IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received January 25, 2022, accepted March 1, 2022, date of publication March 8, 2022, date of current version March 18, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3157290

Identifying Difficulties of Software Modeling Through Class Diagrams: A Long-Term Comparative Analysis

PAMELA FLORES, MAYRA ALVAREZ, AND JENNY TORRES
Department of Informatics and Computer Science, Escuela Politécnica Nacional, Quito 170517, Ecuador

Corresponding author: Pamela Flores (pamela.flores@epn.edu.ec)

This work was supported by the Escuela Politécnica Nacional del Ecuador through the research project under Grant PIS-19-11.

ABSTRACT Software modeling is a creative activity in which software components and their relationships are identified based on customer requirements. Based on the literature, object-oriented software modeling is based on four fundamental pillars which are abstraction, encapsulation, decomposition, and inheritance. However, despite the existence of guidelines and recommendations for implementing the object-oriented approach, novice software designers do not make good design decisions, leading to inefficient designs that cannot be modifiable, understandable, or user-friendly distribute at the development level. The literature reveals that the most common difficulties faced by software designers is a lack of understanding and confusion of concepts related to the object-oriented approach, as well as difficulties in creating Unified Modeling Language diagrams, especially class diagrams. The work presented in this article uses a qualitative and quantitative approach to determine, in a group of university students, what are the most recurrent difficulties and their persistence during the time. The qualitative case study is the method that allowed to generate the documents: *diagnostic* and *evaluation* tests. Additionally, a thematic analysis was used to identify, analyze and report patterns within the data. In order to know the occurrences of the problems in the case study, as part of our quantitative approach, a comparative study was applied to compare the results obtained between the diagnostic and evaluation tests and thus establish the similarities and differences among the cases observed, through the hierarchical clustering technique. The findings of this study show us 16 difficulties identified after the qualitative analysis, while the quantitative analysis shows us the number of occurrences and their persistence over time. The difficulties reported in both analyzes focus on these three difficulties: a) Definition of attributes that could be a class, b) Classes with inadequate or insufficient behavior and, c) Incorrect use of multiplicity between classes. Each of these difficulties is analyzed in depth in this study.

INDEX TERMS Computer science, computer science education, object oriented modeling, software design, software engineering, systems engineering education.

I. INTRODUCTION

Software engineering (SE) is a branch of computer science that studies the creation of reliable and quality software based, on engineering methods and techniques [1]. In other words, the SE is the practical application of scientific knowledge to the design and construction of programmes and to the associated documentation required to develop, operate and maintain these programs [2]. SE processes generally consist of five structural activities: Requirements Definition, System and Software Design, Unit Implementation and Testing, System Integration and Testing; and Operation and Maintenance [3].

Software design is a creative activity where the components of the software and their relationships are identified, based on customer requirements. Software design, particularly the object-oriented approach, is based on four principles: abstraction, encapsulation, decomposition and inheritance [4], [5]. This approach is one of the most used to represent the problem domain and involves the design of classes and the relationships between those classes. Such classes define both the objects in the system and their

The associate editor coordinating the review of this manuscript and approving it for publication was Fabrizio Messina.

interactions [3]; the standardized language to express such a scenario is through Unified Language Modeling (UML).

Software design is the only way to accurately materialize customer requirements, and actually there are guidelines and recommendations [5], [6] for object-oriented software design that designers, generally beginners, fail to implement in practice. As a result, designers have great difficulty in finding good decisions, which leads to the violation of the principles of the object-oriented approach, which results in low-quality designs [7]–[9]. In the literature we find that the most common difficulties faced by software designers are:

- Lack of understanding of object-oriented concepts [10]–[13].
- Confusion between concepts related to object/class, class/collection and concepts involving modeling [9].
- Confusion in the association of class/object, attribute/method and class/subclass related to inheritance concepts [14].
- Difficulties in creating UML diagrams, syntax and notation errors; errors related to attributes, association and classes [14], [15].
- Difficulty in organizing the information in the diagrams and in using correctly the generalization-specialization type association [16].
- Incorrect mapping of the problem concept, misuse of inheritance, relationships and function names [17].
- Lack of creation of classes for relevant aspects of the application [10].

The work presented in this article uses a qualitative and quantitative approach to determine in a group of university students, what are the most recurrent difficulties and their persistence during the time. The qualitative case study is the method that allowed to generate the documents: diagnostic and evaluation tests. Additionally, a thematic analysis was used to identify, analyze and report patterns within the data. In order to know the occurrences of the problems in the students of the case study, as part of our quantitative approach, a comparative study was applied to compare the results obtained between the diagnostic and evaluation tests and thus establish the similarities and differences among the cases observed, through the hierarchical clustering technique.

The findings of this study show us 16 difficulties identified after the qualitative analysis, while the quantitative analysis shows us the number of occurrences and their persistence over time. The difficulties reported in both analyzes focus on these three difficulties: a) Definition of attributes that could be a class, b) Classes with inadequate or insufficient behavior and, c) Incorrect use of multiplicity between classes; each of these is analyzed in depth in this study.

The results of this study are very helpful when identifying the importance of software design and the most common problems that arise from the object-oriented approach. However, few references study the persistence of difficulties and misconceptions of students. Our study focuses on discovering perceptions of students in the long term, when designing software, based on exercises that make it possible to analyze perceptions of students of object-oriented approach concepts and their difficulties. The results will allow the reflection of those who study the object-oriented approach and its influence on educators who focus on teaching object-oriented software design.

The rest of the article is structured as follows. Section II shows several related works found in the literature and the contribution that our work has in this regard. Section III presents the research methodology. Section IV shows the development of the qualitative study. Section V displays the development of the quantitative study. Section VI presents the results of the study and Section VII presents a discussion explaining the advices that emerge from it and the threats of reliability. Finally, Section VIII dconcludes the paper.

II. RELATED WORK

In the literature we have found works related to how students implement software design concepts under the object-oriented approach, considering both qualitative and quantitative perspectives. Several of these works show the difficulties of students in software design and programming.

Some works in the state of the art study how students conceive fundamental concepts of object-classes, modeling and object-oriented programming (OOP) through longitudinal studies with a mixed approach. The researcher Xinogalos, based on a literature review and taking as a model the studies of Eckerdal and Thune [18], [19] compares conceptions about objects and classes from the literature with data from written exams that include open-ended questions applied to a group of students. In this study, the authors identify that the most referenced misconceptions are the confusion between object/class, class/collection and concepts involving modeling, the latter because students perceive class as an abstraction of some type of object/entity in the domain of the problem [9]. Through the exploration of mental models and evaluation of tests taken by students, the authors identify that the most problematic areas are related to the association between class/object, attribute/method and class/subclass including inheritance concepts, in addition to syntactic errors [14].

There are a couple of studies [20], [21], where the authors state that there is a relationship between the performance of students obtained in tests related to software design questions and produce a more reliable program code. In [20], the author studies the relationship between the ability to create a design with the ability to program from the analysis of UML diagrams versus source code. Based on the review of the literature and the analysis of the work presented by the students, the author expresses that the students have difficulties to close the gap between the problem descriptions and the code, however, the author is not explicit in defining which are the problem student face when transferring diagrams to code. In [21], through a planned quantitative method and correlation analysis, the authors conclude that students improved their programming performance thanks to their previous designs. In addition, in the same study, the researches based on test results and object-oriented metrics parameters

conduct an empirical study using the hierarchical clustering technique to compare design quality of students and their performance in terms of correctness. For this purpose, they analyze the diagrams made by the students at the beginning of the course and the corrections they do to those diagrams at the end of the course.

Several researchers conceive the use of UML notation and modeling skills as tools for success in software design, and they study the difficulties presented by students in creating UML diagrams. In [15] the authors conduct a study to determine the typical mistakes students make when creating class diagrams, through error analysis and quantitative approach. The authors consider the results of the final tests performed by the students, consisting of the creation of diagrams, to categorize four errors: a) syntactic errors including notation errors, b) attribute-related errors, c) association-related error and, d) class-related errors. In [17] the authors rely on a literature review and their experience as teachers by suggesting that for many students the visualization of objects is not always obvious and that common errors in modeling include: incorrect mapping of the concept of the problem, misuse of inheritance, misuse of relationships and misuse of function names. The researchers propose a problem-based interactive exercise, consisting of a game called Chitty-Chitty Bang-Bang (CCBB) for a better understanding of object-oriented concepts, obtaining beneficial results in average students and, particularly in below-average students, but little effect on brighter students. In [22] the authors study the difficulties that students face while learning to model UML diagrams, such as: difficulty in understanding the syntax and semantics of the diagram, difficulty in organizing information in diagrams, difficulty in correctly using the generalization-specialization type association. In order to overcome these difficulties, the authors propose to explore pedagogical methods such as: problem-based learning (PBL) and learning from erroneous examples (ErrEx), so that students are actively involved in the learning process. These three papers found that errors of syntax and semantics, inheritance and association, are the most common among beginners.

The previous research to this study [10], focuses on the exploration of object-oriented software design decisions that students make and their possible causes, through a qualitative study. Through a thematic analysis applied to diagnostic and evaluation tests, presented by students, which consist of developing 3 exercises that involve concepts of the object-oriented approach. Researchers identify that the most common errors are related with creating/not creating classes and objects, class behavior, class names, and confusion between subclass/superclass. This reflects a lack of abstraction in students, manifesting an excessive simplification of problems. As to the possible causes, the authors mention: strict copy of reality, influence of the structured approach, simplistic general description and lack of understanding of concepts of the object-oriented approach.

The contribution of this work lies in studying the difficulties that students face in the design of software under the object-oriented approach and in addition the persistence of these difficulties in the long term after academic instruction. The results of this study will allow the reflection of those who study the object-oriented approach and its influence on the learning of object-oriented software design.

III. RESEARCH METHODOLOGY

This section shows the research questions, the chosen methodology, and the details of the selected case study.

A. RESEARCH QUESTIONS

This research was conducted through two research questions:

- *¿What are the difficulties that students present when designing using class diagrams?*
- *¿From the difficulties previously found, what are the most frequent difficulties that students present when designing using class diagrams during the academic period?*

This study requires a qualitative and quantitative data analysis perspective. On the one hand, qualitative research includes all non-numerical data. According to [23], qualitative research consists of extracting descriptions from observations taken from interviews, narrations, recordings, audio transcripts, written records of all kinds, among others. These data are generated by case studies, action-research and ethnography. In cite [24], the objective of qualitative research is understanding and focuses on the investigation of facts, as well as the interpretation of events.

In particular, a qualitative case study is defined as a holistic and intensive description and analysis of a single instance, phenomenon or social unit [25]. In our work the qualitative case study, is the method that allowed to generate the documents, which consist of the evaluation tests. In addition, a thematic analysis was used to identify, analyze and report patterns within the data [26].

On the other hand, quantitative research includes numerical data that try to determine the association or correlation between variables, the generalization and objectification of the results. After the study of the association or correlation, it aims, in turn, to make causal inference that explains why things happen or not, in a certain way [27].

In order to know the occurrences of the problems in case studies, as part of our quantitative approach, a comparative study was applied, to compare the results obtained in the diagnostic test applied to the students at the beginning of the academic period and the evaluation test applied at the end of the course. The comparison aims to understand unknown things from known, with the possibility of explaining and interpreting them; and also aims to systematize the information distinguishing the differences with similar cases [28].

B. RESEARCH METHOD

In this study, we have applied an instance of qualitative case study research called *thematic analysis*, which is defined as a research method that allow to identify, analyze, organize,

describe and report topics that are within a set of data. The thematic analysis allowed the identification, analysis and reporting patterns within the data [22]. In addition, to know the occurrence of problems in students, it was necessary to make use of a quantitative analysis. Most quantitative data analysis involves a process of abstraction that starts from the research of qualitative data that have been previously analyzed and that is important for the research topic [29].

1) SETTING

The case study was conducted with a group of students from a University of the Faculty of Informatics. Details are shown below:

2) SUBJECT

The subject is *Modeling and Software Design*, required subject of the fifth semester in the Faculty of Informatics, this subject is taught for 4 hours a week for 16 weeks. Students taking this subject must have previously studied the subjects of Database Management, Programming I and Programming II; and have as a co-requirement the subject of Software Engineering I. The content of Programming II focuses on the teaching of object-oriented programming languages, which means that students already have prior knowledge about the concepts of this approach at the programming level.

3) LECTURE STRUCTURE

The content of the subject during the 16 weeks of the semester is shown below:

- Software Design and Process Models
- Software Development Paradigms
- Classes Vs. Objects
- Coupling and Cohesion
- Inheritance
- UML Diagrams
- Decomposition in Software Design
- Abstraction in Software Design
- Principle of Concealment of Information
- Observer Pattern
- Façade and MVC Patterns
- Decorator and Factory Patterns
- Status Pattern
- Chain of Responsibility Pattern

4) PARTICIPANTS

The group of students in this subject initially consisted of 26 students. However, the evaluated results were 22 since four students were discarded for the study, three of the discarded students repeated the subject and another student left the course, therefore, in the comparative statistics was not taken into account.

5) TEST

The *diagnostic test* was performed in the second week after the UML Diagram topic and it was composed of three exercises. The exercises presented to the students for this case study were chosen because they allow to apply the concepts learned in UML topic. While the *evaluation test* was presented at the end of the academic period. The first exercise called ''Betting'' involves the use of inheritance and the decomposition of the problem. The second exercise called ''Circle'' is related to graphic objects that allows to know the way in which the students conceive the problem and finally the exercise called ''Hotel'' is a transactional exercise whose characteristics allow us to know the understanding of the objects in an exercise that could be solved in a structured way. The statement of each exercise is shown:

- Exercise Betting
  It is required to make an application for the sports betting service, where a user must register for the bets. Bets can be received by transfer or by card. The system supports different types of bets, for example: Single bet (which team wins), special bet (which minute scores the first goal) and others.
- Exercise Circle
  This application consists on drawing a small circle inside a larger circle. The smaller circle can move inside the larger circle, without getting out of it.
- Exercise Hotel
  This application is responsible for booking rooms in a hotel. It is necessary to take into account the booking dates and the verification of room availability.

IV. QUALITATIVE CASE STUDY

The qualitative approach bases the analysis on the result of the software design exercises performed by the students. The thematic analysis process was based on the model proposed by Seidel [30], which consists of the following stages:

- Data Collection
- Data Encoding
- Data Refinement
- Grouping of Qualitative Data

A. DATA COLLECTION

Two sets of qualitative data (diagnostic and evaluation test) were collected for this study. The data collected were obtained based on the 3 design exercises already described in Subsection III-B5 and an interview was applied to the students who were part of a diagnostic test carried out on 26 students at the beginning of the academic period, this corresponds to the first data set published in a previous work (accepted for publication).

At the end of the academic period an evaluation test was applied with the same exercises, obtaining the second set of data. In both the diagnostic and evaluation tests, the design of each statement was requested through class diagrams.

Prior to the diagnostic test, students received instruction on concepts related to Language Unified Modeling (UML), so that all students have a standard language to design the exercises. In this instruction only sequence diagrams

and class diagrams with their respective relationships were considered, and special attention was paid to the semantic structure of inheritance. Finally, the concepts related to the object-oriented approach such as object, class and message were also updated. In addition, it is important to emphasize that students have already received subjects where they have previously programmed in object-oriented languages, this means that the concepts taught in class were a reinforcement to their knowledge of object-oriented concepts.

B. DATA ENCODING

After obtaining the results of the diagnostic and evaluation tests conducted with the students, we proceeded to assign codes to the collected documents. *"A code in qualitative research is most often defined as a word or short phrase that symbolically assigns a summative, salient, essence-capturing, and/or evocative attribute to a portion of visual or language-based data"* [31].

At this stage a deductive coding was carried out and the test were coded using Atlas.ti software [32]. For this, we have taken as a reference the problems that were previously identified in a work published by the same authors of this study (accepted for publication), however, it is important to mention that some of the findings shown in this study have also been reported by other authors [33]–[36]. In the previous study ten student's design decisions were identified, which are shown below:

1) Tendency to create a third class between two other classes to associate them, instead of creating a many-many association between them.
2) Assigning the behavior of a real-life object to the class diagram as is, instead of using an abstraction of that concept at the software level.
3) Lack of creation of classes for relevant aspects of the application.
4) Designing classes with different names, but with the same structure.
5) Designing classes without any behavior. Special interest in defining classes only through their attributes.
6) Place responsibilities on classes that should not be responsible for that behavior.
7) Creating classes that differ from their superclass or sibling classes only by its attribute values, when the behavior should be the same.
8) Assignment of complex behaviors as attributes.
9) Belief of students that placing an ID attribute in each class will allow them to access all instances of that class.
10) Definition of classes that are not concepts.

It is important to mention that in this study we did not limit ourselves to coding only the problems mentioned above. The experts were free to code problems that were not taken into account in the previous study. Consequently, in the present research we found 8 of the 10 problems previously identified and 9 additional problems, so that the 16 problems found are explained later. At this stage a total of 365 codes were obtained in the diagnostic test and 420 codes in the evaluation test. The coding was performed separately by two experts, and then applied the *peer evaluation technique*, in order to guarantee the coding process.

TABLE 1. Example of codebook by exercise.

Exercise	Code	SuperCode	Prefix
Betting	The student creates an intermediate class between the User class and the Bet class called BetDetail class in order to "combine" the bets made by the user.	Creating intermediate classes.	REL
Circle	The student creates a Circle1 class and a Circle2 class to represent two circles, when he could have made a Circle class that can generate two instances.	Confusion about the concept of class and object.	INS
Hotel	The student does not contemplate in the class diagram the possibility of making more than one reservation.	Incorrect use of multiplicity.	LIS

An example of the codes established in the codebook is shown in the Table 1.

C. DATA REFINEMENT

This is the stage where the preliminary codes obtained in the previous stage were analyzed, matching similar codes or separating them. This stage was carried out in conjunction with the researchers involved in the coding in order to maintain the integrity of the coding.

D. GROUPING OF QUALITATIVE DATA

Finally, the codes generated in the Refinement stage were grouped in section IV-C. The grouping was performed following the criteria of the research questions. These codes were then abstracted into categories through a thematic analysis. The details of the resulting categories and their respective acronyms are shown below:

1) Convert attributes into classes (CLA), refers to extract an attribute and convert it into a class. However, this created a class that represents only data, without behavior. For example when the student designs a `Circle` class and a `Position` class, but the latter has only attributes `Xo` and `Yo`.
2) Not considering the problem from an holistic perspective (HOL). This category is related to the fact that students do not conceive all the aspects necessary to solve the problem. For example, in the exercise *Betting*, the student should check all aspects that influence the resolution of the exercise, such as whether there are sufficient funds for a person to place a bet.
3) Not including the classes necessary for the design (NUM), refers to omitting classes in the diagram in

spite of being explicitly mentioned in the statement, for example, the absence of the `Account` class in the exercise *Betting*.

4) Creation of classes that should be related to a concept (FUN), but the concept itself does not exist in the diagram. For example, when a student has created a class named `BetType`, but the concept `Bet` does not exist in the diagram.
5) Incorrect use of multiplicity between classes (LIS) because the student does not identify the possible existence of several instances of the same class. For example, in the exercise *Hotel*, some students did not identify that multiple reservations can be made to the same room.
6) Classes with inadequate or insufficient behavior (COM), this category refers to those classes that were created with a behavior foreign to the concept of the class or the behavior only partially represents the concept, for example, the creation of a `Hotel` class, which has `reserveRoom()` method.
7) Creating the same class multiple times on a single class diagram (INS), instead of instancing the class multiple times. For example, the creation of different rooms, such as `SingleRoom`, `DoubleRoom`, `TripleRoom`, instead of just the class `Room`.
8) Defining attributes that could be a class (ATR), an example of this is when the student creates a `Reserve` class, and places an attribute `roomNumber`, and there is no `Room` class in the design.
9) Placement of different methods that could have been represented by a single method (MET), for example, when it is placed `move-left`, and `move-right` instead of `move`.
10) Classes built in the image and likeness of concepts of the real life (REA), for example, in the case of a student who created a `Room` class with a `toClean()` method.
11) Creating a `Main` class, which only function should be to trigger the start of the program, and filling it with functions that should be part of other classes (MAI).
12) Creation of classes whose name and behavior represent an action and not a concept (ACC), an example was presented in the exercise *Circle*, where a student diagrammed a class labeled `SmallCircle`, and additionally two classes: one labeled `Draw` and one `Move`.
13) Creating relationships between confusing or erroneous classes (PCL), it refers to syntax errors in UML semantics. For example, using the composition relationship instead of aggregation.
14) Construction of classes with attributes but no methods, even when they needed methods with distinct behaviors in the context of the exercise (SIC). For example, the `Client` class with the attributes `name`, `lastName`, `id`, and without any methods.
15) Construction of inheritance structures whose derived classes only differ from the base class by their attributes (HER). For example, when students created a base class called `Bet` with an implemented `bet()` method, and two subclasses, one called `IndividualBet` with an attribute called `individualFactor` and the other `ComplexBet` with an attribute called `complexFactor` but both with the same implemented method inherited from the base class called `Bet`
16) Similarity to the entity-relationship model (REL), refers to the fact that the learner tends to create an intermediate class for recording details of related classes.

V. QUANTITATIVE ANALYSIS

The quantitative approach allows the search for patterns in the obtained data. Therefore, comparison is essential in any field of research as it allows the establishment of systematic similarities and differences among observed cases, as well as the possible development and testing of hypotheses and theories about their causal relationships [37]. A definition for the term comparison is *"the act of observing two or more things in order to discover their relationships or estimate their differences and similarities"* [38].

A. COMPARATIVE ANALYSIS

In a comparative analysis, a distinction is made between *Most Similar System Design (MSSD)* and *Most Different System Design (MDSD)*. When applying MSSD the research objects are chosen as similar as possible, except for the phenomenon whose effects we are interested in evaluating. While in MDSD the strategy is to choose research units that are as different as possible, the basic logic is that differences cannot explain similarities [39].

Comparative analysis involves several techniques including: case study, statistical analysis and experimental research. In addition, it involves a focus on the analysis of a limited number of cases. The outcome is focused on obtaining data that leads to the definition of a problem or to the improvement of knowledge about it [37].

According to [40], the comparative analysis presents two strategies: case studies and study of variables, which are defined below:

Case study: A small number of cases are defined and experimental rigor is sought through the identification of comparable effects of a phenomenon and the analysis of differences and similarities between them.

Variables study: It aims to formulate broad generalizations about the objects to be studied and to test abstract hypotheses derived from theories applicable to the cases of study.

The efficiency of the different methods available for conducting comparative research will depend on their effectiveness in solving the problem of causal complexity analysis.

In this research the *Hierarchical Agglomerative Clustering (HAC)* method will be implemented. The main idea is to group similar data points in one group and separate the different observations into other groups, calculating the distance between them. The hierarchical grouping is represented by

dendograms that allow a clear analysis of similarities and differences between the individuals in the case study, facilitating the monitoring of the persistence of difficulties and misconceptions of the students in object-oriented software design.

1) VARIABLES OF STUDY

In this study and based on the results of the case studies, it was determined that the comparative entities are:

- Results of the diagnostic tests, performed on students at the beginning of the academic instruction.
- Results of the evaluation tests, conducted at the end of the academic instruction.

2) DEFINITION OF VARIABLES

For this research, two case studies were defined, represented by the results of the diagnostic test (Case 1) and the evaluation test (Case 2) of a group of 22 students, which is equivalent to the dependent variables.

On the other hand, the independent variables are represented by the categories of difficulties found during the qualitative study and which were classified as: CLA, HOL, NUM, FUN, LIS, COM, INS, ATR, MET, REL, REA, MAI, ACC, PCL, SIC and HER described in the section IV-D.

Once the comparable entities and the variables to be studied have been defined, the comparative strategy allows to pay attention to those aspects that are very different (the most differences) and those aspects that are very similar (the most similarities).

3) VARIABLE DICHOTOMIZATION

Cluster analysis is extremely important in scientific research, in any branch of knowledge. Bearing in mind that classification is one of the fundamental objectives of science and to the extent that cluster analysis provides us with the technical means to carry it out, it will be essential in any investigation.

Therefore, once the cases and variables to be studied have been defined, the HAC method is used to categorize the students who took the diagnostic and evaluation tests into groups. The groupings made were based on the type and number of occurrences of problems found in the students of the case study on object-oriented software design.

The cluster analysis technique aims to sort individuals into groups, so that the individuals in the group are as similar as possible to each other and as diverse as possible between elements of other groups. Hierarchical clustering groups data based on the distance between each of the individuals in the group. This technique tries to achieve successive groupings among individuals so that they are progressively integrated into clusters which, in turn, would be joined together at a higher level forming larger groups that will later be joined together to reach the final cluster containing all the cases analyzed. However, it is not appropriate for very large data set [41].

Based on the agglomerative hierarchical method, groups were generated in each of the phases of the process looking for the number of clusters for an optimal grouping. At the beginning, each individual is separated. At each step, the closest individuals are merged to form different clusters. That is, each observation is assigned to its own cluster. Then, the similarity or distance between each of the clusters is calculated and the two most similar clusters are merged into one [42].

4) HIERARCHICAL AGGLOMERATIVE CLUSTERING (HAC)

The HAC algorithm aims to classify individuals. It is fundamentally about solving the following problem: Given a set of individuals of N elements characterized by the information of n variables $Xj, (j = 1, 2, \ldots, n)$. We set ourselves the challenge of being able to classify them in such a way that the individuals belonging to a group are as similar to each other as possible, the different groups being as dissimilar as possible among them.

With cluster analysis, we search a set of groups with different individuals assigned by some criterion of homogeneity, in our study the criterion is given by the type and number of occurrences of problems found in students in the design of object oriented software. Additionally, the possibility of reassignments should be considered throughout the process, establish criteria to stop and/or perform the grouping, and define a measure of similarity or divergence to classify individuals into groups.

The Euclidean distance is the best known and easiest to understand dissimilarity, since its definition coincides with the most common concept of distance (space between two points). The Euclidean distance is recommended when the variables are homogeneous and measured in similar units and/or when the variance matrix is unknown.

Given a set of N elements to be grouped and NxN distance matrix, the basic process of Johnson's hierarchical clustering [43] can be structured according to the following scheme:

- Step 1: Assign each element to a cluster, so that having N elements, we will obtain N clusters. The distances/similarities among the clusters should be equal to the distances between the elements they contain.
- Step 2: Find the closest/similar pair of clusters and combine them into a single cluster, so that we get less clusters in each phase.
- Step 3: Calculate the distances between the new cluster and each of the old clusters.
- Step 4: Update the matrix to specify the distance between the different clusters that are formed as a result of the merger.

Steps 2, 3 and 4 can be repeated according to the researcher's criteria. These steps consist of searching for similarities between clusters. Therefore, it is required to determine a distance measure between each data point. For this purpose, we use the Euclidean distance function using (1). The similarity between individuals is plotted using dendrograms.

$$\|a - b\|_2 = \sqrt{\sum_i (a_i - b_i)^2}. \tag{1}$$

	E1D	E2D	E3D	E4D	E5D	E6D	E7D	E8D	E9D	E10D	E11D	E12D	E13D	E14D	E15D	E16D	E17D	E18D	E19D	E20D	E21D	E22D	E1E	E2E	E3E	E4E	E5E	E6E	E7E	E8E	E9E	E10E	E11E	E12E	E13E	E14E	E15E	E16E	E17E	E18E	E19E	E20E	E21E	E22E
E1D	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	6	4	3	3	1	5	2	1	3	2	6	5	2	1	4	4	4	4	5	5	3	3
E2D	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	10	0	1	1	3	1	2	3	1	2	2	1	2	3	0	0	0	0	1	1	1	1
E3D	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	3	7	6	6	4	8	5	4	6	5	9	8	5	4	7	7	7	7	8	8	6	6
E4D	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	6	4	3	3	1	5	2	1	3	2	6	5	2	1	4	4	4	4	5	5	3	3
E5D	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	9	1	0	0	2	2	1	2	0	1	3	2	1	2	1	1	1	1	2	2	0	0
E6D	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	5	5	4	4	2	6	3	2	4	3	7	6	3	2	5	5	5	5	6	6	4	4
E7D	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	5	5	4	4	2	6	3	2	4	3	7	6	3	2	5	5	5	5	6	6	4	4
E8D	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	5	5	4	4	2	6	3	2	4	3	7	6	3	2	5	5	5	5	6	6	4	4
E9D	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	5	5	4	4	2	6	3	2	4	3	7	6	3	2	5	5	5	5	6	6	4	4
E10D	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	5	5	4	4	2	6	3	2	4	3	7	6	3	2	5	5	5	5	6	6	4	4
E11D	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	5	5	4	4	2	6	3	2	4	3	7	6	3	2	5	5	5	5	6	6	4	4
E12D	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	5	5	4	4	2	6	3	2	4	3	7	6	3	2	5	5	5	5	6	6	4	4
E13D	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	5	5	4	4	2	6	3	2	4	3	7	6	3	2	5	5	5	5	6	6	4	4
E14D	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	5	5	4	4	2	6	3	2	4	3	7	6	3	2	5	5	5	5	6	6	4	4
E15D	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	5	5	4	4	2	6	3	2	4	3	7	6	3	2	5	5	5	5	6	6	4	4
E16D	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	5	5	4	4	2	6	3	2	4	3	7	6	3	2	5	5	5	5	6	6	4	4
E17D	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	5	5	4	4	2	6	3	2	4	3	7	6	3	2	5	5	5	5	6	6	4	4
E18D	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	5	5	4	4	2	6	3	2	4	3	7	6	3	2	5	5	5	5	6	6	4	4
E19D	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	5	5	4	4	2	6	3	2	4	3	7	6	3	2	5	5	5	5	6	6	4	4
E20D	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	5	5	4	4	2	6	3	2	4	3	7	6	3	2	5	5	5	5	6	6	4	4
E21D	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	5	5	4	4	2	6	3	2	4	3	7	6	3	2	5	5	5	5	6	6	4	4
E22D	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	5	5	4	4	2	6	3	2	4	3	7	6	3	2	5	5	5	5	6	6	4	4
E1E	6	10	3	6	9	5	5	9	4	7	8	7	4	9	7	7	6	6	5	0	6	6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
E2E	4	0	7	4	1	5	5	1	6	3	2	3	6	1	3	3	4	4	5	10	4	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
E3E	3	1	6	3	0	4	4	0	5	2	1	2	5	0	2	2	3	3	4	9	3	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
E4E	3	1	6	3	0	4	4	0	5	2	1	2	5	0	2	2	3	3	4	9	3	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
E5E	1	3	4	1	2	2	2	2	3	0	1	0	3	2	0	0	1	1	2	7	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
E6E	5	1	8	5	2	6	6	2	7	4	3	4	7	2	4	4	5	5	6	11	5	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
E7E	2	2	5	2	1	3	3	1	4	1	0	1	4	1	1	1	2	2	3	8	2	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
E8E	1	3	4	1	2	2	2	2	3	0	1	0	3	2	0	0	1	1	2	7	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
E9E	3	1	6	3	0	4	4	0	5	2	1	2	5	0	2	2	3	3	4	9	3	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
E10E	2	2	5	2	1	3	3	1	4	1	0	1	4	1	1	1	2	2	3	8	2	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
E11E	6	2	9	6	3	7	7	3	8	5	4	5	8	3	5	5	6	6	7	12	6	6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
E12E	5	1	8	5	2	6	6	2	7	4	3	4	7	2	4	4	5	5	6	11	5	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
E13E	2	2	5	2	1	3	3	1	4	1	0	1	4	1	1	1	2	2	3	8	2	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
E14E	1	3	4	1	2	2	2	2	3	0	1	0	3	2	0	0	1	1	2	7	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
E15E	4	0	7	4	1	5	5	1	6	3	2	3	6	1	3	3	4	4	5	10	4	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
E16E	4	0	7	4	1	5	5	1	6	3	2	3	6	1	3	3	4	4	5	10	4	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
E17E	4	0	7	4	1	5	5	1	6	3	2	3	6	1	3	3	4	4	5	10	4	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
E18E	4	0	7	4	1	5	5	1	6	3	2	3	6	1	3	3	4	4	5	10	4	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
E19E	5	1	8	5	2	6	6	2	7	4	3	4	7	2	4	4	5	5	6	11	5	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
E20E	5	1	8	5	2	6	6	2	7	4	3	4	7	2	4	4	5	5	6	11	5	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
E21E	3	1	6	3	0	4	4	0	5	2	1	2	5	0	2	2	3	3	4	9	3	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
E22E	3	1	6	3	0	4	4	0	5	2	1	2	5	0	2	2	3	3	4	9	3	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FIGURE 1. 4-quadrant matrix from which case studies will be classified into different clusters.

Step 1: The initial matrix was obtained based on the study cases formed by a group of diagnostic tests and an evaluation test group. The equation (1), resulting in a matrix of 4 quadrants, was applied to these data.

In Figure 1, the data from matrix are shown from right to left and corresponding to the relationship between diagnostic tests vs diagnosis, diagnosis vs evaluation, evaluation vs diagnosis and evaluation vs evaluation. The grey boxes represent the clusters formed, the blue boxes are clusters with a single set of results. The yellow boxes are the cases of students who were part of the cluster and the red boxes are cases that have already been part of a cluster.

Initially, a total of 11 clusters were obtained, distributed as shown in Figure 2. The groupings were made on the basis of the shortest distance between individuals. The largest number of clusters are given in quadrant 1 and 2, since belonging to a specific cluster cannot at the same time belong to another cluster. Therefore, once the grouping is done, the individual is not taken into account for another grouping.

Quadrant 3 does not record the formation of clusters. While quadrant 4 records the formation of two clusters, one of them formed by a single individual.

Step 2: To find the closest pair of clusters and combine them into a single cluster we use the centroid method. This method consists of obtaining the average of each individual contained in the clusters. For example, in Figure 2 it can be seen that Cluster 1 contains 6 individuals corresponding to the diagnostic test, in this sense applying the centroid method corresponds to a value of 36 whose average would be 6. Cluster 2 has a total of 12 whose average is 2. Cluster 4 has

CLUSTER 1	CLUSTER 2	CLUSTER 3	CLUSTER 4	CLUSTER 5	CLUSTER 6	CLUSTER 7	CLUSTER 8	CLUSTER 9	CLUSTER 10	CLUSTER 11
1D	2D	3D	5D	6D	9D	10D	11D	20D	6E	11E
4D	2E		8D	7D	13D	12D	7E	1E	12E	
17D	15E		14D	19D		15D	10E		19E	
18D	16E		3E			16D	13E		20E	
21D	17E		4E			5E				
22D	18E		9E			8E				
			21E			14E				
			22E							
36	12	9	24	21	16	35	16	24	4	0
6	2	9	3	7	8	5	4	12	1	0

FIGURE 2. Initial cluster.

a total of 24 and its average is 3 and so on with the rest of the clusters.

Step 3: We then calculate the Euclidean distances of the centroids by pairs of clusters and observe which is the smallest distance recorded, which will constitute the first clusters to be grouped. the cluster 2 that will be joined will be those whose calculated distance is the smallest, Figure 3.

Step 4: Updating the matrix, from 11 clusters that were initially obtained, after the applied process it was reduced to 6 clusters as shown in Figure 4. In order to get this pairing, we take into account the results of Figure 3. Where, cluster 1 and cluster 5 become one because the distance between them is the smallest. In the same way, cluster 2 and cluster 4; cluster 3 and cluster 6; cluster 7 and cluster 8; cluster 10 and cluster 11 will be joined together, while cluster 9 does not match with any other cluster since the distances are very large.

	CLUSTER 1	CLUSTER 2	CLUSTER 3	CLUSTER 4	CLUSTER 5	CLUSTER 6	CLUSTER 7	CLUSTER 8	CLUSTER 9	CLUSTER 10	CLUSTER 11
CLUSTER 1	0	4	3	3	1	2	1	2	6	5	6
CLUSTER 2		0	7	1	5	6	3	2	10	1	2
CLUSTER 3			0	6	2	1	4	5	3	8	9
CLUSTER 4				0	4	5	2	1	9	2	3
CLUSTER 5					0	1	2	3	5	6	7
CLUSTER 6						0	3	4	4	7	8
CLUSTER 7							0	1	7	4	5
CLUSTER 8								0	8	3	4
CLUSTER 9									0	11	12
CLUSTER 10										0	1
CLUSTER 11											0

FIGURE 3. Initial distance matrix.

CLUSTER 1	CLUSTER 2	CLUSTER 3	CLUSTER 4	CLUSTER 5	CLUSTER 6
1D	2D	3D	10D	20D	6E
4D	2E		12D	1E	12E
17D	15E		15D		19E
18D	16E		16D		20E
21D	17E		5E		
22D	18E		8E		
			14E		

6D	5D	9D	11D		11E
7D	8D	13D	7E		
19D	14D		10E		
	3E		13E		
	4E				
	9E				
	21E				
	22E				

FIGURE 4. Initial matrix update.

It is worth mentioning that the minimum distance between clusters is 1.

Steps 2, 3 and 4 were performed repeatedly obtaining the dendrogram in Figure 5. Where, the height of the branches represents the similarity that exists between individuals/clusters and each vertical line of the dendrogram represents 1 cluster. It is up to the researcher how many clusters to use for the respective comparison.

Applying the agglomerative cluster technique, we obtained 5 possible clusters to analyze. In the initial grouping a total of 11 clusters were obtained. Where clusters 6 and 11 had only 1 individual to analyze, which is why it was discarded.

Group 2 is constituted with a total of 6 clusters, which became the study group since all the groups have at least 2 individuals to analyze. In addition, there is the possibility of analyzing the similarities and differences between the individuals that make it up.

Group 3 is made up of a total of 4 clusters. In this case, clusters 3 and 4 together represent 5 of the 22 individuals to be analyzed. While clusters 1 and 2 comprise the majority of individuals.

Group 4 is constituted with a total of 3 clusters. Where cluster 4 has 2 individuals to analyze, cluster 2 has the same results as in the previous grouping, and cluster 1 comprises the majority of individuals.

Group 5 consists of a total of 2 clusters. Here, all cluster groupings form one cluster. Cluster 1 comprise the majority of individuals while cluster 2 has just 2 individuals to analyze.

Since, Cluster 1 has 2 clusters with a single individual to analyze and that the groups 3, 4 and 5 comprise the majority of individuals in a given cluster making it difficult to establish similarities and differences between these. We chose to analyze the 2 groups formed by 6 clusters.

B. CLUSTER ANALYSIS

After prior analysis of clusters and number of clusters, it was decided to work with the set of 6 clusters, for comparison of results. The 1 to 4 groups represent similar characteristics in terms of number and type of occurrences in categories. While groups 5 and 6 represent characteristics that were found with least frequency.

1) CLUSTER 1

This grouping shows that the category with the highest number of occurrences is ATR, followed by FUN and COM categories. It is observed that this group of characteristics are presented only in the diagnostic tests of the students, Figure 6.

2) CLUSTER 2

In this grouping we observe results of occurrences in both diagnostic and evaluation tests, Figure 7. In diagnostic tests the category with the most occurrences is LIST, followed by NUM and in equal proportions HOL, FUN, COM, REL and HER. While in the evaluation tests the category with the highest number of occurrences is LIS, followed by NUM and ATR.

When comparing the results obtained in the diagnostic and evaluation tests, it is observed that the problem related to categories LIS and NUM continues and, in addition, they are presented in greater number in the evaluation tests.

FUN and COM categories are maintained in both diagnostic and evaluation tests. Also, we found a particular case where student 2, has recurrences in the LIS category in both tests.

The categories HOL, REL and HER present a minimum number of occurrences in the diagnostic tests, but do not appear in the evaluation tests. On the contrary, the categories REA, ACC and ATR do not appear in the diagnostic tests, while in the evaluation tests they present a minimum number of occurrences.

Moreover, in this grouping, one of the students (E2) who is part of the group presents the same number of occurrences in the LIS category in both diagnostic and evaluation tests. He overcomes the difficulties with the FUN category. However, in the evaluation test, the ACC category appears. In general, the student has the same number of occurrences in both tests in the LIS category.

3) CLUSTER 3

This grouping is made up of data only from the diagnostic test, Figure 8. Where, it is observed that there are two categories with the highest number of occurrences COM and HOL, followed by the category ATR.

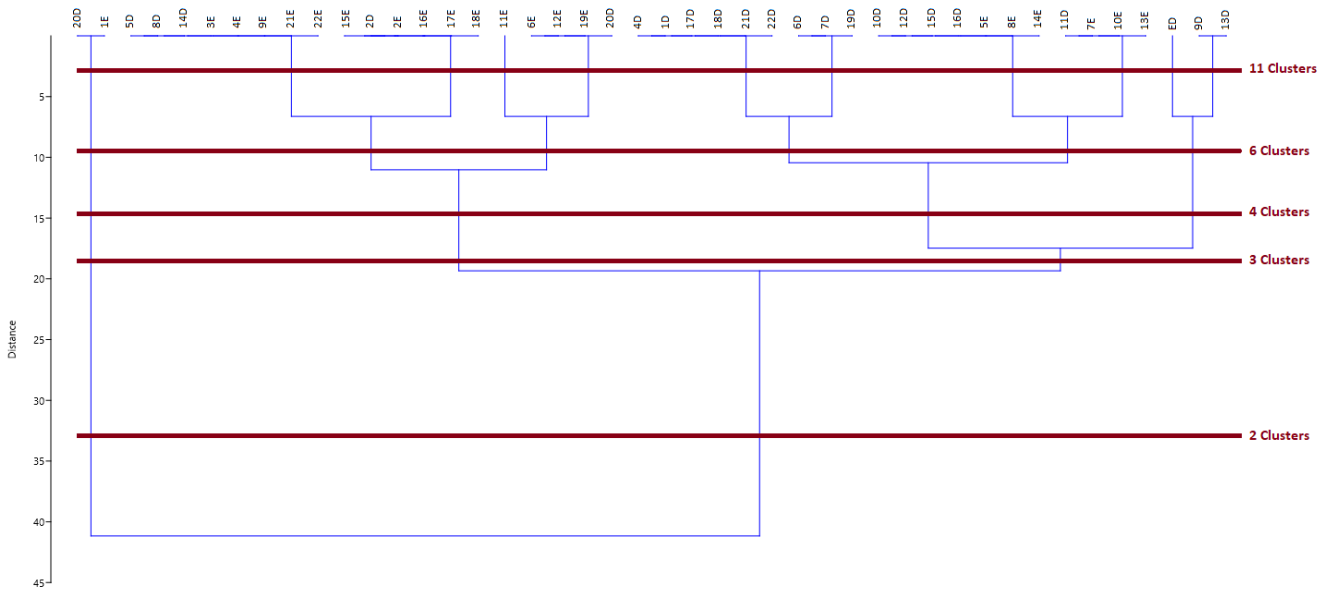


FIGURE 5. Dendrogram of representation of similarities and differences between clusters.

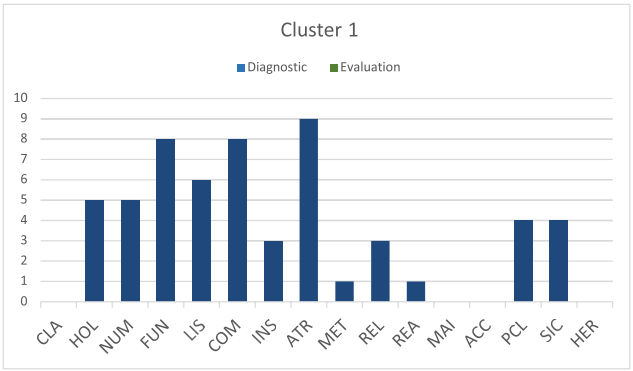


FIGURE 6. Cluster 1.

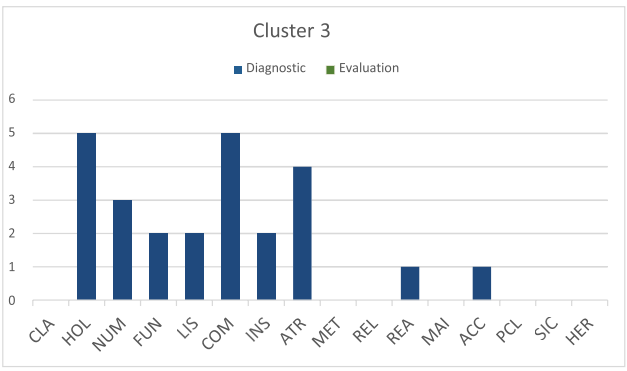


FIGURE 8. Cluster 3.

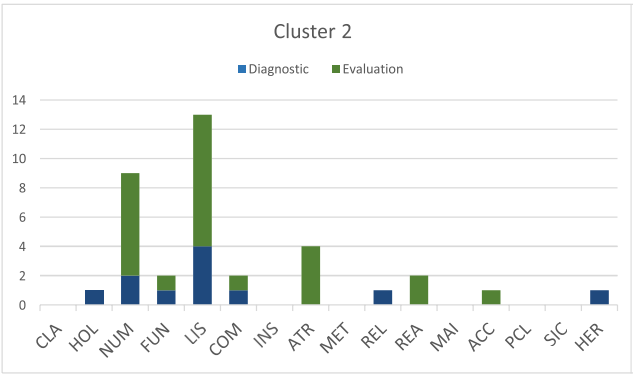


FIGURE 7. Cluster 2.

4) CLUSTER 4

In this group, occurrences are observed in diagnostic tests and evaluation tests, Figure 9. In diagnostic tests the category with the most occurrences is COM, followed by SIC and in equal proportions FUN and INS. While in the evaluation tests the category with the highest number of occurrences is LIS, followed by the categories NUM and ATR.

The categories CLA, HOL, COM, INS PCL and SIC register occurrences in the diagnostic tests, but not in the evaluation tests. This could be due to the fact that the concepts covered by the aforementioned categories were clearer for this group of students. However, in the evaluation test, the categories ATR and MAI appear.

It is evident that the problems related to the categories NUM and LIS in the evaluation tests occur in greater numbers than in the diagnostic tests. On the other hand, the FUN category registers a minimal reduction of occurrences in the evaluation test. While the categories REA and ACC remain with the same number of occurrences in diagnostic and evaluation.

Furthermore, in this grouping we find that student E10 presents problems with the LIS category in the diagnostic

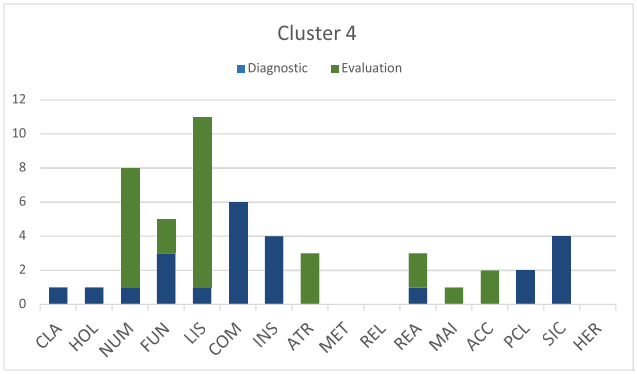


FIGURE 9. Cluster 4.

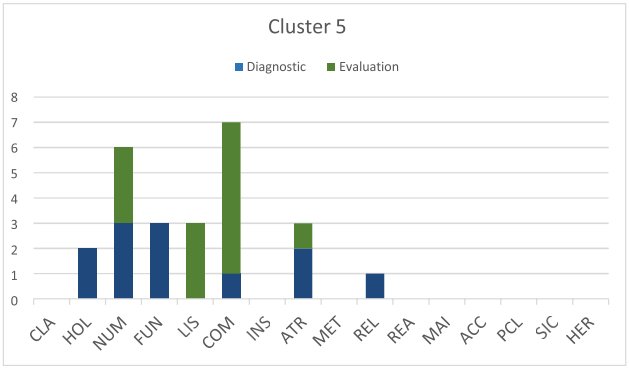


FIGURE 10. Cluster 5.

and evaluation test, with a greater number of occurrences in the evaluation test. Here overcomes the problems with the categories COM, INS and REA. However, new categories such as NUM and ATR appear. In general, in the evaluation test it has fewer occurrences (1 less) but the problem with the LIS category persists.

5) CLUSTER 5

Occurrences in diagnostic and evaluation tests are visualized, Figure 10. In diagnostic tests the categories with the most occurrences are NUM and FUN alike. Followed by the categories HOL and ATR. While in the evaluation tests the predominant category is COM, followed by NUM and LIS.

When comparing the results obtained in the diagnostic and evaluation tests, the COM category increases considerably in the evaluation tests. Category NUM is presented in the same number of occurrences in both diagnosis and evaluation. While the ATR category decreases the number of occurrences in evaluation tests.

In addition, the HOL, FUN and REL categories appear in the diagnostic tests, but not in the evaluation tests. The opposite of the LIS category, which is this group, appears only in the evaluation tests.

6) CLUSTER 6

This group consists of evaluation test results only, Figure 11. It is noted that the category with the highest number of occurrences is NUM, followed by COM and REL categories with the same number of occurrences.

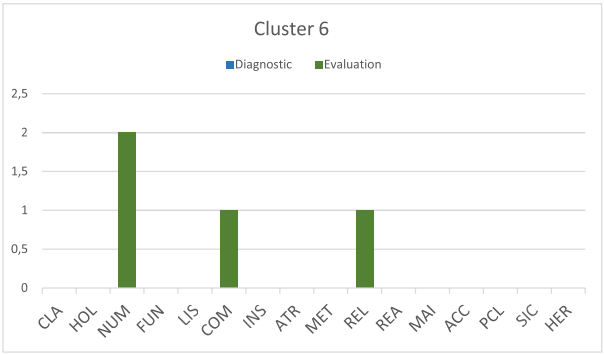


FIGURE 11. Cluster 6.

VI. RESULTS

This section details the results by clusters, the findings of all clusters in general and finally a discussion.

A. RESULTS BY CLUSTERS

- Clusters 1 and 3 represent individuals from diagnostic tests only. In cluster 1 the category with the most occurrence is ATR followed by FUN and COM. While in cluster 3 the categories with the highest number of occurrences equally are HOL and COM followed by the ATR category. In addition, it is evident that in grouping 1 and 3 the categories that represent the greatest difficulty for students are COM and ATR.
- In Clusters 2 and 4, there is a greater number of occurrences in the evaluation tests. Where, the categories that represent greater difficulty to students in both groupings are LIS, NUM and ATR.
- In Cluster 5, the category with the greatest difficulty in both diagnosis and evaluation is NUM, since the number of occurrences is maintained. In addition, in the evaluation test, the COM category increases considerably the number of occurrences compared to the diagnostic test. The opposite happens with the ATR category where the number of occurrences in the evaluation tests decreases compared to the diagnostic tests.
- Clusters 1, 2, 3 and 4 have in common that the category where students present more problems is ATR. In addition, it is observed that in Cluster 2, 4 and 5 the category in common, where there are more occurrences, is NUM.
- Clusters 2 and 4 exist particular cases of the students E2 and E10 respectively, both students present problems in the LIS category in both diagnostic and evaluation tests.

B. OVERALL RESULTS

- It is observed that the LIS and NUM categories considerably increase the number of occurrences in the evaluation tests. This means that, the problems in terms

of the aforementioned categories not only remain the same, but increase in number of occurrences.
- The HOL category occurs only in diagnostic tests with a high number of occurrences. This means that the concept related to holistic has been surpassed.
- Categories FUN, COM, INS and ATR are presented with a high number of occurrences in diagnostic tests while in evaluation tests the number of occurrences is reduced considerably. In other words, the concepts that involve these categories have been partially overcome.
- Categories CLA, MET, PCL show the lowest number of occurrences in diagnostic tests. While in the evaluation tests there are no occurrences.
- Categories MAI and HER do not record occurrences in diagnostic tests. However, they record a minimum number of occurrences in the evaluation tests.

VII. DISCUSSION

This section discusses the results of the difficulties identified in software modeling through class diagram. The findings of this study have implications also for students as instructor of software modeling. Moreover, we present the limitations of this study, especially in the qualitative part.

A. DIFFICULTIES OF SOFTWARE MODELING

In the case of diagnostic tests, the two categories with the highest occurrences are COM and ATR. While those of evaluation tests are LIS and ATR. Below we discuss this results:

1) DEFINING ATTRIBUTES THAT COULD BE A CLASS (ATR)

From the comparative study, it can be seen that the category that persists both, in the diagnostic and in the evaluation test, is called ATR, which we will discussed below.

This category refers to the simplification of a concept by defining it as an attribute of a class, instead of having conceived it as a class by itself due to its complexity. Some students believe that placing "few attributes" is a way to define correctly a concept. They showed this behavior, when they placed in the `Circle` class an attribute called `type` and in `Bet` class an attribute called `typeOfBet`.

Furthermore, difficulties related to misassigned attributes and missing attributes have also been found in the literature [34]. However, there is an important tendency to think that a concept can be defined only with attributes, leaving aside methods. This is also related to the behavior we get used to see in the structured approach, where data is used by functions, as [44] defines systems under the structured approach: *"A software system is a system that manipulates and stores data"*, so that data under this approach have a leading role. The influence of the structured approach on the implementation of the object-oriented approach has already been discussed in the literature [10]. However, their manifestations go beyond giving more relevance to the attributes; these results coincide with those analyzed in [45], where students assigned to the `Employee` class the methods to calculate the salary of an employee, when these should belong to the `Human Resources` class. This behavior shows a clear procedural design where the `Employee` class is in control and the `Human Resources` class is just a data. Detienne [46], [47] also shows his findings related to the problems that novice learners have when decomposing large procedures into smaller functional units, thus reflecting the tendency to place all or most of the functional procedure in a single class. Finally, in the work presented by Ven Yu Sien [34], his findings show a lack of identification of related concepts within the domain problem and problems with misassigned or not assigned attributes.

As in the previous category, students show a clear lack of abstraction by not being able to conceptualize a concept through a class with its own behavior or by reducing a concept to an attribute.

2) CLASSES WITH INAPPROPRIATE OR INSUFFICIENT BEHAVIOR (COM)

Initially, students presented the highest number of problems in the COM category, a problem that decreases in 30% after the previous instruction.

The concepts of a class and an object are very similar in the object-oriented approach, however an object is a concrete entity that exists in time and space, while a class represents only an abstraction [48]. That is why abstraction is a fundamental concept in the object-oriented approach. When defining a real life object as a class, with its attributes and relevant methods, it is a necessity to use abstraction to reduce the object to only the parts that are needed for the software that is being designed [45]. In this sense students in this study have difficulties in giving the class the right behavior and this has been represented in different ways. Sometimes because methods associated with the class do not correspond to the concept that this class represents, or because there is an overload of methods with low cohesiveness between them, or there is a class without a behavior.

Different examples of the COM category were seen in the exercise *Hotel*, when the `Room` class has a method `moveFurniture()`, or in the exercise *Betting* where students assigned to the `Bet` class behaviors related to the verification of aspects of the event. We have also seen classes with an overload of methods with little cohesion between them, for example, a `Bet` class with methods related to the payment and the registration of the gambler. Although the `Bet` class at first glance has one "behavior". The `Bet` class is a clear example of an overloaded class that does many different things. The overloading of methods in a class has also been cited in other works [49], [50]. We also find classes defined only with attributes, such as `Client` class and `Hotel` class, or absence of methods in classes [51].

Many authors describe this problem when defining classes, some of them attribute it to the confusing behavior of assigning a "real" behavior of the physical object to the software object. This was also a finding reported in [52], who conducted a study where students were asked to create a composite class consisting of several simple classes, where

the composite class was called `Room` and the simple classes: `Mirror`, `Bed`, and `Cupboard`. The students placed the `addMirror` method in the `Room` class. The authors interpret this behavior as a student confusion, since it is a possible situation in real life. This involves assigning the erroneous behavior to the `Room` class; related results were also reported by [10].

Other studies conducted by [9], report difficulties of students in conceiving a class as an abstraction of some kind of entity in the real-world problem domain. Although some authors defend the idea that objects have the property of naturalness, which is understood as the property that allows mapping the physical objects of the problem domain to the software [53], [54].

Also some students have created classes built only with the `get()` and `set()` methods, giving the false sensation that these have behavior, when these methods indicate that through them the attributes of that class can be accessed from outside, rather than the behavior of the class itself. Students are often motivated to use `get()` and `set` methods to hide the modules, being a misinterpretation of the Information Hiding Principle [55]. The difficulty of defining objects has also been documented in the literature [45], [54], [56].

3) INCORRECT USE OF MULTIPLICITY BETWEEN CLASSES (LIS)

In the Evaluation tests, the LIS category is found with the highest number of occurrences, which will be discussed below.

Class diagrams allow us to show the classes and the associations between them. Additionally it allows us to visualize the number of objects involved in the association through multiplicity. Thanks to the multiplicity it is possible to define an exact number of objects that are involved; or, if * is used, it indicates that there are an indefinite number of objects in the association [3]. In this way, UML allows to specify the role of the objects that participate in the association.

In this study there were manifestations related to the LIS category in the exercise *Betting*. One of the expected multiplicities was between the `Bet` and `Gambler` classes, since the person making the bet could place several bets, and this was not considered by many students. Most of the students who made the `Bet` and `Gambler` classes performed a multiplicity of 1 to 1 instead of 1 to *. This is evidenced when students did not draw any multiplicity or when they wrote methods such as `getAllBet()` in the class `Bet`, without knowing where or how they handle all bets.

At the software design level, another relationship is aggregation relationship which is used between two classes and is a type of association, which means that an object (the whole) is formed by other objects (the parts) [3]. It is required to define this multiplicity when it want to express the existence of more than one object of the same type. It can also be used aggregation to represent a physical container.

Students do not abstract globally, usually thinking that an object has a specific task. When students realize that the task is to manage a set of objects, they understand the need for some mechanism to deal with multiple instances; however, they are unable to define multiplicity correctly. The difficulty is also related to the conception that a whole and its parts is not always considered like a container, rather this whole/parts relationship is more conceptual [48].

The difficulty of define the multiplicity is a persistent problem that has been manifested in several nuances, being a possible cause of this problem, the difference in between structured and object-oriented approach, where conceptually there is no data and all elements are variables. This possible cause lies largely in a lack of understanding of the object concept rather than in a direct relation to problems with the UML.

LIS is the category that had the highest number of occurrences of problems in the *Bet* and *Hotel* exercises, however in exercise *Circle* it did not have many appearances, because in that exercise it was not required to use multiple objects for its resolution, unlike the first exercises.

We found that 2 of the 22 students who are part of the comparative study, have problems in the same categories in both tests. In both cases the category with which they present problems is LIS. The first case (E2) presents the same number of occurrences in both tests. While the second case (E10) presents a greater number of occurrences in the evaluation test.

B. ADVICES

several authors mention the possible causes of the difficulties in modeling that students usually present [57]–[60]. For example, with respect to assigning an appropriate behavior to classes, some authors agree that students do not perceive the fact that a class models some real-world phenomenon, something in the problem domain. In this sense, the literature contemplates the use of UML to carry out the design process correctly, such is the case of Prasad and Iyer [61] who claim that UML diagrams specify behaviors and scenarios of a given system at various levels of abstraction.

One of the difficulties shown in the diagnostic stage, it was the transfer of models from the structured approach such as the Entity-Relationship model to the object-oriented approach (REL). For this, some authors propose tools to bridge the gap between object-oriented programming and procedural programming, one of them is Web Plan Object Language (WPOL), proposed by Ebrahimi and Schweikert [62].

WPOL is a solution based on a Plan-Object paradigm, where a plan must exist to request, dictate and guide the creation of objects. With a similar intention, Xinogalos [63] uses the objectKarel tool in his study to help students in their transition from procedural/imperative programming to object orientation. Xinogalos introduces object-oriented programming concepts using the microworld approach with objectKarel for a clear, playful and practical presentation of objects and classes, without neglecting other fundamental concepts such as inheritance and polymorphism.

C. THREATS TO RELIABILITY

Qualitative research has been widely criticized for not providing enough information about the analysis of the data and how it has worked from the raw data to its conclusions. This study adheres to the quality criteria presented by Lincoln *et al.* [64], Neuman [65], and Merriam [25] in the educational context.

On one hand, the work has *reliability*, that is, the consistency of the results obtained from the data. To ensure reliability, the researchers of this study, instead of requiring that people outside the research agree that, based on the data collected, the results make sense, are consistent and reliable. They detailed the traceability of the source data and the decisions taken to reach their conclusions. The details of the environment and participants are also described, which will allow other researchers to apply this study in similar contexts.

On the other hand, *validity* that means truthfulness, but in the qualitative context we could rather speak of authenticity, which means capturing a detailed view of the research process. To ensure validity in this research, we have applied strategies such as triangulation, by using several researchers so that each exercise was analyzed and coded separately. The codes and categories were consensual through peer debriefing techniques, ensuring the credibility of the research in this way.

In the presented research students go through different stages of learning: a) when the concepts are presented to the students, b) when they do exercises to try to learn the concepts, and c) when the students take the assessments. In this sense, it should be noted that there is a possible threat to the validity of the research because the stage in which the students present the problems was not identified, nor were the causes of the problems. It is important to recognize that the problems might have been caused by the approach of the teacher while teaching the topic rather than the approach of the students while learning it. Nevertheless, neither the identification of the stage nor the causes were considered within the scope of the study.

In addition, to avoid ethical conflicts regarding the manipulation of the data collected from the students, informed consent forms were prepared to guarantee anonymity and confidentiality of the data obtained from the students. This report was read and signed by the students before the research.

VIII. CONCLUSION

The work carried out allowed us to determine which are the most recurrent difficulties in object-oriented software design and their persistence in a group of university students.

The qualitative study approach was used to generate the documentation from the diagnostic test and student interviews. The thematic analysis of this documentation allowed us to identify, analyze and report patterns within the data, resulting a total of 16 categories.

As a result of the quantitative approach, it was possible to determine the occurrences of the problems of the students in the case study. In addition, the results obtained between the diagnostic and evaluation tests were compared to establish similarities and differences between the cases observed, using the hierarchical clustering technique.

When comparing the number of occurrences of the categories where students present greater difficulty, between the diagnostic and evaluation tests applied at the beginning and at the end of the course respectively, it was found that students present a greater number of difficulties in the LIS and NUM categories. The number of occurrences in these categories not only remains the same, but also increases in the evaluation test. The categories REA and ACC register a lower number of occurrences in the diagnostic tests, but increase their number in the evaluation tests.

On the other hand, the concepts related to the categories FUN, COM, ATR and REL have been partially overcome, their number of occurrences in the evaluation test is lower than the number of occurrences in the diagnostic test.

The categories CLA, HOL, INS, MET, PCL, SIC and HER register a minimum number of occurrences in the diagnostic tests, but they do not register occurrences in the evaluation tests. This could be due to the fact that the concepts covered by the aforementioned categories were clearer for this group of students. No occurrences are recorded in the MAI category in the diagnostic tests, however, it appears in the evaluation tests.

Consequently, the comparative study allowed us to know if difficulties of the students in object-oriented software design have been overcome or not at the end of the course, or at the same time, to know what new difficulties they present when making design decisions.

The comparative study also shows that there are students who have difficulties in the same category with a similar number of occurrences in both diagnostic and evaluation tests. As is the case of students E2 and E10, both students present problems in the LIS category.

ACKNOWLEDGMENT

The authors would like to acknowledge and thank the Anonymous Reviewers for their valuable recommendations, which contributed to improving the quality of this article.

REFERENCES

[1] P. Bourque, R. Dupuis, A. Abran, J. W. Moore, and L. Tripp, "The guide to the software engineering body of knowledge," *IEEE Softw.*, vol. 16, no. 6, pp. 35–44, Nov. 1999.

[2] M. Boehm, "Software engineering," *IEEE Trans. Comput.*, vol. C-25, no. 12, pp. 1226–1241, 1976.

[3] I. Sommerville, "Software engineering 9th edition," Tech. Rep., 2011, p. 18.

[4] C. Hu, "The nature of software design and its teaching: An exposition," *ACM Inroads*, vol. 4, no. 2, pp. 62–72, Jun. 2013.

[5] K. Sanders and R. McCartney, "Threshold concepts in computing: Past, present, and future," in *Proc. 16th Koli Calling Int. Conf. Comput. Educ. Res.*, Nov. 2016, pp. 91–100.

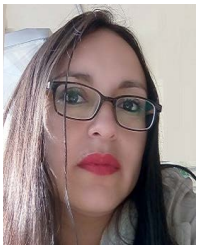
[6] J. Börstler, L. Kuzniarz, C. Alphonce, W. B. Sanders, and M. Smialek, "Teaching software modeling in computing curricula," in *Proc. Final Rep. Innov. Technol. Comput. Sci. Educ. Workshop Groups*, 2012, pp. 39–50.

[7] F. Steimann, "Fatal abstraction," in *Proc. ACM SIGPLAN Int. Symp. New Ideas, New Paradigms, Reflections Program. Softw.*, Oct. 2018, pp. 125–130.
[8] V. Thurner, "Fostering the comprehension of the object-oriented programming paradigm by a virtual lab exercise," in *Proc. 5th Exp. Int. Conf.*, Jun. 2019, pp. 137–142.
[9] S. Xinogalos, "Object-oriented design and programming: An investigation of Novices' conceptions on objects and classes," *ACM Trans. Comput. Educ.*, vol. 15, no. 3, pp. 1–21, Sep. 2015.
[10] P. Flores, J. Torres, and R. Fonseca-Delgado, "Design decisions under object-oriented approach: A thematic analysis from the abstraction point of view," in *Proc. 8th Comput. Sci. Educ. Res. Conf.*, 2019, pp. 89–97.
[11] Z. Ma, "An approach to improve the quality of object-oriented models from novice modelers through project practice," *Frontiers Comput. Sci.*, vol. 11, no. 3, pp. 485–498, Jun. 2017.
[12] D. P. Tegarden and S. D. Sheetz, "Cognitive activities in OO development," *Int. J. Hum.-Comput. Stud.*, vol. 54, no. 6, pp. 779–798, Jun. 2001.
[13] T. Gorschek, E. Tempero, and L. Angelis, "A large-scale empirical study of practitioners' use of object-oriented concepts," in *Proc. 32nd ACM/IEEE Int. Conf. Softw. Eng. - ICSE*, 2010, pp. 115–124.
[14] P. Hubwieser and A. Mühling, "What students (should) know about object oriented programming," in *Proc. 7th Int. workshop Comput. Educ. Res.*, Aug. 2011, pp. 77–84.
[15] M. Kayama, S. Ogata, K. Masymoto, M. Hashimoto, and M. Otani, "A practical conceptual modeling teaching method based on quantitative error analyses for novices learning to create error-free simple class diagrams," in *Proc. 3rd Int. Conf. Adv. Appl. Informat.*, Aug. 2014, pp. 616–622.
[16] W. Silva, I. Steinmacher, and T. Conte, "Students' and instructors' perceptions of five different active learning strategies used to teach software modeling," *IEEE Access*, vol. 7, pp. 184063–184077, 2019.
[17] S. Frezza and W. Andersen, "Interactive exercises to support effective learning of UML structural modeling," in *Proc. Frontiers Education. 36th Annu. Conf.*, 2006, pp. 7–12.
[18] A. Eckerdal and M. Thuné, "Novice Java programmers' conceptions of 'object' and 'class', and variation theory," *ACM SIGCSE Bull.*, vol. 37, no. 3, pp. 89–93, 2005.
[19] M. Thuné and A. Eckerdal, *Students' Conceptions of Computer Programming*. 2010.
[20] J. W. Coffey, "Relationship between design and programming skills in an advanced computer programming class," *J. Comput. Sci. Colleges*, vol. 30, no. 5, pp. 39–45, 2015.
[21] Q. Sun, J. Wu, and K. Liu, "Toward understanding Students' learning performance in an object-oriented programming course: The perspective of program quality," *IEEE Access*, vol. 8, pp. 37505–37517, 2020.
[22] W. A. F. Silva, I. F. Steinmacher, and T. U. Conte, "Is it better to learn from problems or erroneous examples?" in *Proc. IEEE 30th Conf. Softw. Eng. Educ. Training (CSEE&T)*, Nov. 2017, pp. 222–231.
[23] M. LeCompte, "Un matrimonio conveniente: Diseño de investigación cualitativa y estándares para la evaluación de programas," *Revista Elect. Invest. Evalu. Educativa*, vol. 1, no. 1, pp. 1–13, 1995.
[24] R. E. Stake, *The Art Case Study Research*. Thousand Oaks, CA, USA: Sage, 1995.
[25] S. B. Merriam, *Qualitative Research and Case Study Applications in Education*. Chennai, India: ERIC, 1998.
[26] V. Braun and V. Clarke, "Using thematic analysis in psychology," *Qualitative Res. Psychol.*, vol. 3, no. 2, pp. 77–101, 2006.
[27] Y. Sarduy Domínguez, "El análisis de información y las investigaciones cuantitativa y cualitativa," *Revista Cubana Salud*, vol. 33, p. 5, Sep. 2007.
[28] C. Gómez Díaz de León and E. A. D. León de la Garza, "Método comparativo," Tech. Rep., 2014.
[29] B. J. Oates, *Researching Information System Computing*. Thousand Oaks, CA, USA: Sage, 2005.
[30] J. V. Seidel and J. A. Clark, "THE ETHNOGRAPH: A computer program for the analysis of qualitative data," *Qualitative Sociol.*, vol. 7, nos. 1–2, pp. 110–125, 1984.
[31] D. Wicks, "The coding manual for qualitative researchers," *Qualitative Res. Org. Manage.*, Jan. 2017.
[32] S. Friese. (Aug. 2018). *Atlasti 8 user Manual*. Accessed: Sep. 1, 2021. [Online]. Available: http://downloads.atlasti.com/docs/manual/atlasti_v8_manual_en.pdf
[33] K. Sanders and L. Thomas, "Checklists for grading object-oriented CS1 programs: Concepts and misconceptions," *ACM SIGCSE Bull.*, vol. 39, no. 3, pp. 166–170, Jun. 2007.
[34] V. Y. Sien, "An investigation of difficulties experienced by students developing unified modelling language (UML) class and sequence diagrams," *Comput. Sci. Educ.*, vol. 21, no. 4, pp. 317–342, 2011.
[35] P. Pourali and J. M. Atlee, "An empirical investigation to understand the difficulties and challenges of software modellers when using modelling tools," in *Proc. 21th ACM/IEEE Int. Conf. Model Driven Eng. Lang. Syst.*, New York, NY, USA, 2018, pp. 224–234, doi: 10.1145/3239372.3239400.
[36] D. R. Stikkolorum, F. G. de Oliveira Neto, and M. R. V. Chaudron, "Evaluating didactic approaches used by teaching assistants for software analysis and design using UML," in *Proc. 3rd Eur. Conf. Softw. Eng. Educ.*, New York, NY, USA, Jun. 2018, pp. 122–131, doi: 10.1145/3209087.3209107.
[37] D. Berg-Schlosser and G. De Meur, "Comparative research design: Case and variable selection," in *Proc. Configurational Comparative Methods, Qualitative Comparative Anal.*, 2009, pp. 19–32.
[38] L. Bartlett and N. Krawczyk, "Apresentação da seção temática-métodos de educação comparada," *Educação Realidade*, vol. 42, no. 3, pp. 815–819, 2017.
[39] C. Anckar, "On the applicability of the most similar systems design and the most different systems design in comparative research," *Int. J. Social Res. Methodology*, vol. 11, no. 5, pp. 389–401, Dec. 2008.
[40] J. Caïs, L. Folguera, and C. Formoso, *Investigación Cualitativa Longitudinal*, vol. 52. CIS-Centro de Investigaciones Sociológicas, 2014.
[41] J. Marín, "Los análisis cláster de tipo jerárquico y los dendrogramas," *Una Visión Para la Triangulación Metodológica en Los Estudios Comparativos Regionales*. Costa Rica, America: Universidad de Costa Rica, 2008.
[42] R. V. Baños, M. J. R. Hurtado, V. Berlanga, and M. T. Fonseca, "Cómo aplicar un cluster jerárquico en spss," *Revista Innov. Recerca en Educ.*, vol. 7, no. 1, pp. 113–127, 2014.
[43] S. C. Johnson, "Hierarchical clustering schemes," *Psychometrika*, vol. 32, no. 3, pp. 241–254, 1967.
[44] R. Wieringa, "A survey of structured and object-oriented software specification methods and techniques," *ACM Comput. Surv.*, vol. 30, no. 4, pp. 459–527, Dec. 1998.
[45] R. Or-Bach and I. Lavy, "Cognitive activities of abstraction in object orientation: An empirical study," *ACM SIGCSE Bull.*, vol. 36, no. 2, pp. 82–86, Jun. 2004.
[46] F. Détienne, "Design strategies and knowledge in object-oriented programming: Effects of experience," *Hum.-Comput. Interact.*, vol. 10, no. 2, pp. 129–169, Sep. 1995, doi: 10.1207/s15327051hci1002263_1.
[47] F. Détienne, *Software Design–Cognitive Aspect*. Springer, 2001.
[48] G. Booch, R. Maksimchuk, M. Engle, B. Young, J. Conallen, and K. Houston, *Object-Oriented Analysis and Design with Applications*, 3rd ed. Reading, MA, USA: Addison-Wesley, 2007.
[49] P. Flores, N. M. Martínez, and S. P. Roche, "Persistent ideas in a software design course: A qualitative case study," *The Int. J. Eng. Educ.*, vol. 32, no. 2, pp. 937–947, 2016.
[50] M. P. Monteiro, "On the cognitive foundations of modularity," in *Proc. PPIG*, 2011, p. 22.
[51] S. Chren, B. Buhnova, M. Macak, L. Daubner, and B. Rossi, "Mistakes in UML diagrams: Analysis of Student projects in a software engineering course," in *Proc. IEEE/ACM 41st Int. Conf. Softw. Eng., Softw. Eng. Educ. Training (ICSE-SEET)*, May 2019, pp. 100–109.
[52] N. Ragonis and M. Ben-Ari, "A long-term investigation of the comprehension of OOP concepts by novices," *Comput. Sci. Educ.*, vol. 15, no. 3, pp. 203–221, Sep. 2005, doi: 10.1080/08993400500224310.
[53] G. White and M. Sivitanides, "Cognitive differences between procedural programming and object oriented programming," *Inf. Technol. Manage.*, vol. 6, no. 4, pp. 333–350, Oct. 2005.
[54] F. Détienne, "Assessing the cognitive consequences of the object-oriented approach: A survey of empirical research on object-oriented design by individuals and teams," *Interacting Comput.*, vol. 9, no. 1, pp. 47–72, Aug. 1997.
[55] P. Flores, N. Medinilla, and S. Pamplona, "What do software design students understand about information hiding: A qualitative case study," in *Proc. 14th Koli Calling Int. Conf. Comput. Educ. Res.*, 2014, pp. 61–70, doi: 10.1145/2674683.2674697.
[56] D. Svetinovic, D. M. Berry, and M. Godfrey, "Concept identification in object-oriented domain analysis: Why some students just don't get it," in *Proc. 13th IEEE Int. Conf. Requirements Eng. (RE)*, 2005, pp. 189–198.
[57] B. Thomasson, M. Ratcliffe, and L. Thomas, "Identifying novice difficulties in object oriented design," in *Proc. 11th Annu. SIGCSE Conf. Innov. Technol. Comput. Sci. Educ.*, 2006, pp. 28–32, doi: 10.1145/1140124.1140135.

[58] K. Sanders and L. Thomas, "Checklists for grading object-oriented CS1 programs: Concepts and misconceptions," in *Proc. 12th Annu. SIGCSE Conf. Innov. Technol. Comput. Sci. Educ.*, 2007, pp. 166–170, doi: 10.1145/1268784.1268834.
[59] K. Sanders, J. Boustedt, A. Eckerdal, R. McCartney, J. E. Moström, L. Thomas, and C. Zander, "Student understanding of object-oriented programming as expressed in concept maps," *ACM SIGCSE Bull.*, vol. 40, no. 1, pp. 332–336, Feb. 2008, doi: 10.1145/1352322.1352251.
[60] A. Eckerdal and M. Thuné, "Novice Java programmers' conceptions of 'object' and 'class', and variation theory," in *Proc. 10th Annu. SIGCSE Conf. Innov. Technol. Comput. Sci. Educ.*, New York, NY, USA, 2005, pp. 89–93, doi: 10.1145/1067445.1067473.
[61] P. Prasad and S. Iyer, "How do graduating students evaluate software design diagrams?" in *Proc. ACM Conf. Int. Comput. Educ. Res.*, Aug. 2020, pp. 282–290, doi: 10.1145/3372782.3406271.
[62] A. Ebrahimi and C. Schweikert, "Empirical study of novice programming with plans and objects," in *Proc. Innov. Technol. Comput. Sci. Educ.*, New York, NY, USA, 2006, pp. 52–54, doi: 10.1145/1189215.1189169.
[63] S. Xinogalos, "Object-oriented design and programming: An investigation of novices' conceptions on objects and classes," *ACM Trans. Comput. Educ.*, vol. 15, no. 3, pp. 1–21, Jul. 2015, doi: 10.1145/2700519.
[64] Y. Lincoln, E. Guba, and S. Publishing, *Naturalistic Inquiry*. Thousand Oaks, CA, USA: SAGE, 1985. [Online]. Available: https://books.google.com.ec/books?id=2oA9aWlNeooC
[65] W. Neuman, *Social Res. Methods: Qualitative Quant. Approaches*. London, U.K.: Pearson, 2014. [Online]. Available: https://books.google.com.ec/books?id=_o0rnwEACAAJ

PAMELA FLORES received the Engineering degree in computer systems from the Escuela Politécnica Nacional (EPN), in 2005, and the master's degree in information technologies and the Ph.D. degree in software and systems from the Universidad Politécnica de Madrid (UPM), in 2011 and 2016, respectively.

She is currently a Professor at EPN. She also coordinated the Doctorate in Informatics for three years, and now she coordinates the Master in Software at EPN. Her research area is related with object-oriented approach; she has also worked on qualitative research in computer science.



MAYRA ALVAREZ received the Engineering degree in computing for management from the Universidad Politécnica Salesiana (UPS), Quito, Ecuador, in 2018. She is currently pursuing the Master of Research degree in computing, mentioning intelligent systems, from the Escuela Politécnica Nacional (EPN).

She is also working as a Research Technician at EPN. Her research interests include the novice difficulties in programming, emotion recognition, and machine learning.

JENNY TORRES received the Engineer degree in computer systems from the Escuela Politécnica Nacional (EPN), Quito, Ecuador, in 2006; the master's degree in management of networks and telecommunications from the Universidad de las Fuerzas Armadas ESPE, in 2008, before obtaining a SENESCYT Scholarship in Ecuador; the M.Sc. degree in computer science security from the University of Paris-Est Créteil, in 2009; and the Ph.D. degree in computer science from Sorbona University, France, in 2013. She is currently a Professor and a Researcher with the Faculty of Engineering Systems, EPN. She was the Associate Dean and the Director of the Doctoral Program in Computer at EPN. Currently, she is a member of the Incident Response Center CSIRT-EPN and an Editor of the scientific journal *Revista Politecnica* (Scopus indexed).