IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received June 7, 2019, accepted June 18, 2019, date of publication June 21, 2019, date of current version July 10, 2019.

*Digital Object Identifier 10.1109/ACCESS.2019.2924343*

# Data Mining Framework to Analyze the Evolution of Computational Thinking Skills in Game Building Workshops

**ALEXANDRA A. DE SOUZA[1,2], THIAGO S. BARCELOS[1], ROBERTO MUNOZ[3,4], (Member, IEEE), RODOLFO VILLARROEL[5], AND LEANDRO A. SILVA[2]**

[1]Laboratório de Computação Aplicada–LABCOM3, Instituto Federal de Educação, Ciência e Tecnologia de São Paulo, Guarulhos 07115000, Brazil
[2]Programa de Pós-Graduação em Engenharia Elétrica e Computação, Universidade Presbiteriana Mackenzie, Campinas 01302-907, Brazil
[3]Escuela de Ingeniería Civil Informática, Universidad de Valparaíso, Valparaíso 2362735, Chile
[4]Centro de Investigación y Desarrollo en Ingeniería en Salud, Universidad de Valparaíso, Valparaíso 2362735, Chile
[5]Escuela de Ingeniería Informática, Pontificia Universidad Católica de Valparaíso, Valparaíso 2362807, Chile

Corresponding author: Roberto Munoz (roberto.munoz@uv.cl)

This work was supported in part by the Programa de Suporte à Pós-Graduação de Instituições de Ensino Particulares da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES-PROSUP), Brazil, and in part by the Corporación de Fomento de la Producción (CORFO) 14ENI2-26905 "Nueva Ingeniería para el 2030"—Pontificia Universidad Católica de Valparaíso, Chile.

**ABSTRACT** Computational thinking has become a required capability in the student learning process, and digital games as a teaching approach have presented promising educational results in the development of this competence. However, properly evaluating the effectiveness and, consequently, student progress in a course using games is still a challenge. One of the most widely implemented ways of evaluation is with an automated analysis of the code developed in the classes during the construction of digital games. Nevertheless, this topic has not yet been explored in aspects such as incremental learning, the model and teaching environment and the influences of acquiring skills and competencies of computational thinking. Motivated by this knowledge gap, this paper introduces a framework proposal to analyze the evolution of computational thinking skills in digital games classes. The framework is based on a data mining technique that aims to facilitate the discovery process of the patterns and behaviors that lead to the acquisition of computational thinking skills, by analyzing clusters with an unsupervised neural network of self-organizing maps (SOM) for this purpose. The framework is composed of a collection of processes and practices structured in data collection, data preprocessing, data analysis, and data visualization. A case study, using Scratch, was executed to validate this approach. The results point to the viability of the framework, highlighting the use of the visual exploratory data analysis, through the SOM maps, as an efficient tool to observe the acquisition of computational thinking skills by the student in an incremental course.

**INDEX TERMS** Analytical models, computer science education, data mining, data models, education, games, self-organizing feature maps.

## I. INTRODUCTION

The influence and impact of technology in the present social context has provoked debates regarding the educational process, seeking to address what skills will be required for professionals in the 21st-century [1]. Previous works such as [2] and [3] claim that deep knowledge of Computer Science principles and practices should be an integral part of the educational process, regardless of the professional activity that the student will choose in the future.

In addition, [4] suggested that a subset of skills related to Computer Science, defined as *Computational Thinking*, should be developed from the beginning of the students' academic life. Computational Thinking addresses concepts that can help students identify problems and find the best way to solve them using computational resources, i.e., produce a diagnosis and a computational solution.

One of the ways to implement Computational Thinking is through the construction of classes involving the development of digital games, as reported by the works of [5] and [6] through experiments with high school and undergraduate students of courses related to Computer Science.

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

Teaching strategies based on digital games aimed at developing Computational Thinking skills have generated promising results from an educational point of view [7]. Environments such as Scratch have contributed to such results. Scratch uses a block-based syntax and is available in an online web page or as downloadable application. Initially designed for children over 6 years of age, Scratch is currently the most widely used tool for programming education in the world [8] and is being employed not only for children as an extracurricular activity but also for high school and undergraduates students in the area of Computer Science [9]–[11].

Nevertheless, this approach lacks a proper evaluation method that makes it possible to measure the development of competences and skills related to Computational Thinking in students in a more precise manner. In general, teachers and researchers use techniques related to the qualitative paradigm, such as ethnographic observation and surveys, as well as specific tests and quizzes [12]. In addition, there are systems to analyze code, aiming at identify usage statistics of programming commands [13] and structural analyses of the code [14]–[16]. Furthermore, in [1] it is argued that it is necessary to evaluate the skills and competencies of Computational Thinking in educational environments. The proposed goal is to determine the efficacy of the teaching method through learning gains, reflecting the progression over time of students.

Previous works such as [17] propose to measure the acquisition of skills and competencies of Computational Thinking oriented by a framework based on the curriculum of *Exploring Computer Science* and *Advanced Placement Computer Science Principles* [18]. The research started with an analysis of artifacts generated in the Scratch environment available in the online community, using approaches such as structural code analysis, artifact-based interviewing and development of source code improvement activities. This research has influenced other initiatives for automated Scratch code analysis. For example, [14] presents Hairball, a system based on plug-ins with the objective of extracting several characteristics through Scratch code analysis; [19] propose to identify code features that address the use of good and bad programming practices; and [20] introduces a surveys of works that analyze the structures present in an algorithm code.

In complement to the above mentioned works, there are initiatives in literature such as [15], [21], [22] that focus on evaluation rubrics generated from Scratch code through Hairball plug-ins [14] for analysis of the generated artifacts with the goal of supporting teachers on the evaluation of students. The most important result from this initiative is available on the web with the name of Dr. Scratch, which consists of a tool for automatic code analysis based on the proposed rubric for Computational Thinking skills.

However, a gap is still identified in the literature regarding the following: (1) the validity of the evaluation methods to use as a comparative analysis among the students and (2) the use of validity approaches when applied to more structured educational activities that happen during a period of time, as it commonly occurs in a course that lasts, for example, one semester [23].

Motivated by the above mentioned two gaps, this work proposes a data mining framework to discover students' patterns and behaviors that refer to the acquisition of Computational Thinking skills by using Scratch to develop digital games. The analysis of data from game development is supported by the use of the self-organizing maps (SOM) neural network [24]. The main contribution of this framework is to support the teacher in the evaluation of the practical effects produced in the class throughout the execution period and to evaluate the learning method and rubric developed when an incremental teaching model is adopted, in which problems with increasing complexity are presented.

The specific objectives are based on aspects to support the teacher in an exploratory analysis method consisting of the following:

- detecting patterns of students that reveal the evolution of Computational Thinking skills and competencies along the participation in classes of digital game building workshops;
- evaluating the practical effect of the teaching methodology adopted for the development of Computational Thinking skills and competencies during the workshop application.

Thus, a framework for the application of analytical methods of educational data is presented, detailing the activities and practices adopted. The framework was evaluated using real data from case studies based on the application of a digital Scratch Games Workshop, with 130 students from 3 courses and 2 different levels. All activities and the results achieved are described in the following stages: collection of generated codes for each Scratch activity; preprocessing of data involving code structuring, summarization of activities and elimination of outliers; data analysis with application of descriptive statistics and the SOM neural network; and finally, data visualization, including additional information on classroom practices, class assignments, and the results to be achieved by the student with those inferred from the code analysis.

The results obtained demonstrate the feasibility of using the proposed data mining framework, highlighting the use of visual exploratory analysis of the data through the SOM neural network. The findings emphasize the following key aspects related to the development of skills and competencies of Computational Thinking: the topological distribution of the artifacts generated by the students on the SOM network lattice allowed identification of the problem-solving pattern; the weights resulting from the training of the SOM network allowed identifying the realization of the strategy of increasing complexity; and these weights also allowed verification of adherence, deviations, and points of attention linked to the application of the education rubric.

The paper continues as follows. Section II discusses some relevant works related to measuring the acquisition of the knowledge that constitutes Computational Thinking.

Section III presents the proposed data mining framework. Section IV presents and discusses the main results, and finally, Section V exposes the conclusions and future work.

## II. DATA MINING FRAMEWORK TO ANALYZE THE EVOLUTION OF COMPUTATIONAL THINKING SKILLS

The works introduced above to evaluate the computational rubrics developed using data generated in Scratch are based on specific interventions of game workshops provided by teachers [15], [17], [21], [22]. In general, the works selected artifacts of only one game, predominantly those generated at the end of the workshops. Nevertheless, they have a generic use purpose for the validation of Scratch environment artifacts available in the online community. The analyses are based on statistical techniques such as the mean, median, standard deviation, histograms, and hierarchical clustering. More specifically, the studies of [16], [25], [26] that analyze data generated by the Dr. Scratch tool from artifacts available in the Scratch online community have limited results due to the use of the duplicate code, employment of bad programming practices and the various nature of the files (stories, animations and games) as shown by [10], [27], [28]. In addition, the analysis of artifacts indicates their complexity, i.e., it has limitations in the process of measuring the acquisition of Computational Thinking. The authors indicated as possible causes the following problems in the proposal of the workshop: activities of increasing complexity, an educational rubric structure and the long term [10], [27], [28].

The absence of dominion over the data generation environment and the learning track that addresses the complexity of the artifacts is presented in [29] as a gap to be addressed. The study reinforces that the purely data-driven analysis obtained in ready-made artifacts in an online database, without the knowledge of the construction process and the teaching method adopted, can generate information that does not reflect the real progress on Computational Thinking skills in the student. This fact was also identified in the works of [1] and [30].

Another aspect to be noted in these studies is the analysis of data produced through artifacts available in the online community from Scratch. Studies such as [1], [10], [27]–[30] have shown that the absence of knowledge about how the artifacts were generated can provide information that does not reflect the real progress in Computational Thinking skills in the student.

In light of the above, the framework addressed in this paper for measuring Computational Thinking competencies is based on educational data mining, specifically with the use of cluster analysis through the SOM neural network. The choice of SOM was based mainly on providing the visual analysis of the clusters, in addition to generating the mapping of high-dimensional data for a two-dimensional view.

The framework structure was designed following the concept of elements of analysis adopted in modeling systems of software engineering. The meaning of this concept is that when designing a system from its requirements, it must be considered from different views (perspectives): functional, structural and behavioral [31], [32], exploring all aspects involved in the system development. Based on this principle, the framework proposed in this study has a structure based on views. The intention is, similar to the modeling of systems, to know all the perspectives that are involved in the measurement of acquisition of the Computational Thinking competencies.

Thus, the proposed framework is composed of 4 views that will be guiding the activity of measuring the skills: view teaching learning, structural view, process view, and cluster analysis view, as shown in Figure 1.

The framework also proposes the establishment of guidelines that will conduct the elaboration of the other views. Entitled information needs, these guidelines are the questions that must be answered at the end of the process of measuring Computational Thinking skills and competencies. Information needs is represented by circular arrows in Figure 1 and is intended to indicate the guiding role that it performs for the other views.

Each of the views and the information needs structure that compose the proposed framework are presented in detail throughout this section.

### A. INFORMATION NEEDS

According to the work of [33], it is fundamental to define a priori questions to be answered at the end of the process so that the application of data mining tasks can achieve an effective result.

As it is an activity that involves considerable subjectivity, we propose in this work the instantiation of the Practical Software Systems Measurement (PSM) guide, a structural model for measurement activities of a software project [34]. The PSM has as an auxiliary proposal to specify the measures to be used in a given problem and how to conduct the measurement process.

One of the fundamental concepts of the PSM is the so-called information need, which has the purpose of directing the selection of measures and analysis techniques to generate the necessary information for decision making.

In the proposed framework, it is established that the information needs are to focus on the search facts that identify the practical effect of the teaching methodology during application of the digital games workshop. It is the information needs, as already discussed, that will guide the work to be carried out in the other views.

### B. TEACHING LEARNING VIEW

The purpose of this view is to organize the model of the teaching-learning process so that it is possible to identify the student's development through the generated code structure.

To identify the practical effects of the teaching methodology during the execution of the digital games workshop, it is proposed to analyze the predominant code structures in each game by identifying how they are specified in the teaching

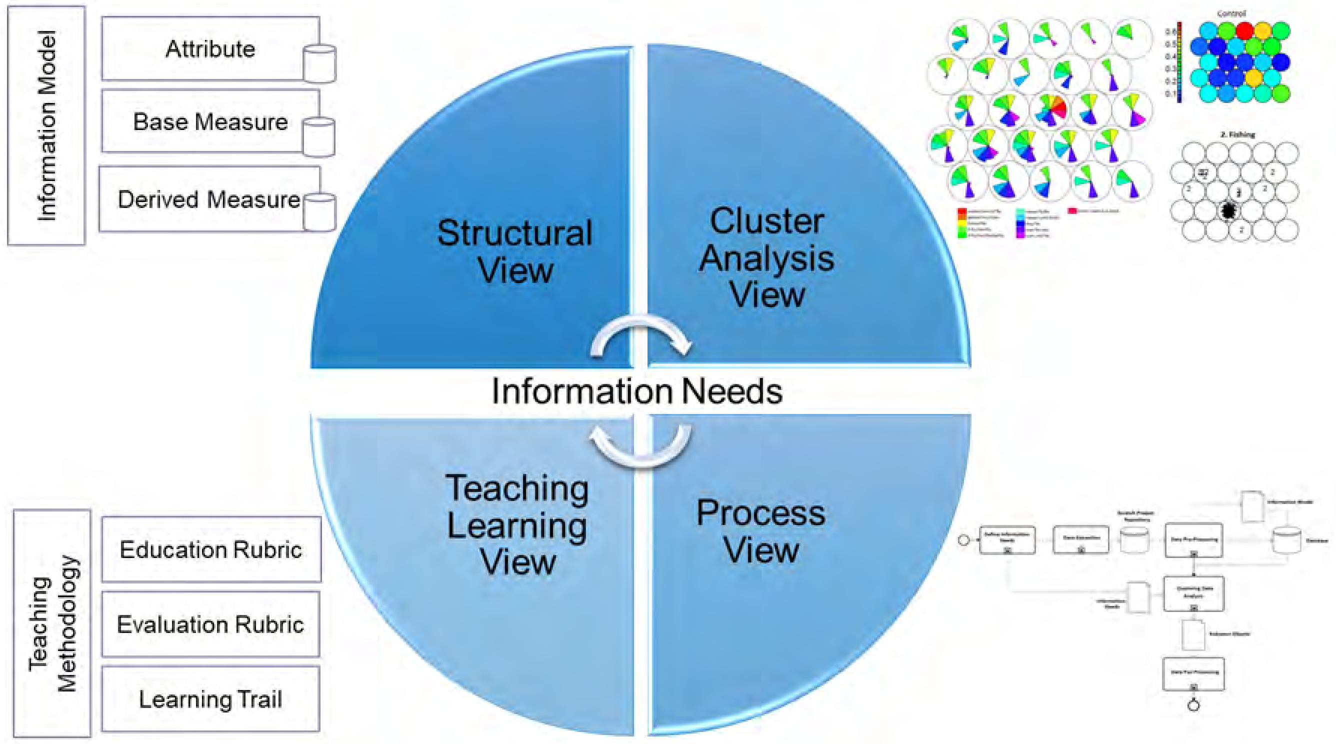

**FIGURE 1.** Proposed framework for educational data mining.

rubric and how they were used by the students in the execution of the workshop.

Therefore, it is necessary to establish a standard for the teaching rubric structure, identifying which Computational Thinking skills and competencies are being developed in each workshop game. The command structures to be used by the student to solve the proposed problem should be detailed, and the skills and abilities to be developed should be linked to those structures.

In this way, the model of the teaching-learning process that is generating the data to be analyzed can be known, with necessary details.

### C. STRUCTURAL VIEW

This view is responsible for defining the structure of the database that will be generated and then analyzed.

To organize the formation of the database structure, another PSM concept called model information [34] is used. It aims to define the metrics used by setting the structure to be collected and the information generated. It has the following composition:

- Information Attribute: characteristics of the artifact or process to be measured.
- Base Measure: quantification of a single attribute.
- Derivative measure: value resulting from the application of a calculation, function or algorithm in two or more base measures.
- Indicator: base or derived measure with decision criteria.

In this study, only the first three concepts are used.

The framework proposes to identify and quantify each command (information attribute) used in the programs, organized by the syntax category of the programming language employed in the digital games workshop.

In the composition of the information model, facts should also be considered that can direct deviations in the acquisition of Computational Thinking knowledge, such as the use of bad programming practices. The study proposes the identification and accounting of *dead code*, that is, occurrences of commands present in blocks of code that will not run at runtime.

The result of the interpretation will be the basic measures of the information model. The database will contain two variables by programming language category of each game developed in the workshop: one representing the total number of commands utilized, and another containing only the total number of commands present in *dead code* blocks.

Another basic measure to be generated is the total number of commands used by the student in game development. It is a derivation of the size-oriented software measurement metric in software engineering called LOC (lines of code) [31], which measures the number of lines of code in a program.

All derived measures will have their specifications guided to the satisfaction of information needs. Using a top-down approach to knowledge structuring, this study proposes the fulfillment of information needs using two different types of structure of the derived measures:

- the first one is composed of variables containing the sum of the occurrences of all the commands present in each category of programming language syntax. The sum of the occurrences of commands present in *dead code* blocks must also be generated by category. The goal is the reduction in the data set dimension, caused by the large number of types of commands in programming languages.

- the second contains binary variables that address the predominance of the use of programming language syntax commands.

In addition, the study proposes the creation of derivative measures related to software engineering product metrics. The objective is to use them as support tools to find patterns that identify the student's difficulty in solving the proposed problem.

### D. CLUSTER ANALYSIS VIEW

Cluster analysis aims to uncover patterns and behaviors that refer to the acquisition of Computational Thinking. Among the set of algorithms used for cluster analysis are self-organizing maps (SOM) [24]. The adoption of this algorithm is justified in this work by performing a mapping of the multidimensional data space in a two-dimensional plane, preserving the topology of the data and mainly allowing the graphical visualization of the relationships between data clusters, contributing to the discovery of knowledge. SOM are used to solve problems involving data clustering, visualization, and abstraction in multivariable correlation studies, converting complex nonlinear statistical relationships to simple geometric relations [24]. SOM have been applied in studies of several research areas such as economics [35], health [36], image analysis and recognition [37], and education [38].

The clustering in the SOM network is performed by calculating a measure of similarity. Commonly, the Euclidean distance is used. Thus, exemplars that are considered as similar are grouped in the same units of the SOM map, while the distance between units indicates the degree of disparity between exemplars contained in each of the units.

The similarity is measured in two moments in the algorithm: during the (1) training and in the (2) mapping of exemplars to the map. Therefore, these phases are complementary to each other. Firstly, each exemplar $\mathbf{x}_i$ represents an artifact generated in the workshop activity, described by features from its source code (as provided in Structural View). Therefore $\mathbf{x}_i = \{x_1, x_2, x_3, \ldots, x_n\}$ represents a exemplar of the data set, that is compared with each synaptic weight $w_u$ of the SOM network, which in general is initialized randomly in the training phase. Being the unit $u$ of the map defined as $\mathbf{w}_u = \{w_{u1}, w_{u2}, w_{u3}, \ldots, w_{un}\}$. In this way, the Euclidean distance is described as:

$$d = \mathbf{x}_i - \mathbf{w}_u, \quad u = 1 \ldots U \tag{1}$$

The shortest distance, obtained from comparing a exemplar with all map units during training, defines the winning neuron and the adjustment of weights occurs in order to preserve the topology of data. In this way, artifacts with similar features will have the same neuron as the winner and, as the features are differentiated, the exemplars are represented by adjacent neurons, as defined in [24].

The second time of Euclidean distance is used is in the exemplar mapping phase. In this moment, the synaptic weights are adjusted and the exemplars are mapped to the SOM network neurons. Therefore, the same operation of "(1)," is applied to associate each artifact produced in the digital game workshop on the map. In practical terms, this is performed by assigning a new identification to the exemplar, that is the neuron number in which it is being mapped ($u$) and, analogously, the artifact information (which can be the identification of the game that was developed) is mapped on the map, thus making visual explorations possible and, at the same time, allowing the identification of the original students data.

As already discussed, this study proposes the use of the top-down approach to knowledge structuring. Thus, the analysis of clusters through the SOM neural network seeks to meet the information needs from a more general view (data by programming language syntax categories), arriving at the specificity of using the commands. Therefore, the cluster analysis will be carried out through the training of the SOM network using the derived measures established by the structural view.

This view is also responsible for ensuring that only relevant information is presented for the knowledge discovery process. The three different approaches of visualization of the SOM map that will be described next are intended to present the features and the relationship of the clusters formed, allowing the assertive interpretation of the data generated by the digital games workshop.

The feature vector visualization map (Figure 5 and 8) is the map where the data related to the synaptic weight vector resulting from the SOM training for each neuron are plotted. In this plot, the neurons are represented in the form of colored circular elements in which it is possible to observe the topological ordering of the sample space of the data set and the contribution of each feature (variable) used in the formation of the clusters.

The heat map of the features vector (Figure 7 and 10) is what represents the density of each of the features present in the cluster by means of colored areas. Areas with greater color scales contain a higher concentration of the features.

The topology distribution map (Figure 6, 9 and 11) presents the correlation of data from the database records, which was not included in the training, with the clustering performed. Its analysis has the objective of identifying trends, seeking to meet the established information needs.

### E. PROCESS VIEW

This view is responsible for the execution of the activities that elaborate the other views. It has a set of steps that allow the collection of data, where the structure of the teaching rubric is defined and carries out the application of digital games workshops; the preprocessing of the collected data, involving the interpretation of codes generated in the application of digital games workshops, their structuring in dimensions of

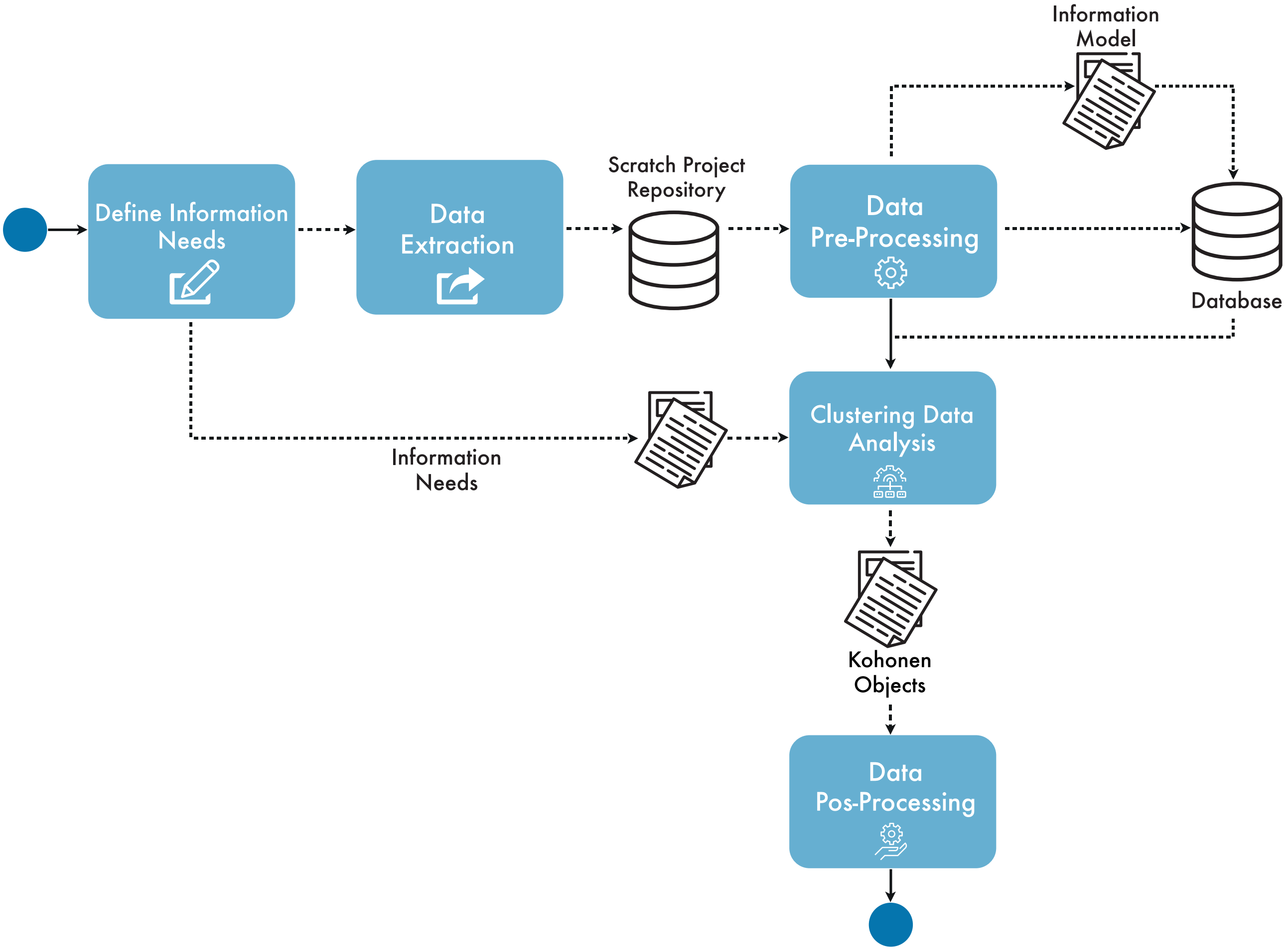

**FIGURE 2. Structure of the proposed process for educational data mining.**

analysis (variables) and data cleaning; and finally, the cluster analysis discovered from the training of the SOM network, with emphasis on visual interpretations of results. This process is depicted in Figure 2, with its details summarized in pseudocode in Table 1 and described in detail in this section.

### 1) DEFINE INFORMATION NEEDS

This stage of the process is responsible for defining the information needs that should guide the knowledge generated by data mining, that is, the information needs that should be set to detect student patterns that indicate the evolution of Computational Thinking skills and abilities.

### 2) DATA EXTRACTION

This stage is responsible for generating the file repository containing each generated code artifact in the execution of the digital games workshop. It is composed of the activities: (1) structure the workshop and (2) generate file repository.

The activity (1), structure the workshop, contemplates the application of the workshop for groups of students. It involves the following tasks:

- Define the teaching rubric structure for each game of the digital games workshop. It is worth mentioning that the proposal of a digital games workshop is not part of the scope of this paper, but rather the proposition of a structural standard for the teaching rubric.
- Define the level of education (basic, high school or undergraduate), course (complementary activity, technical, technologist or engineering) and its subject matter (with or without Computer Science), in addition to the discipline in which it will be applied.
- Establish the number of students, structure the teaching environment, the period of execution and the schedule and duration of each meeting, respecting the sequence defined in the workshop design.
- Execute the workshop following the established method, that is, the sequence of activity games developed.

The activity (2), generate file repository, consists of storing all artifacts generated at each meeting in a central repository.

### 3) DATA PREPROCESSING

This stage of the process is composed of the following activities: (1) process the files of the generated artifacts, (2) data clearing, (3) data aggregation, and (4) data transformation. This stage intends to build the database that will be analyzed,

**TABLE 1.** Details of the process view in pseudocode.

| View | Stage |
|---|---|
| Information Needs | **Define Information Needs**<br>**Input:** To know the evolution of Computational Thinking skills and abilities.<br>**Activity:**<br>(1) Define Information Needs that should guide the knowledge generated by data mining.<br>**Output:** List of Information Needs. |
| Teaching Learning | **Data Extraction**<br>**Input:** List of Information Needs.<br>**Activity:**<br>(1) Structure the workshop;<br>(2) Generate file repository.<br>**Output:** Teaching rubric; File repository with all artifacts (Scratch projects) generated. |
| Structural | **Data Preprocessing**<br>**Input:** List of Information Needs; File repository with all artifacts (Scratch projects) generated.<br>**Activity:**<br>(1) Process the files of the generated artifacts;<br>(2) Data clearing;<br>(3) Data aggregation;<br>(4) Data transformation;<br>**Output:** Information Model; Database with the basic, derived measures and information attributes of each generated artifact. |
| Cluster Analysis | **Clustering Data Analysis**<br>**Input:** List of Information Needs; Database with the basic, derived measures and information attributes of each generated artifact.<br>**Activity:**<br>(1) Define the training attributes;<br>(2) Training;<br>(3) Analyze the quality of training are performed;<br>**Output:** The R object of the SOM algorithm with the cluster analysis and the existing correlation between the data; Average Distance of Exemplars mapped on Neuron map; Distance of Neighborhood Neuron map; Number of Exemplars at each Neuron map. |
|  | **Data Postprocessing**<br>**Input:** List of Information Needs; Teaching rubric; The R object of the SOM algorithm with the cluster analysis and the existing correlation between the data.<br>**Activity:**<br>(1) Generate SOM network maps;<br>**Output:** Feature vector visualization map; Heat map of the features vector; Topology distribution maps. |

including its structure, such as composition of rows and columns, following the proposed information model.

The activity (1), process the files of the generated artifacts, is the processing of files collected using static code analyzers. Guided by the information needs, data extraction should be performed that is relevant to the analysis of the predominance of code structures in each game.

The result of the interpretation is exported in files with a .csv extension, consisting of raw data (without additional processing). It must be organized to have in each line an artifact generated in the workshop, which, for the data analysis, is considered as being a record. In addition, each column is a variable that represents the basic measures of the database.

The activity (2), data clearing, involves the removal of records that represent the database *outliers*. For this action, the mean is used as the position measure and the standard deviation as the dispersion measure of the variable representing the total number of commands used by the student in the game development (basic measure). The objective is to remove the files with extreme aspects in the composition, obtaining a sample that preserves the real context of the artifacts generated in the workshop.

Records that were above and below the established limit (mean +/− standard deviation of the variable representing the total number of commands used) must be removed from the database to be evaluated in isolation.

At the end of the cleaning activity, it has a set of data corresponding to the records that will be used in the analysis cluster.

The activities (3) data aggregation and (4) data transformation must also be guided by the information need. Both have the purpose of generating derived measures that comprise the data to be analyzed in the clustering, as established in the information model. The first activity is responsible for the production of the variables containing the sum of the command occurrences by category of programming language syntax. The second activity is accountable for generating the binary variables that address the predominance of the command's usage.

In the end, the database is generated with the basic and derived measures, in addition to the information attributes referring to the game, class and year of workshop application.

### 4) CLUSTERING DATA ANALYSIS

In this stage, the two forms of cluster analysis (general or specific) provided in cluster analysis view are performed. The activities of (1) define the training attributes, (2) training, and (3) analyze the quality of training are performed.

The activity (1), define the training attributes, must be carried out taking into consideration the forms of analysis. In the most general view, the derived measures must be used as training attributes with the sum variables of the use of the commands of each category of the programming language syntax. For those specified, the training attributes must be the set of derived measures with the binary values that indicate the occurrence of using the command.

For the activity (2), the training is executed with the information defined by the previous activity, the cluster analysis and the existing correlation between the data through the R language and its Kohonen package [39] with the parameterized "som" function are presented in Table 2.

Activity (3), analyze the quality of training, evaluates each result generated by the SOM network training. It can be realized through two criteria: the first represents the intensity of similarity between the elements that are clustered in the same neuron, denominated compactness [41] or cohesion [40]. The second corresponds to the objective of finding

**TABLE 2. Parameterization of the SOM function for cluster analysis application.**

| | |
|---|---|
| Data set | Number of records in the database after completion of the preprocessing stage |
| Grid | The grid size should be the result of the square root of the amount of records in the database, with hexagonal topology and toroidal configuration. |
| Number of Times | The calculation rule presented in [40]: $(1000/\ logd) + 500n$ where: $d$ is the grid dimension and $n$ is the number of neurons in the grid |
| Learning Rate | Initial 0.05 with linear reduction up to 0.01, *default* configuration of the SOM function |
| Distance Functions | General Analysis: *Euclidean*, Specific analysis: *Tanimoto* |

clusters that minimize the similarity between the exemplars and maximize the dissimilarity between the clusters, called separability [41] or dispersion [40]. These measures can be generated using external criteria, which use knowledge about the structure of the data set and the clusters it represents, or the internal ones that use only the information generated by the algorithm during the clustering process.

To validate the quality of the training, this work uses the internal criteria, using the graphs shown in Figures 3 of the case study section. The first graph, called Average Distance of Exemplars mapped on Neuron, represents the aspect of cohesion; in this context, the lower the value is, the better the quality of the clustering. The second map, called Distance of Neighborhood Neuron, represents the dispersion; in this situation, the highest values represent the separability of the clusters.

Another way of validating the quality of the clustering is related to the number of copies mapped in each neuron, also represented in Figure 3 in the graph called Number of Exemplars at each Neuron. The smaller the number of nulls with no record is, represented by the gray color, the better the quality of the cluster mapping.

At the end, the R object of the SOM algorithm is generated and will be used in the data postprocessing stage.

#### 5) DATA POSTPROCESSING

This stage is responsible for addressing the knowledge discovery process. The SOM map view approaches established by the cluster analysis view are generated by performing the initial activity of this step: generate SOM network maps.

This activity will be responsible for the generation of the three types of maps: feature vector visualization map, heat map of the features vector and topology distribution map. It must be executed whenever a new type of visualization is required (using data from the database records as well as the density of each of the features present in the cluster) to meet the information needs.

## III. CASE STUDY

In this section, a study of the application of the proposed framework on data generated in the execution of a digital games workshop is presented. Its description is structured through the stages of the view process, as it is responsible for the implementation of the activities that address the other views.

Although the workshop is not the focus of the work, it is necessary to discuss it in terms of the Computational Thinking skills to be explored.

### A. THE DIGITAL GAMES WORKSHOP

The Workshop on Digital Game Production used in this study [5], [42], was designed with the objective of promoting the development of Computational Thinking skills related to process automation and algorithm construction.

The workshop's structure proposes challenges of increasing complexity in which participants are invited to explore concepts related to game development (*sprites* animation, collision, keyboard and mouse control) and programming fundamentals (variables, conditional structures, loops and messages). The challenges are, most of the time, related to building real digital games.

The workshop's activities are distributed over 12 weeks, with each session lasting approximately two and a half hours. Each activity seeks to develop Computational Thinking through the construction of a game. The proposal is that each of the skills and competence is carried out incrementally and progressively by mobilizing those already developed as new topics that are explored by the students.

The workshop learning trial is composed of 10 games: 1. Cat Encounter with Mouse, 2. Fishing, 3. Guess the Number, 4. Air Battle Simulation, 5. Manual Rock-Paper-Scissors, 6. Automatic Rock-Paper-Scissors, 7. Hash, 8. War Simulation, 9. Breakout and 10. Pacman.

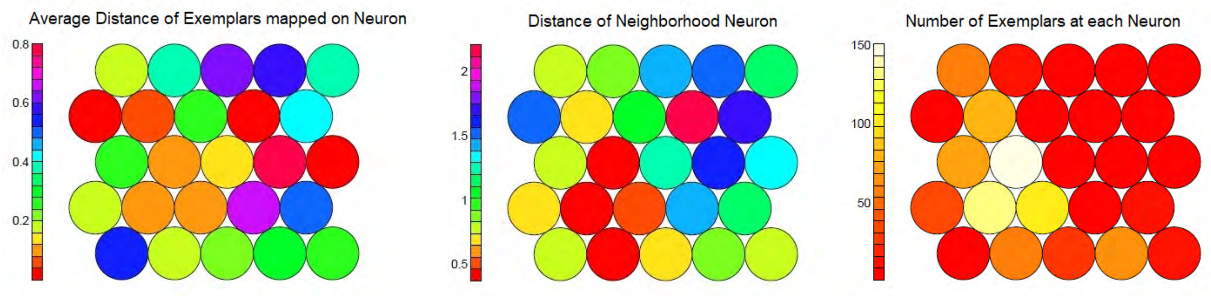

**FIGURE 3. Phase 1 training quality graphics.**

As established in the teaching learning view, the teaching rubric of each game defined by [43] was adapted to a new structural pattern, which maps blocks of commands with Computational Thinking skills to be developed. This 7 of the appendix demonstrated this teaching rubric.

### B. DEFINE INFORMATION NEEDS

As defined in the proposed process, this stage is responsible for defining the information needs that should guide the generated knowledge, aiming to discover patterns in the students that point out the evolution of the Computational Thinking skills and competencies.

Following the top-down approach to knowledge structuring, two sets of information needs were established and organized to be attended in two stages called Phase 1 and Phase 2. The objective is to address aspects that indicate the practical effect of the teaching methodology during the application of the digital games workshop.

For Phase 1, which uses the information obtained from the syntax categories of the Scratch language, the following information needs were defined:

- N1. Identify which code structures and categories of the Scratch environment predominate in the execution of the workshop and its relation to teaching methodology.
- N2. Identify which structures and categories of the Scratch environment predominate in each game and its relation to the teaching rubric.

For Phase 2, which uses the predominance information of the commands present in each category, the following information needs to be satisfied:

- N1. Identify which Scratch commands predominate in the generated artifacts in each category.
- N2. Know which Scratch commands predominate in the games generated in the workshop, as well as their relationship with the teaching rubric.
- N3. Discover which Scratch commands are influencing the calculation of the cyclomatic complexity and what effect they have on the execution of the workshop (applies only to the *Control* category).

### C. DATA EXTRACTION

In the scope of this study, four offers of the workshop were selected. It was offered in 2013 to 31 students in the technical course of high school of the São Paulo Federal Institute, who were enrolled in the discipline of Programming Logic over 12 weeks. In the same year, the workshop was also applied to 18 incoming students of the Informatics Engineering course as a complementary activity to the regular activities of the course at the University of Valparaíso, Chile.

In the first semester of 2017 and 2018, the Workshop was again proposed at the São Paulo Federal Institute in the discipline of Programming Logic for students of the technical course integrated with the high school, comprising 40 participants in 2017 and 41 in 2018.

For the composition of the repository, a total of 940 Scratch code files were collected, and everything was generated in the four proposals of the Digital Games Workshop.

### D. DATA PREPROCESSING

This stage is responsible for generating the database containing the information extracted from the 940 files in the repository.

Following the script presented for the activity (1) process the files of the generated artifacts, a file with the .csv extension was produced with the information produced by the *Hairball* tool *plug-ins* [14], without any additional treatment and respecting the structure presented in Tables 3 and 4. The first table contains the basic measurements extracted by the code analyzers for the Scratch environment, and the second one contains the information attribute with identification data of the executed workshops.

**TABLE 3. Database structure - Specifying base measure for scratch environments.**

| Variable | Description |
|---|---|
| $Commands_{nc}$ | Accumulation of the amount of each coding construct present in the Scratch Project. The index *n* is used to represent coding constructs that constitute each *c* Scratch category. This variable was generated by (*nxc*) columns in the database. |
| *TotalCommands* | The total number of coding constructs used in all Scratch Project. |
| $CommandsDead_{nc}$ | Accumulation of the amount of each dead coding construct present in the Scratch Project. The index *n* is used to represent coding constructs that constitute each *c* Scratch category. This variable was also generated by (*nxc*) columns in the database. |
| *Costumes* | The total number of *costumes* in the Scratch Project. |
| *Sprites* | The total number of *sprites* in the Scratch Project. |
| *Backdropss* | The total number of *backdrops* in the Scratch Project. |

**TABLE 4. Database structure - Specifying information attributes.**

| Variable | Description |
|---|---|
| *PeriodWorkshop* | Period (year, semester, month) in which the workshop was performed. |
| *Game* | The game Scratch workshop project. |

Next, the activity (2) data clearing was performed for the removal of data representing the *outliers* of the database. For this, we used the mean and standard deviation of the *TotalCommands* variable of Table 3. As established in the proposed process, this is the case study variable that represents the number of Scratch command blocks used in the artifacts, a derivation of the software engineering LOC metric.

The records that were above and below the established limit (mean +/− standard deviation of the variable *TotalCommands*) were then removed. After completing the cleaning process, the database was composed of 861 files with 205 attributes.

After the activity (3) data aggregation was executed, generating the derived measures to meet the information needs established for Phase 1 of this case study. In this way, we created a set of 10 variables "(2)" containing the sum of occurrences of the commands by Scratch categories: *Motion*, *Looks*, *Sound*, *Pen*, *Data*, *Events*, *Control*, *Sensing*, *Operators* and *More Blocks*. Again, this process was performed using only the occurrences of commands present in blocks that

**TABLE 5. Database structure - Specifying derived measure.**

| Variable | Description | Formula |
|---|---|---|
| $SumCommands_c$ | Sum of the total of each coding construct used in the Scratch Project, organized by ten Scratch categories. The index $c$ is used to represent the Scratch category. | $$SumCommands_c = \sum_{c=1}^{10} Commands_{nc} \quad (2)$$ |
| $SumCommandsDead_c$ | Sum of the total of each dead coding construct used in the Scratch Project, organized by ten Scratch categories. The index $c$ is used to represent the Scratch category. | $$SumCommandsDead_c = \sum_{c=1}^{10} CommandsDead_{nc} \quad (3)$$ |
| $BinaryCommands_{nc}$ | Conversion to the binary of the accumulation of each coding construct used in the Scratch Project, organized by ten Scratch categories. The index $n$ is used to represent coding constructs that constitute each $c$ Scratch category. This variable will generate $(nxc)$ columns in the database. | $$\begin{gathered} if\ (\ Commands_{nc}\ > 0\ ) \\ then\ BinaryCommands_{nc} = 1 \\ else\ BinaryCommands_{nc} = 0 \end{gathered} \quad (4)$$ |
| $Complexity$ | Scratch Project Cyclomatic Complexity. According to [44], this metric can be represented by a control flow graph ($G$), where nodes ($n$) are control statements and edges ($e$) indicate the direction of control flow. For programs without "goto" declarations, the value of the cyclomatic complexity is one more than the number of program conditions. | $$Complexity_G = e\ -\ n\ +\ 2 \quad (5)$$ |

are not executed at runtime (*dead code*), creating 10 other variables "(3)".

Next, the data transformation activity (4) was performed, generating the variables that are used to meet the information needs of Phase 2 of this study. New derived measures are created in the database, containing the binary conversion of the values of all ($n$x$c$) attributes representing the $Commands_{nc}$ variable. For this, the rule defined in the framework was applied: values greater than zero were assigned a value of 1, and the others remained as zero "(4)".

The derived measure representing the cyclomatic complexity metric was also generated. The form of calculation "(5)" described in [44] was employed, using the commands with conditional characteristics of the Scratch environment: *if %s then %s*, *if %s then %s else %s*, *repeat until %s* and *wait until %s*. This measure was used to engage the third Phase 2 information need.

All derived measures are presented in Table 5.

### E. CLUSTERING DATA ANALYSIS

This section aims to present how the database generated by the previous step will be submitted to cluster analysis to meet the information needs established for the first and second phases of the proposed method. Again, it is emphasized that for Phase 2, the activities of this step should be performed for all 10 Scratch categories. In this study, this stage is configured to perform the cluster analysis only of the *Control* category.

As established in the activity (1), define the training attributes, the variables were extracted from the database structure to be submitted to the training algorithm:

- For Phase 1, the 22 variables that met the derived measures: *Sprites*, *Costumes*, *Backdrops*, $SumCommands_c$ and $SumCommandsDead_c$, are presented in Tables 3 and 5.

**TABLE 6. Parameterization of the SOM function for cluster analysis application - Case study.**

| | |
|---|---|
| Data set | 861 records |
| Grid | Dimension 5x5, with Hexagonal topology and Toroidal configuration |
| Number of Times | 14.000 |
| Learning Rate | Initial 0,05 with linear reduction up to 0,01 |
| Distance Functions | Phase 1 - *Euclidiana*<br>Phase 2 - *Tanimoto* |

- For Phase 2, the 11 columns that met the derived measures were extracted from the database structure: *BinaryCommands*$_{nc}$ of the *Control* category commands employed in the execution of the workshop (*create clone of %s*, *delete this clone*,*forever%s*,*if %s then %s*, *if %s then %s*, *repeat %s%s*, *repeat until %s%s*, *stop %s*, textit wait %s secs, *wait until %s*, *when I start as a clone*).

The other variables of each record are not included in the training. However, those that represent the cyclomatic complexity and the game of the workshop were analyzed for the identification of trends, seeking to meet the established information needs.

Then, the data belonging to the variables of both phases were sent to the existing clustering and correlation analysis using the R language and its Kohonen package with the parameterized "som" function as presented in Table 6, following the guidelines established by the activity (2) training.

The analysis of the quality of the clustering generated for each execution of the SOM function was carried out using the criteria established in the activity (3), analyze the quality of training. Thus, the graphs of Figures 3 and 4 represent the objects resulting from the training process with:

- short **Average Distance of Exemplars mapped on Neuron**, that is, more similarity between the elements that are clustered in the same neuron.

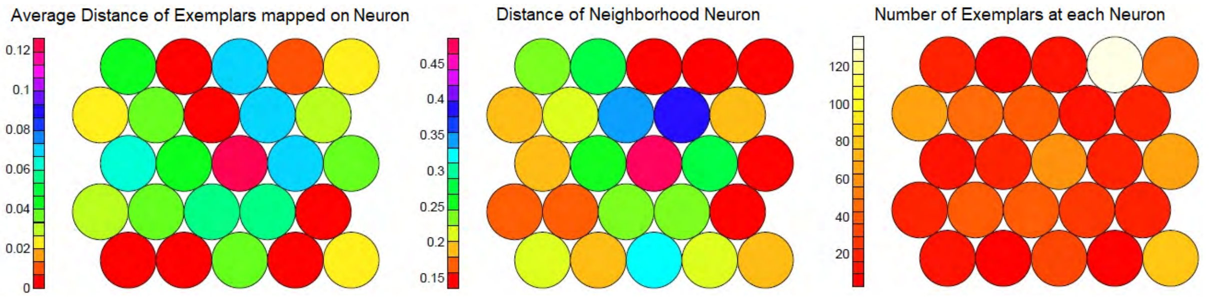

**FIGURE 4.** Phase 2 training quality graphics.

- larger **Distance of Neighborhood Neuron**, that is, less similarity among the elements represented in each cluster.
- **Number of Exemplars at each Neuron** larger than zero, that is, no neurons in the map have the gray color.

### *F. DATA POSTPROCESSING*

This step seeks to meet the established information needs. This is conducted by visualizing the relationships between the clusters and the knowledge provided by the SOM map, which allows the presentation of the features of the neurons. To fulfill this purpose, this work uses the maps described in the cluster analysis view of the proposed framework, and the obtained results are presented in the results section below.

## IV. RESULTS

This section is structured in two parts, demonstrating the results obtained to meet the information needs established for Phases 1 and 2.

### *A. PHASE 1*

This section presents the analyses performed using the SOM maps proposed in the cluster analysis view, aiming to meet the established information needs for Phase 1.

#### 1) FIRST INFORMATION NEED

Initially, we present the results obtained that meet the first information need of this phase, namely, "Identify which code structures and categories of the Scratch environment predominate in the execution of the workshop and its relation to teaching methodology".

The map containing the vector of resulting weights of SOM training (Figure 5) and the map with the topological distribution of games (Figure 6) were created to identify which code structures and Scratch category were most frequently employed in offering the workshop.

From the analysis of Figure 5, it is possible to identify the weight of each category of the Scratch environment in each neuron, both in the global context of the code and in the *dead code*, represented by the segments highlighted in the legend: All Code Lines and Exclusive *Dead Code* Lines (snippets of code that will not run at runtime).

Analyzing both Figures 5 and 6, it can be identified that the behavior approaches the realization of the strategy showing that the activities of increasing complexity are related to the acquisition of Computational Thinking skills and competencies. Analyzing the weights assigned to each category of the Scratch environment in neurons and the topological distribution of games identifies a relationship between the growth of the weight values and the learning trial followed by the workshop.

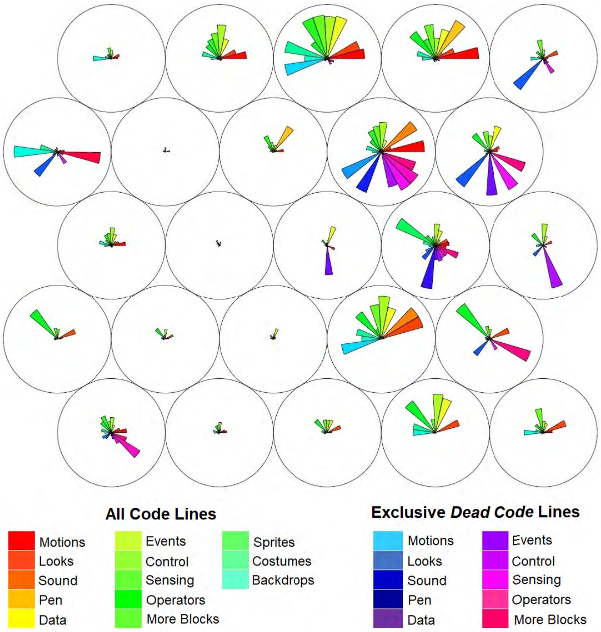

**FIGURE 5.** Phase 1 feature vector visualization map.

The visual limitation due to the spread of the artifacts caused by the *dead code* intensity can lead to the student's difficulty in solving proposed problems and consequently to the development of Computational Thinking skills. In this way, identifying the categories in which this behavior predominates helps the teacher to know the limitations of the students regarding the correct understanding of how to use the commands of these categories.

For this, the heat map of the 10 Scratch categories was generated, shown in Figure 7. The graph visualization enables identifying which categories have a higher trend for this behavior. In this case, the predominance occurred in the categories *Looks*, *Control*, *Sensing* and *Operators*, indicating

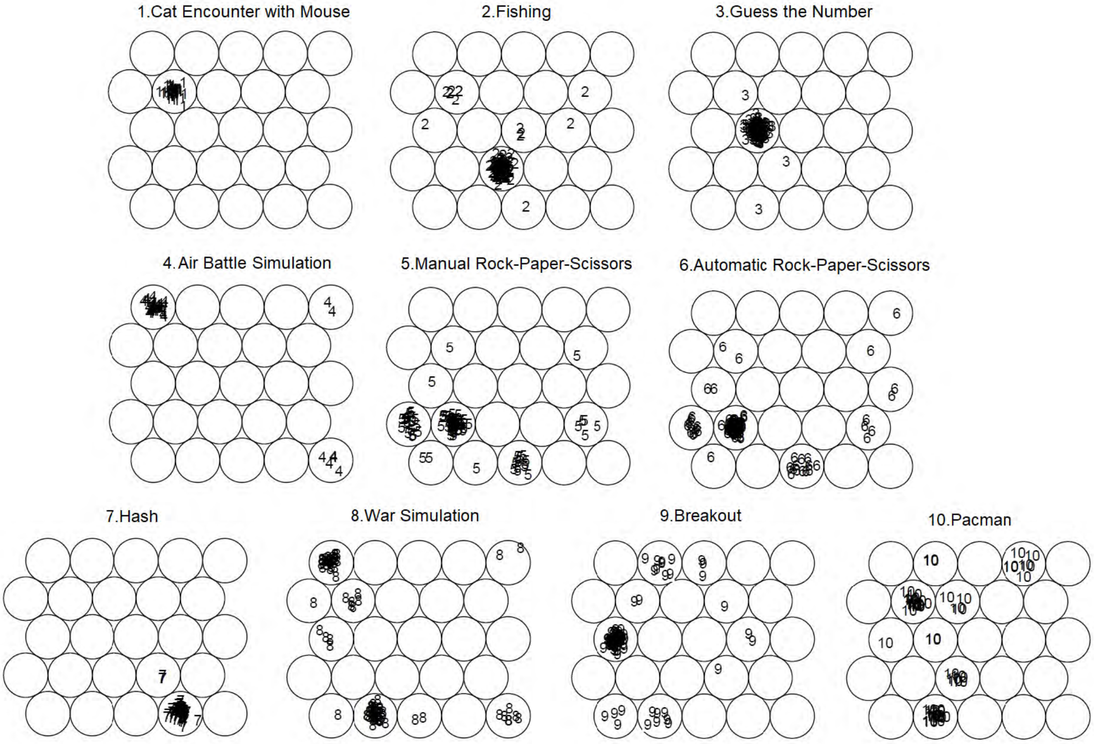

**FIGURE 6.** Phase 1 map with each workshop game.

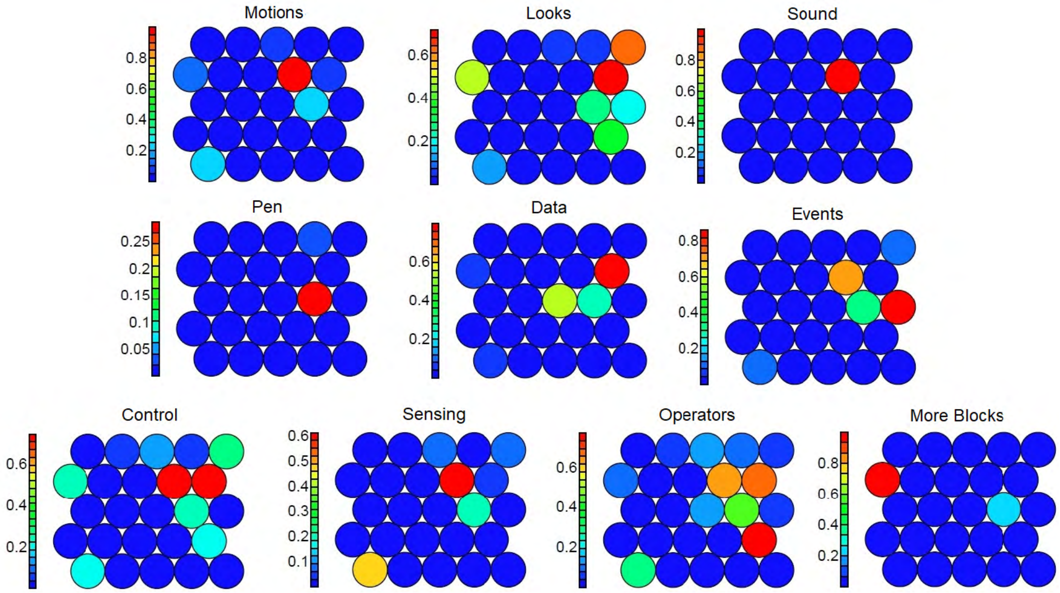

**FIGURE 7.** Heat map of scratch categories with *dead code*.

that through application of the workshop, the students had the highest difficulty in understanding and practicing the correct use of their commands.

Another way to help the teacher understand the spread of artifacts caused by *dead code* is the identification of which games in the workshop dominate this behavior.

Analyzing the concentration of the *dead code* in the neurons of the right region of the map shown in Figure 7 and of the artifacts present in this area in Figure 6, it is possible to identify that the games with the greater amount of *dead code* are 2. Fishing, 4. Air Battle Simulation, 5. Manual Rock-Paper-Scissors, and 6. Automatic Rock-Paper-Scissors, where the last two had numerous occurrences. This fact can demonstrate the difficulty that some students had in determining which commands to use to generate one solution to the proposed problem and, consequently, problems encountered in the evolution of the skills and competencies of Computational Thinking.

### 2) SECOND INFORMATION NEED

This section will present the analysis performed to meet the second need of Phase 1: "Identify which structures and categories of the Scratch environment predominate in each game and its relation to the teaching rubric."

The set of artifacts generated for each game was evaluated in the analysis, aiming to understand the process of the development of Computational Thinking realized in the execution of the workshop, at the time of its elaboration.

The joint analysis of the maps of Figures 5 and 6, in conjunction with the table with the teaching rubric of the workshop 4, identifies the categories of the Scratch environment defined for each game and the related Computational Thinking skills and competences. In this way, it is possible to check how adherent the execution of the workshop was to the design.

Taking as an example of this analysis the game 5. Manual Rock-Paper-Scissors and its teaching rubric present in Table 7, the following remarks are made:

- 1. The Scratch categories directly linked to the Computational Thinking skills and competence indicated by the teaching rubric of this game were those that have higher weights: *Motion*, *Looks*, *Events*, *Control*, *Operators* and *Sensing*. This makes it possible to infer that these skills and abilities were directed, at least in a broader context.
- 2. It is also possible to identify the high presence of *dead code* in the categories mentioned in item 1, a fact that can address some weaknesses in the approach used for this game and, consequently, issues that affect the development of the Computational Thinking skills and competencies proposed.
- 3. The structure that directs the creation of the new *sprite* of the teaching rubric is present in the neurons where this game concentrated, but weighing slightly below those mentioned in item 1.
- 4. The weight of categories *Sound*, *Pen*, *Data* and *More Blocks* do not have weights in the neurons of this game, as predicted in the teaching rubric.

In light of the above, it is evident that throughout the construction of game 5. Manual Rock-Paper-Scissors, the Scratch categories representing the structures designed by the teaching methodology were used. However, some students were generating code that would not run at runtime (*dead code*). Regarding the development of Computational Thinking skills and competencies described in the teaching rubric of this game, the analysis demonstrates that the code structure of the artifacts generated in the workshop was adherent to the specified one. However, the high number of *dead code* occurrences can indicate some divergence in this process that, if during the workshop offering is identified, allows the teacher to make some kind of adjustment.

## B. PHASE 2

To meet the information needs established for Phase 2 of this study, this section presents the analysis activities to discover the influence of the code instructions of a Scratch category in the digital games workshop application. This stage must happen for all 10 Scratch categories. This study presents the analyses uniquely in the *Control* category.

### 1) FIRST INFORMATION NEED

Knowing which Scratch commands in the *Control* category were the most frequently used in the workshop's artifacts meets the following information need: "To Identify which Scratch commands predominate in the generated artifacts in each category." Thus, in response to this need, the map shown in Figure 8 containing the weight vector resulting from the SOM training was generated.

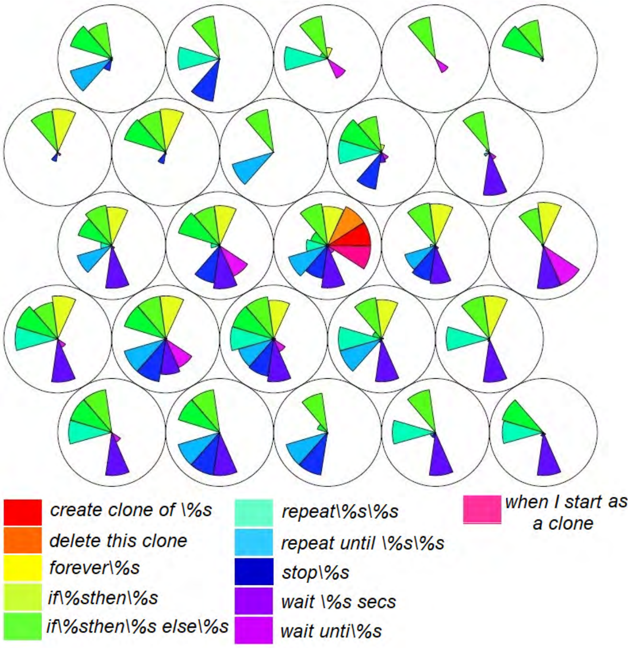

**FIGURE 8.** Phase 2 feature vector visualization map *Control* category.

Analyzing Figure 8, it is possible to observe that the commands: *forever%s*, *if%sthen%s*, *if%sthen%s else%s*, *repeat%s%s*, *repeat until %s%s*, *stop%s*, *wait %s secs* and *wait unti%s*, are the most used commands during the workshop and are present in the teaching rubrics. Additionally, in this figure, the predominance of the *if%sthen%s* command is identified, which may indicate the excessive use of this command by the students during the workshop.

The *create clone of %s*, *delete this clone* and *when I start as a clone* commands are less frequently used in running the workshop. This behavior adheres to the teaching methodology, since only the teaching rubric of the game 4. Air Battle Simulation has these commands, as can be observed in Table 7.

### 2) SECOND INFORMATION NEED

Satisfying the second information need concerns the following: "Know which Scratch commands predominate in the games generated in the workshop, as well as their relationship with the teaching rubric." The evaluation of this need utilized the map of Figure 9, which shows the topological distribution

**TABLE 7. Adaptation of the scratch teaching rubric of the games used in this study ([43], page 245).**

| Game | Objective | Scratch Category | Scratch Blocks | Computational Thinking Skills |
|---|---|---|---|---|
| 1.Cat Encounter with Mouse | In an already elaborated program, the student assigns behaviors to two *sprites* as they move via the left and right arrows and touch each other. | *Motion* | *move %s steps* | Modeling and Simulation |
| | | Sensing | *touching %s?* | |
| | | Events | *when %s key pressed* | Algorithms |
| | | Control | *if %s then %s* | |
| 2.Fishing | In an already developed program, the student creates a *sprite* that appears in successive randomly drawn screen positions. Clicking the mouse on the *sprite* causes the increase in a variable with its value displayed in the corner of the screen. | *Sprite* | create a new | Modeling and Simulation |
| | | *Looks* | *say %s* | Data Representation |
| | | *Data* | create one variable | |
| | | *Events* | *when this sprite clicked* | Algorithms |
| | | *Control* | *wait %s secs, forever%s,repeat %s%s, if %s then %s* | |
| | | Operator | *%s < %s* | |
| 3.Guess the Number | The first game produced from the beginning by students, it generates random numbers stored in a variable. The *sprite* asks the player for a number. The game must inform if the value entered is lower or higher than the drawn number, allowing a new play. | *Sprite* | create a new | Modeling and Simulation |
| | | *Looks* | *say %s* | Data Representation |
| | | *Data* | creates two variables | |
| | | *Events* | *when @greenFlag clicked, ask %s and wait, answer* | Algorithms |
| | | *Control* | *stop %s, forever%s, repeat %s%s* | |
| | | *Operator* | *pick random to %s, %s < %s,%s = %s, %s > %s* | |
| 4.Air Battle Simulation | This game must implement control of the movement of actors with logical conditions and repetition. Three *sprites* move on a Cartesian plane, leaving random positions. At each interval of 1 to 3 seconds, there must be a change of costumes. | *Sprite* | create three new | Modeling and Simulation |
| | | *Motion* | *point in direction %s, change %s by%s, x position, y position* | |
| | | *Looks* | *switch costume to %s,next costume, go to front* | |
| | | *Sensing* | *touching %s?,%s of %s* | |
| | | *Data* | creates three variables, *set %s to %s, change %s by %s* | Data Representation |
| | | *Events* | *when @greenFlag clicked, when this sprite clicked* | Algorithms |
| | | *Control* | *wait %s secs, forever%s,repeat %s%s, if %s then %s* | |
| | | *Operator* | *pick random to %s, %s < %s,%s = %s, %s and %s* | |
| 5.Manual Rock-Paper-Scissors<br>6.Automatic Rock-Paper-Scissors | In this game, two *sprites* (players) must be created. When activated by a set of three keys, it changes the costume for the gesture of the stone, paper or scissors. In this game, the students define keys for each possible action of both players. They also determine how the result (win or draw) will appear on the screen. | *Sprite* | creates two new | Modeling and Simulation |
| | | *Motion* | *glide %s secs to x:%s y:%s* | |
| | | *Looks* | *switch costume to %s, switch backdrop to %s* | |
| | | *Events* | *when I receive %s, broadcast %s* | |
| | | *Control* | *forever%s, wait until %s, if %s then %s* | Algorithms |
| | | *Events* | *when @greenFlag clicked, when %s key pressed* | |
| | | *Sensing* | *%s of %s* | |
| | | *Operator* | *%s = %s* | |
| 7.Hash | In a nine-position tray, each position has a *sprite* that has three costumes: an "O", "X" and white rectangle. When clicked, the *sprite* changes its costume depending on the player who will participate. The algorithm must, at each move, check the win settings. | *Sprite* | creates nine new | Modeling and Simulation |
| | | *Looks* | *switch costume to %s* | |
| | | *Data* | creates one variable, *set %s to %s* | Data Representation |
| | | *Events* | *when @greenFlag clicked, when this sprite clicked* | Algorithms |
| | | *Control* | *if %s then%s else %s* | |
| | | *Operator* | *%s = %s* | |
| 8.War Simulation | In this game, two *sprites* (tanks) must be created. One stays static in the lower left corner of the screen and can only be rotated clockwise or counterclockwise. The other part has a fixed position and moves towards the fixed tank, defending itself from the projectile (*sprites* triggered by the spacebar). The linear trajectory of the projectile must be defined by the student so that, when fired at an enemy, it explodes. | *Sprite* | creates three new | Modeling and Simulation |
| | | *Motion* | *turn @turnLeft %s degrees, if on edge bound,*<br>*turn @turnRight %s degrees, point towards %s* | |
| | | *Looks* | *next costume, backdrop name,switch costume to %s* | |
| | | *Events* | *when I receive %s, broadcast %s* | |
| | | *Sensing* | *touching %s?* | |
| | | *Control* | *create clone of %s, delete this clone, when I start as a clone* | Abstraction |
| | | *Events* | *when %s key pressed,* | Algorithms |
| | | *Control* | *repeat %s%s, repeat until %s%s* | |
| | | *Operator* | *%s or %s* | |
| 9.Breakout | In this game, the student must define a linear trajectory for a ball (*sprite*). This ball should bounce when touching an obstacle or when triggering the bar under the player's control. Through repetition loops and conditional structures, this activity controls variables and *sprites*. This exercise also involves the direct manipulation of the x- and y-coordinates of a *sprite*. | *Sprite* | creates five new | Modeling and Simulation |
| | | *Motion* | *go to %s, if on edge bound* | |
| | | *Looks* | *next costume* | |
| | | *Events* | *when I receive %s, broadcast %s* | |
| | | *Sensing* | *touching %s?* | |
| | | *Data* | creates two variables, *change x by %s* | Data Representation |
| | | *Events* | *when @greenFlag clicked, when %s key pressed* | Algorithms |
| | | *Control* | *if %s then %s, repeat until %s%s* | |
| | | *Operator* | *%s < %s* | |
| 10.Pacman | It is a combination of all presented concepts. This activity includes the following: controlled *sprite* displacement; use of a mechanism to prevent the Pacman from changing direction when there is a labyrinth wall; constant update of four variables; and distance calculation (red ghost x Pacman) to guide the direction of the ghost's shift. | *Sprite* | creates three new | Modeling and Simulation |
| | | *Motion* | *go to %s, move %s steps, turn @turnLeft %s degrees,*<br>*turn @turnRight %s degrees* | |
| | | *Events* | *when I receive %s, broadcast %s* | |
| | | *Pen* | *pen down, pen up* | |
| | | *Data* | creates four variables, *change x by %s* | Data Representation |
| | | *Events* | *when @greenFlag clicked, when %s key pressed* | Algorithms |
| | | *Sensing* | *touching %s?* | |
| | | *Control* | *forever%s, repeat until %s%s, if %s then %s* | |
| | | *Operator* | *%s = %s, %s and %s* | |

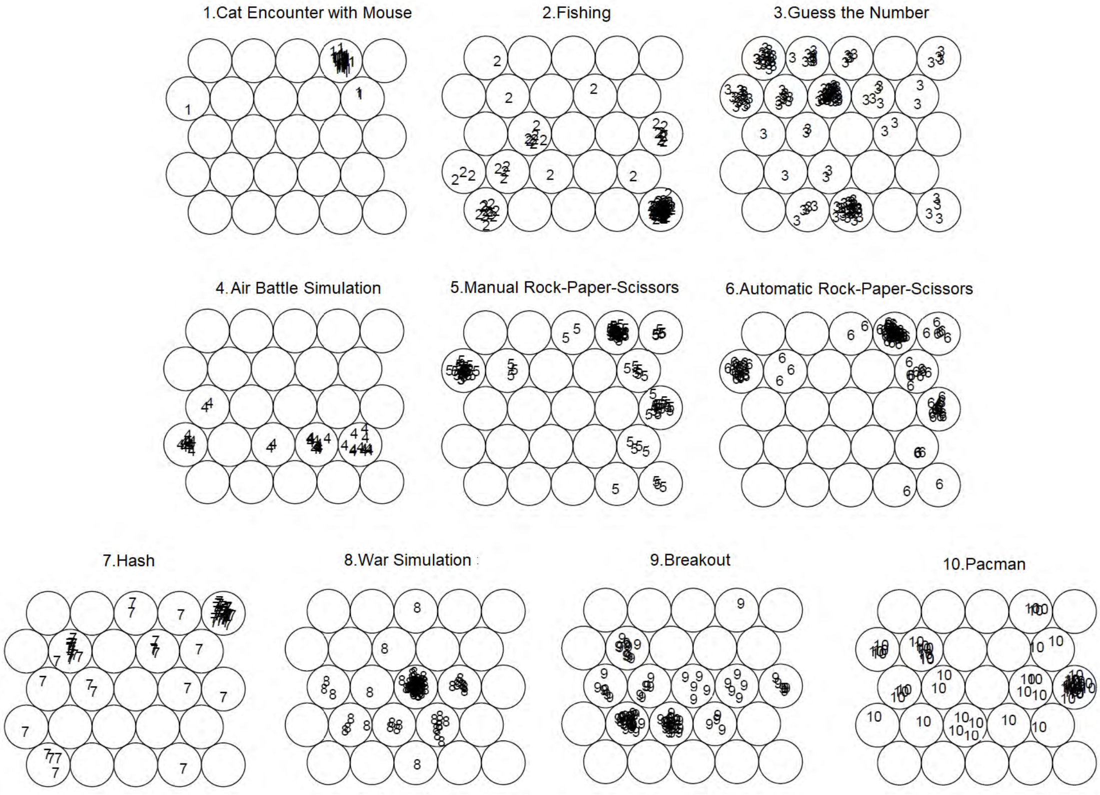

**FIGURE 9.** **Phase 2 map with each workshop game.**

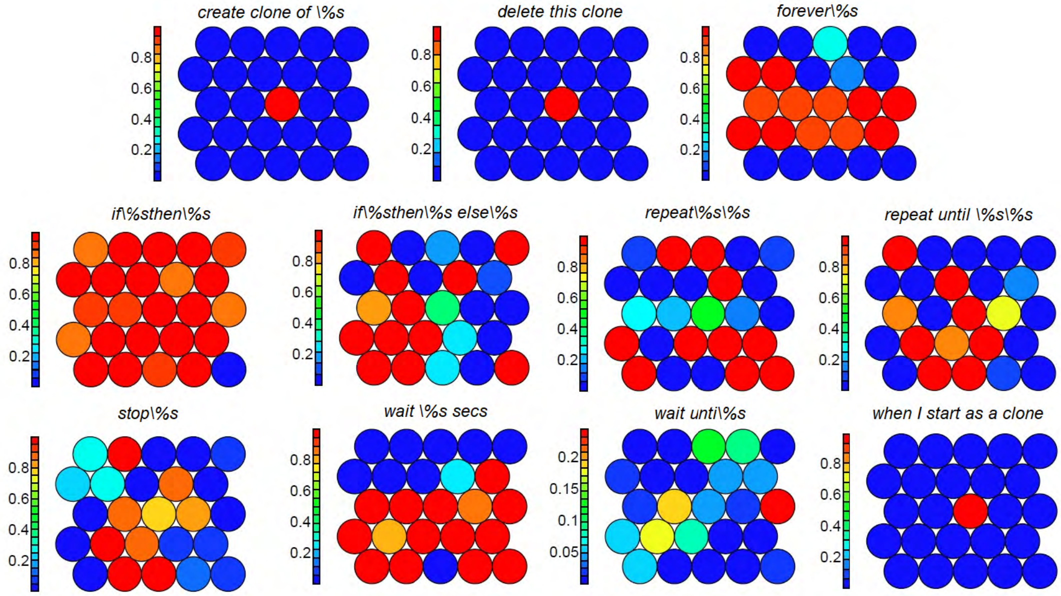

**FIGURE 10.** **Heat map of commands *Control* categories.**

of the games workshop for mapping the code instructions of the *Control* category.

To verify the frequency of category commands under study, the heat map was utilized for each instruction used in the offering of the discipline, as shown in Figure 10.

Analyzing Figures 9, 10 and Table 7 (the command structures of the Scratch environment) can identify whether the code instructions in the *Control* category were defined for each game and, consequently, the Computational Thinking skills and competencies attached to them (''Abstraction'' and ''Algorithms'') that were utilized during the workshop.

Taking again as an example for this analysis the game 5. Manual Rock-Paper-Scissors and its rubric presented in Table 7 where, in the *Control* category, the commands used

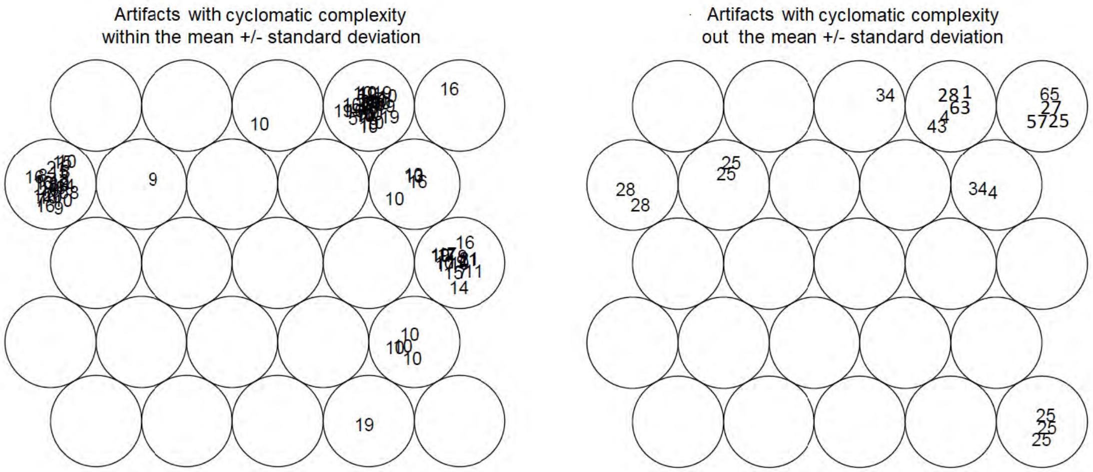

**FIGURE 11.** Phase 2 map with cyclomatic complexity of the game 5. Manual Rock-Paper-Scissors artifacts.

were *forever%s*, *wait unti%s* and *if%sthen%s*, as related to the development of Computational Thinking and competence regarding "Algorithms", the following is identified:

- three artifacts of 5. Manual Rock-Paper-Scissors game are located on the last neuron present on the right side of the map's top row map, where the weight in the feature related to the command *if%sthen%s* does not exist. However, it has a large concentration of *if%sthen%s else%s*, indicating that this command was utilized by the students to solve the problem proposed.
- neurons located in the upper and lower right region of the map have no weight in the *forever%s* feature, but there is a large concentration of the game artifacts. In this area, the textit *repeat%s%s* and *repeat until %s%s* commands have a presence, though discrete. This case directs the teacher to consider that some of the students were using another format of *loop* to solve the proposed problem, which may show a deviation in the workshop operation process.
- in relation to the command *wait unti%s*, no other type of behavior was found than expected.

In light of the above, it is evident that throughout the construction of game 5. Manual Rock-Paper-Scissors, some commands that were designed to be developed by the teaching methodology were not used. Considering that the method of teaching adopted poses challenges of increasing complexity, this aspect can indicate a gap in the strategy that directs the acquisition of knowledge, but precisely in the competence shown regarding "Algorithms" in Computational Thinking.

Another consideration is related to the commands *create clone of %s*, *delete this clone* and *when I start as a clone*, which are present just in 4. Air Battle Simulation, to develop the "Abstraction" competence. It was identified that only part of this game's artifacts that were produced have these commands, leading us to infer that this ability may not have been developed by all students.

### 3) THIRD INFORMATION NEED

To meet the third information need from the Phase 2 cluster analysis "Discover which Scratch commands are influencing the calculation of the cyclomatic complexity and what the effect is on the execution of the workshop (applies only to the *Control* category)", this need was examined again in game 5. Manual Rock-Paper-Scissors.

Before analysis of the topological distribution of this game's cyclomatic complexity, the average and standard deviation of the value of this metric was calculated in the game artifacts. The purpose was to identify which are above and below the established limit to make two plots: one with the topological distribution of the cyclomatic complexity of the artifacts within the boundary (mean +/− standard deviation of the *Complexity* variable of Table 5) and another above and below the limit, as shown in Figure 11. In this way, it is possible to isolate the artifacts that have discrepant values, from both higher and lower cyclomatic complexity, by projecting these cases on the map of the topological distribution, as shown in Figure 11.

It is possible to identify neurons containing only artifacts with values within the mean and standard deviation and others only with discrepants (lower region of the map). However, the neurons present in the two upper region lines of the map have artifacts in both cases. This is because clustering considers the predominance of commands, not quantity. However, the presence of artifacts with discrepant values in these clusters indicates, directly, that they have some anomaly in the structure of their code.

Analyzing the topological distribution of the cyclomatic complexity values presented in Figure 11, it was possible to identify neurons containing only artifacts with values within the mean and standard deviation and others only with discrepants (lower region of the map). Nevertheless, the neurons present in the two upper region lines of the map have artifacts in both cases. This was because clustering considers the predominance of commands, not quantity. However, this scenario indicates that the codes may contain some anomaly in their structure. Examining the artifacts with values above the mean/standard deviation was identified:

- the presence of *if%sthen%s*, *if%sthen%s else%s*, *repeat until %s%s* and *wait unti%s* commands in *dead code* blocks.
- use of duplicate code to solve the problem of costumes change of the *Sprite* proposed by the mechanics of the game.
- excessive use of commands, especially *if%sthen%s*.

On the other hand, the artifacts with values below the mean/standard deviation were found to have no executable code to meet the proposed problem, providing evidence that the student did not perform the required activity.

In general, the following is emphasized: the ability to use SOM in different analyses, including scenarios of phased approaches; the selection of the maps of this proposal that allow evaluation of the incremental development of Computational Thinking skills and competencies; and finally, discoveries that allowed identifying the students' difficulties in the elaboration of each game of the workshop, addressing adjustments in future.

## V. DISCUSSION

This research was motivated by the need to identify the evolution of the acquisition of Computational Thinking, taking into account all the aspects involved in the learning process that address the complexity of the artifacts. In light of the above, a framework for educational data mining is presented, containing a structure based on views that direct the activity of measuring skills, and the use of the SOM network for cluster analysis and identification of patterns and behaviors. The framework usefulness was evaluated, and data generated using the Scratch code artifacts produced in 4 applications of a digital games workshop were used. The objective was to verify its viability through the results obtained, aiming to meet the objectives that guide this research.

Regarding the first objective of the research, ''To detect patterns of students that show the evolution of Computational Thinking skills and competencies along with the participation in classes of construction of digital games'', the result of the application of the process and other proposed views in the framework made it possible to identify the realization of the strategy regarding activities of increasing complexity in the development of Computational Thinking skills and competencies. It also allowed verifying how the Scratch categories environment was defined for each game, and how the commands that constitute them and the Computational Thinking skills and competencies were associated and developed throughout the workshop. In this way, it was possible to indicate the adherence, the deviations and the attention points linked to the teaching rubric.

To meet the second objective of this study, ''To evaluate the practical effect of the teaching methodology adopted for the exposition of Computational Thinking skills and competencies during the workshop application'', the framework application result discovered a relation between the weights of the growth features of the clustering and the trial of games followed by the workshop. This allows knowing the categories, structures, and commands of the Scratch environment (designed by the teaching methodology) that were truly used during the workshop by the students. It also enabled identifying deviations in the artifacts elaboration, such as the *dead code* presence and repeated multiple codes, indicating a possible irregularity in the execution of the workshop or the method of teaching adopted. The advantage of visualizing the SOM maps allowed the detailing of this behavior, identifying which games had the highest incidence of these deviations. This approach enables the teacher to further enhance the adopted digital games workshop model. In the same way that the visual exploratory analysis of weights of each neuron in the map allows identifying the improper use of commands, it is also possible to detect positive scenarios of command usage, since the weight of the features vector will also be different. In this way, it is possible to identify disparate solutions carried out by the students. Although this scenario was not observed in the case study, this type of discovery may be addressed in further works. The obtained results also present the cyclomatic complexity metric as a support tool to evaluate the practical effect of the teaching method adopted. By indicating the high commands use, in this specific case of the *Control* category of the Scratch environment, it helps the teacher to identify programming ''habits'' acquired by the students throughout the application of the digital games workshop.

Specifically, on the Kohonen SOM map, the results show that, considered in a joint manner, the analysis of the weights of the features that compose each neuron, the topological distribution of the elements of the data set implemented in the map (including the student's personal identification, if necessary), and the advantage of visualization provided by this neural network addressed the specific objectives of the study.

## VI. CONCLUSION

This article presents a framework for educational data mining, based on the use of the SOM network for the analysis of groups and identification of patterns and behavior that address the evolution of the acquisition of Computational Thinking. A case study was carried out using Scratch code artifacts produced in a Digital Games Building Workshop, structured with didactic activities of increasing complexity and with a minimum duration of 12 weeks. The analysis of the data presented in the artifacts allowed identifying how the Computational Thinking skills and competencies were being developed based on the pattern of problem-solving adopted by the students and presented in the generated code structure.

The main findings of the research up to that point indicate the feasibility of the proposed data mining framework. Its structure of 4 views (teaching learning view, structural view, cluster analysis view, and process view) guided by information needs, the practices adopted (the PSM model and the cyclomatic complexity of software engineering metric) and the analysis of clusters through the SOM neural network provided the fulfillment of the specific established objectives: (1) detecting student patterns that point to the evolution of Computational Thinking skills and competencies throughout participation in a game building workshop; and (2) evaluating the practical effect of the teaching methodology adopted for the exposition of Computational Thinking skills and competencies during the workshop application.

Some limitations to the validity of the study should be emphasized, such as the complexity present in the use of the SOM algorithm and the interpretation of the results generated, such that its use may be limited to teachers with experience in the area of data mining. However, it is possible to perform the automation of the evaluation process, creating an analysis guide to help the teacher in the application of the said workshop.

Considering the potential and limitations of the present study, the following lines are proposed as future work: (i) to incorporate other software engineering metrics into the mapping process that, similar to cyclomatic complexity, can be used as support tools to identify the evolution of the acquisition of Computational Thinking; (ii) to propose the implementation of a tool to automate part of the proposed process in the framework; (iii) to define a *learning analytics* method to evaluate the Computational Thinking skills and competencies in students based on the knowledge acquired through the application of the proposed educational data mining framework; and (iv) to adopt the framework in the analysis of teaching activities that use distinct approaches compared to the one used in this case study (e.g. gamification techniques [45], [46], unplugged activities [21], among others) in which a teaching rubric of Computational Thinking skills and competences can defined, as well as tangible evidence can be collected throughout its execution, such as narrative texts, robotic artifacts, among others [47].

## REFERENCES

[1] V. J. Shute, C. Sun, and J. Asbell-Clarke, "Demystifying computational thinking," *Educ. Res. Rev.*, vol. 22, pp. 142–158, Nov. 2017.

[2] D. Seehorn, S. Carey, B. Fuschetto, I. Lee, D. Moix, D. O'Grady-Cunniff, B. B. Owens, C. Stephenson, and A. Verno, "CSTA K–12 computer science standards: Revised 2011," Comput. Sci. Teachers Assoc., New York, NY, USA, Tech. Rep. 104111, 2011.

[3] F. J. García-Peñalvo, "Editorial computational thinking," *IEEE Rev. Iberoam. Tecnol. Aprendizaje*, vol. 13, no. 1, pp. 17–19, Feb. 2018.

[4] J. M. Wing, "Computational thinking," *Commun. ACM*, vol. 49, no. 3, pp. 33–35, 2006.

[5] T. S. Barcelos, G. C. Costa, R. Munoz, and I. Silveira, "Informal HCI: Fixing playability issues as a strategy to improve the skills of novice programmers," *IEEE Latin America Trans.*, vol. 12, no. 1, pp. 29–35, 2014.

[6] G. Costa, T. Barcelos, C. Oliveira, R. Muñoz, R. Nöel, and I. Silveira, "Construindo jogabilidade: Como a percepção dos jogadores afeta o desenvolvimento de jogos em um contexto escolar," in *Proc. 7th SBGames*, 2013, pp. 16–18.

[7] R. Munoz, R. Villarroel, T. S. Barcelos, F. Riquelme, A. Quezada, and P. Bustos-Valenzuela, "Developing computational thinking skills in adolescents with autism spectrum disorder through digital game programming," *IEEE Access*, vol. 6, pp. 63880–63889, 2018.

[8] M. Resnick, J. Maloney, A. Monroy-Hernández, N. Rusk, E. Eastmond, K. Brennan, A. Millner, E. Rosenbaum, J. Silver, B. Silverman, and Y. Kafai, "Scratch: Programming for all," *Commun. ACM*, vol. 52, no. 11, pp. 60–67, 2009.

[9] E. B. Costa, B. Fonseca, M. A. Santana, F. F. de Araújo, and J. Rego, "Evaluating the effectiveness of educational data mining techniques for early prediction of students' academic failure in introductory programming courses," *Comput. Hum. Behav.*, vol. 73, pp. 247–256, Aug. 2017.

[10] A. Souza, T. Barcelos, R. Munoz, I. Silveira, N. Omar, and L. Silva, "Self-organizing maps to find computational thinking features in a game building workshop," in *Proc. 43rd Latin Amer. Comput. Conf. (CLEI)*, 2017, pp. 1–8.

[11] T. Barcelos, A. Souza, L. Silva, R. Muñoz, and R. V. Acevedo, "Mensurando o desenvolvimento do pensamento computacional por meio de mapas auto-organizáveis: Um estudo preliminar em uma oficina de jogos digitais," in *Anais dos Workshops do Congresso Brasileiro de Informática na Educação*, vol. 6, no. 1, p. 932, 2017.

[12] A. L. Araujo, W. Andrade, and D. Guerrero, "Um mapeamento sistemático sobre a avaliação do pensamento computacional no Brasil," in *Proc. Anais dos Workshops do Congresso Brasileiro de Informática na Educação*, vol. 5, no. 1, p. 1147, 2016.

[13] U. Wolz, C. Hallberg, and B. Taylor, "Scrape: A tool for visualizing the code of scratch programs," presented at the 42nd ACM Tech. Symp. Comput. Sci. Educ., Dallas, TX, USA, 2011.

[14] B. Boe, C. Hill, M. Len, G. Dreschler, P. Conrad, and D. Franklin, "Hairball lint-inspired static analysis of scratch projects," in *Proc. 44th ACM Tech. Symp. Comput. Sci. Educ. (SIGCSE)*, New York, NY, USA, 2013, pp. 215–220. doi: 10.1145/2445196.2445265.

[15] J. Moreno-León, G. Robles, and M. Román-González, "Dr. scratch: Automatic analysis of scratch projects to assess and foster computational thinking," *Revista de Educación a Distancia*, vol. 45, no. 46, pp. 1–23, 2015.

[16] J. M. León, G. Robles, and M. Román-González, "Comparing computational thinking development assessment scores with software complexity metrics," in *Proc. IEEE Global Eng. Edu. Conf. (EDUCON)*, Apr. 2016, pp. 1040–1045.

[17] K. Brennan and M. Resnick, "New frameworks for studying and assessing the development of computational thinking," in *Proc. Annu. Meeting Amer. Educ. Res. Assoc.*, Vancouver, BC, Canada, vol. 1, 2012, p. 25.

[18] M. Bienkowski, E. Snow, D. Rutstein, and S. Grover, "Assessment design patterns for computational thinking practices in secondary computer science: A first look," SRI Int., Washington, DC, USA, Tech. Rep., 2015.

[19] P. Techapalokul, "Sniffing through millions of blocks for bad smells," in *Proc. ACM SIGCSE Tech. Symp. Comput. Sci. Educ.*, 2017, pp. 781–782.

[20] D. A. Fields, Y. B. Kafai, and M. T. Giang, "Youth computational participation in the wild: Understanding experience and equity in participating and programming in the online scratch community," *ACM Trans. Comput. Educ.*, vol. 17, no. 3, p. 15, 2017.

[21] S. Grover, S. Basu, and P. Schank, "What we can learn about student learning from open-ended programming projects in middle school computer science," in *Proc. 49th ACM Tech. Symp. Comput. Sci. Educ.*, 2018, pp. 999–1004.

[22] S. Basu, "Using rubrics integrating design and coding to assess middle school students,"open-ended block-based programming projects," in *Proc. 50th ACM Tech. Symp. Comput. Sci. Educ.*, 2019, pp. 1211–1217.

[23] C. Avila, S. Cavalheiro, A. Bordini, M. Marques, M. Cardoso, and G. Feijo, "Metodologias de avaliação do pensamento computacional: Uma revisão sistemática," in *Proc. Brazilian Symp. Comput. Educ. (Simpósio Brasileiro de Informática na Educação-SBIE)*, 2017, vol. 28, no. 1, p. 113.

[24] T. Kohonen, "Essentials of the self-organizing map," *Neural Netw.*, vol. 37, pp. 52–65, Jan. 2013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0893608012002596

[25] S. F. Browning, "Using Dr. Scratch as a formative feedback tool to assess computational thinking," M.S. thesis, School Educ., Brigham Young Univ., Provo, UT, USA, 2017.

[26] G. Robles, J. Moreno-León, E. Aivaloglou, and F. Hermans, "Software clones in scratch projects: On the presence of copy-and-paste in computational thinking learning," in *Proc. IEEE 11th Int. Workshop Softw. Clones (IWSC)*, Feb. 2017, pp. 1–7.

[27] R. Ribeiro, T. Barcelos, A. Souza, and L. A. Silva, "Mensurando o desenvolvimento do pensamento computacional por meio de mapas auto-organizáveis: Comparação de métricas de complexidade de software com Dr. Scratch e ct-test," in *Proc. Anais dos Workshops do Congresso Brasileiro de Informática na Educação*, 2018, vol. 7, no. 1, p. 609.

[28] T. Barcelos, L. F. S. do Nascimento, A. Souza, L. Silva, and R. Munoz, "Análise automatizada do discurso de aprendizes de programação: Relações entre emoções e nível de experiência," in *Proc. Anais dos Workshops do Congresso Brasileiro de Inf. na Educacao*, 2017, vol. 6, no. 1, p. 1202.

[29] S. Grover, S. Basu, M. Bienkowski, M. Eagle, N. Diana, and J. Stamper, "A framework for using hypothesis-driven approaches to support data-driven learning analytics in measuring computational thinking in block-based programming environments," *ACM Trans. Comput. Edu.*, vol. 17, no. 3, p. 14, 2017.

[30] J. Moreno-León, G. Robles, and M. Román-González, "Towards data-driven learning paths to develop computational thinking with scratch," *IEEE Trans. Emerging Topics Comput.*, to be published.

[31] S. R. Pressman, *Software Engineering: A Practitioner's Approach*. New York, NY, USA: McGraw-Hill, 2016.

[32] P. B. Kruchten, "The 4+1 view model of architecture," *IEEE Softw.*, vol. 12, no. 6, pp. 42–50, Nov. 1995.

[33] A. L'heureux, K. Grolinger, H. F. Elyamany, and M. A. Capretz, "Machine learning with big data: Challenges and approaches," *IEEE Access*, vol. 5, pp. 777–797, 2017.

[34] J. McGarry, *Practical Software Measurement: Objective Information for Decision Makers*. Reading, MA, USA: Addison-Wesley, 2002.

[35] P. Kouontchou, A. Lendasse, Y. Miche, A. Modesto, P. Sarlin, and B. Maillet, "A R-SOM analysis of the link between financial market conditions and a systemic risk index based on ICA-factors of systemic risk measures," in *Proc. 49th Hawaii Int. Conf. Syst. Sci. (HICSS)*, Jan. 2016, pp. 1759–1770.

[36] A. H. Osman and A. A. Alzahrani, "New approach for automated epileptic disease diagnosis using an integrated self-organization map and radial basis function neural network algorithm," *IEEE Access*, vol. 7, pp. 4741–4747, 2019.

[37] Y. Arima and A. Hirose, "Performance dependence on system parameters in millimeter-wave active imaging based on complex-valued neural networks to classify complex texture," *IEEE Access*, vol. 5, pp. 22927–22939, 2017.

[38] U. F. Alias, N. B. Ahmad, and S. Hasan, "Mining of e-learning behavior using SOM clustering," in *Proc. 6th ICT Int. Student Project Conf. (ICT-ISPC)*, May 2017, pp. 1–4.

[39] R. Wehrens and L. M. C. Buydens, "Self- and super-organizing maps in R: The kohonen package," *J. Stat. Softw.*, vol. 21, no. 5, pp. 1–19, 2007. [Online]. Available: http://www.jstatsoft.org/v21/i05

[40] S. Haykin, *Redes Neurais: Princípios e Prática*. Bookman, 2007.

[41] L. A. da Silva, S. M. Peres, and C. Boscarioli, *Introdução à Mineração de Dados: Com Aplicações em R*. Salvador, Brasil: Elsevier, 2017.

[42] R. Mu noz, T. Barcelos, R. V. Acevedo, and I. F. Silveira, "Diseño e implementación de un taller de programación de juegos digitales con scratch como apoyo a fundamentos de programación," in *Proc. Anais dos Workshops do Congresso Brasileiro de Informática na Educação*, 2015, vol. 4, no. 1, p. 1495.

[43] T. S. Barcelos, "Relações entre o pensamento computacional e a matemática em atividades didáticas de construção de jogos digitais," Ph.D. dissertation, Univ. Cruzeiro do Sul, Sao Paulo, Brazil, 2014.

[44] I. Sommerville, *Software Engineering*. Reading, MA, USA: Addison-Wesley, 2003.

[45] I. Kotini and S. Tzelepi, "A gamification-based framework for developing learning activities of computational thinking," in *Gamification in Education and Business*. Springer, 2015, pp. 219–252.

[46] J. Isaac and S. V. Babu, "Supporting computational thinking through gamification," in *Proc. IEEE Symp. 3D User Inter. (3DUI)*, Mar. 2016, pp. 245–246.

[47] D. Kotsopoulos, L. Floyd, S. Khan, I. N. Namukasa, S. Somanath, J. Weber, and C. Yiu, "A pedagogical framework for computational thinking," *Digit. Experiences Math. Edu.*, vol. 3, no. 2, pp. 154–171, 2017.

**ALEXANDRA A. DE SOUZA** received the technological graduation degree in data processing from the Faculdade de Tecnologia de São Paulo, and the master's degree in electrical engineering and digital systems from the University of São Paulo. She is currently pursuing the Ph.D. degree with the Electrical Engineering and Computing Program, Universidade Presbiteriana Mackenzie. Her thesis is about data mining to analyze the evolution of computational thinking skills. She is currently an Associate Professor with the Instituto Federal de Educação, Ciência e Tecnologia de São Paulo. Her research interests include computers and education, software engineering, data mining, and artificial neural networks.

**THIAGO S. BARCELOS** received the bachelor's and master's degrees in computer science from the University of São Paulo, in 2002 and 2005, respectively, and the Ph.D. degree in science and mathematics teaching from Cruzeiro do Sul University, São Paulo, in 2013. In his Ph.D. thesis, he investigated collaborative approaches based on building digital games for the development of computational thinking skills. He is currently an Associate Professor with the Instituto Federal de Educação, Ciência e Tecnologia de São Paulo. His research interests include computers and education, agile methods, and multimodal learning analytics. He is currently a member of the Brazilian Computer Society (CEIE-SBC). He is an Associate Editor of the *International Journal on Computational Thinking* (IJCThink).

**ROBERTO MUNOZ** received the master's degree in computer engineering, engineering science, and education, and the Ph.D. degree in computer engineering. He is currently an Associate Professor with the School of Informatics Engineering and also an Adjunct Researcher with the Center of Cognition and Language (CIDCL), Research and Development Center in Healthcare Engineering (CINGS), Universidad de Valparaíso. He has authored more than 70 scientific papers in refereed international conferences and highly reputed journals. His research interests include computers and education, multimodal learning analytics, human-computer interaction, and health informatics. He is currently an Editor of the *International Journal on Computational Thinking* (IJCThink).

**RODOLFO VILLARROEL** received the Ph.D. degree from the University of Castilla-La Mancha, Ciudad Real, Spain, in 2005, and the master's degree in computer science engineering from the Universidad Técnica Federico Santa María, Valparaíso, Chile, in 2000. He is currently an Associate Professor and a Researcher with the School of Computer Engineering, Pontificia Universidad Católica de Valparaíso, Chile. His research interests include software engineering, quality and information security, and computers and education.

**LEANDRO A. SILVA** is currently pursuing the Ph.D. degree in systems engineering from the School of Engineering, University of São Paulo, Brazil. He is currently Adjunct Professor with the School of Computing and Informatics, Universidade Presbiteriana Mackenzie. He is also a Computer Engineer. His research interests include artificial neural networks, pattern recognition, data mining, machine learning, and big data analytics.