliations

Luz E. Gutiérrez[1] · **Carlos A. Guerrero**[2] · **Héctor A. López-Ospina**[3]

Carlos A. Guerrero
carlos.guerrero@usantoto.edu.co

Héctor A. López-Ospina
hhlopez@uninorte.edu.co

[1] Department of System Engineering, Faculty of Engineering, Universidad del Norte, Barranquilla, Colombia

[2] Faculty of System Engineering, Universidad Santo Tomás, Tunja, Colombia

[3] Department of Industrial Engineering, Faculty of Engineering, Universidad del Norte, Barranquilla, Colombia