Chapter 4
Performance on Software Architecture Design to Serious Games for Mobile Devices

Leticia Davila-Nicanor, Irene Aguilar Juarez, Joel Ayala de la Vega, Abraham Banda Madrid, and Sochitl Cruz López

Abstract *Proposal*: This proposal has considered techniques to improve the software architecture performance in serious games. To validate and quantify the design approach have integrated the software architecture evaluation by design quality attributes complexity and coupling.

Design: Memory data handling on mobile devices is limited; this situation affects efficiency and slows interaction mechanisms of learning environments. In the software process, design patterns are a technique to solve this problem; the use of these in software architecture allows for the improvement of the distribution of device resources: memory and fast processing set objects at runtime.

Findings: The proposal describes a technique to perform and validate the design architecture; the advantage of evaluating the system in early phases like design is cost reduction to remove defects and better the software performance.

Limitations: The presented work has focused on the construction and evaluation of the quality of the software system; however, the aspects of pedagogical evaluation belong to another study.

Practical implications and value addition: If software architecture design improves, then the learning process also improves. In order to better the performance design, Wrapper, Singleton, and MVC are implemented. Quality evaluation is through software architecture analysis through graph theory and software metrics, the metrics of the resulting system architecture. The dispersion diagram shows us an architecture with acceptable quality levels.

L. Davila-Nicanor (✉) · A. B. Madrid
Laboratorio de Evaluación y Calidad de Software, Centro Universitario UAEMex Valle de México, Boulevard Universitario s/n. Predio de San Javier, Atizapán de Zaragoza, Estado de México, Mexico
e-mail: ldavilan@uaemex.mx; abandam@uaemex.mx

I. A. Juarez · J. A. de la Vega · S. C. López
Centro Universitario UAEMex Texcoco, Av. Jardín Zumpango s/n. Fraccionamiento El Tejocote, Texcoco, Estado de México, Mexico
e-mail: iaguilarj@uaemex.mx; jayalad@uaemex.mx

K. M. L. Cooper, A. Bucchiarone (eds.), *Software Engineering for Games in Serious Contexts*, https://doi.org/10.1007/978-3-031-33338-5_4

Keywords Serious game · Architectural performance · Design patterns · Design software metrics

4.1 Introduction

In the new normality reached by the Covid-19 pandemic, virtual environments use serious games focused on education at all levels. Serious games have been applied in a wide range of scenarios to improve learning and reaction mechanisms, for instance, fires and earthquakes, and this approach has been shown to statistically contribute to better decision-making [1–5]. In education research, the positive impact of serious games on training and education has been studied in several research papers [6–10]. Nowadays, serious game use is important to improve learning. The studies analyzed by [7–9] show how serious games have increased learning in many areas like computer science.

In serious games development, the collaboration between areas of knowledge is intrinsic and pedagogical, and software engineering experts participate in the software development process. From the definition of requirements, the quality attributes are specified, and this depends on the purpose of the system to be developed. In serious games, the *performance* is related to the learning process [10–12]; if the game manages to keep the attention of the player, the learning process improves. However, if the game is slow, loses the score, and does not update efficiently, the player's attention is lost, and implicit gamification is affected, so the learning process cannot mature [13].

Programming object-oriented approach and design patterns are a resource-optimizing technique because through a holistic vision, the performance of applications is improved, optimizing resources, for instance memory. After all, only executed functions are loaded at runtime, which differs from other programming paradigms that load all the system functions without being sure they will be used [14–16]. This scheme is adequate in a dynamic and random context like serious games because it is possible to build runtime scenarios depending on player preferences and game context variables.

The proposed approach addresses the following four research questions as the main contributions:

Do the *design patterns* contribute to improving the performance of software architecture on mobile serious games?

How does the software development process integrate *design patterns* on software architecture in mobile serious games?

Do *design patterns* contribute to improving the quality attributes complexity and coupling on software architecture to serious games on mobile applications?

Are software design architecture's complexity and coupling valid to determine an indirect study of the efficiency of software for serious games on mobile devices?

Answers to these research questions will contribute to improving studies and software development techniques on the performance of design architecture in serious games. This proposal has considered techniques to improve the software architecture performance and consequently the learning process, so the Wrapper, Singleton, and MVC design software patterns are implemented. To validate and quantify the design approach have integrated the software architecture evaluation by design quality attributes complexity and coupling, so we have implemented an evaluation process for architectural design described in [17]. This approach is applied to the case study of professional-level subjects.

This chapter has been organized as follows: Sect. 4.2 is about the didactic methodology used for the design proposed. Section 4.3 depicts the works that relate to a recent review of the literature where the needs that serious games have about architectural design are highlighted, as well as the analysis of design metrics proposed to date. In Sect. 4.4, the proposal is developed integrating design patterns on software architecture and their evaluation. In Sect. 4.5, the discussion of the results is presented, and finally, in Sect. 4.6, the conclusions are presented.

4.2 Motivation and Research

The serious game is based on information theory [18], which is a model that explains learning as a model analogous to the information processing of a computer in which there are temporary and permanent information storage units, as well as devices to capture, search, produce, and transform information. Under this approach, learning is understood as the process of incorporating new learning into memory and recovering and using it.

According to Benzanilla, J. M. et al. [19], the structural components definition of a serious game for the design of the formal model is as follows:

Objectives: they must be clearly defined and known by the player. In the context of a serious educational game, the objectives will be explicit in the competitions performed.

Rules: This component will determine the order, rights, and responsibilities of the players, as well as the objectives to be met by each player to achieve the challenge they face.

Challenge: Determines when the game ends. The player will face problems related to learning data structures for which solutions will be sought and, once all is resolved, will face the challenge. The endgame criteria, both partial and general, will be specified in the learning outcomes for the proposed serious game.

Interaction: It is the component that arises from the mechanics and dynamics of the game, which will give rise to all the experiences that the player will enjoy. These will continually surface because of the game's immediate feedback, which will reflect evidence of progress toward the final challenge.

Gu, S. M. et al. [20] approach a system like a pair (U, A), where U = {x_1, x_2,..., x_n} is a finite and non-empty set of objects called the universe of discourse, which in the case of serious games can be taken as the rules of operation, and A = {a_1, a_2,..., a_m} is a non-empty finite set of attributes, which in this case are the challenges, so that a: U → It goes for any ∈ A. That is, a software system fulfills its purpose based on its attributes. In serious game software design, the biggest problem is focused on the scenarios set that can be expected in the scope of the system to establish them, which has to do with the number of variables that intervene in the context and the number of possibilities to whom it is addressed, which makes biggest option set, which also can only be specified until the moment when the players select their arguments. A multi-scale software system is needed to represent data sets with hierarchical-scale structures measured at different levels of granularity at which each challenge and player interacts.

4.3 Background and Recent Review of the Literature

Software architectures set the necessary components that a software system must have, based on its functional requirements. The main goal is to reach the best interaction between components, so software architecture has been defined as "the fundamental organization of a system, embodied in its components, their relationships to each other, and the environment, and the principles and guidelines governing its design and evolution over time" [21]. Non-functional requirements are also considered because they are related to quality attributes, for example, complexity, coupling, performance, reusability, cohesion, and reliability [22]. Regarding software architectures for the design of serious games, there are a few contributions. In Mizutani's research [23], the authors reviewed about 512 studies from 3 publishers of prestige, and under reuse criteria, they found only 36. This study found the approach based on *data-driven design* is, by a considerable margin, the most common, present in 45% of the studies. In minor frequency, other practices like *entities based on inheritance*, *layered systems*, and *design patterns* use are present in between 20% and 30% of the selected studies. The highlighted aspect's study is the relative absence of *test-driven development*. Also, *design patterns* are rarely seen applied in studies on the development of digital game mechanics. These patterns are currently a technique that has shown completeness and robustness in the domain of applications that implement them. The study authors expressed concern regarding what they consider to be a lack of community interest in software engineering to apply their knowledge to the context of serious game design, development, and testing and how much you could benefit from your asportation in these applications.

Regarding the performance of the application in the quality evaluation of the serious game, the metric reported in [6] work is the performance rate. Efficiency is a quality attribute related to the computer equipment's resources, memory, and processing speed, at runtime faster response times, are expected in software systems. It is an important quality attribute in serious games, this is because several studies

[10–12], there are set a direct relationship between attention and improvement in learning. On the contrary, a slow game generates disinterest and a lack of concentration, which inversely affects the learning process. Efficiency refers to the ability of a software system to do its job quickly and effectively, without consuming too many resources, such as memory or processor power. On the other hand, complexity refers to the number of components and the interconnection of these components in a system. Generally speaking, the greater the complexity of a software system, the more resources may be required to maintain its efficiency [24, 25]. For example, a simple software system that performs a specific task can be very efficient because it is easy to understand and maintain. But a complex software system that has many components and dependencies may be less efficient, as it may require more resources to maintain and function properly. In summary, while complexity is not an obstacle to efficiency, it can increase the need for resources and time to maintain the efficiency of a software system. Therefore, in the development of object-oriented software systems, it is important to find a balance between complexity and efficiency to develop a system that is effective and easy to maintain and uses available resources efficiently [26].

Design patterns are a good technique to solve the efficiency of software architecture [16]. The use of a design pattern abstracts and identifies key aspects in the solution of a highly complex problem. The kind of patterns is structural, behavioral, and creation, some of these are applied to set new functionality at runtime, having they limited only by the size of the memory of the equipment where they are executed. Researchers from [14] and [15] analyze how pattern design gives a better software solution in architecture to solve functionality. There are patterns to abstract and solve problems that are repeated daily in the design of software systems.

4.3.1 Metrics to Evaluate Architectural Software Design

To determine software architectural quality, software metric computation is widely used [27]. The evaluation of software architectures is a different process from the testing phase; the evaluation of the design involves data about the relations of the components, class, and the method's software system at rest, without system execution. In this case, the inputs are algorithms, class diagrams, diagrams of flow, etc. The advantage of evaluating the system in early phases like design is *cost reduction to remove defects*. According to the Carnegie and Mellon University study [28], if the evaluation is carried out in the testing phase, it is 12 times more expensive, but if software evaluation is done during start-up, it is 20 times more expensive than in the design phase.

There exist many metrics that measure the complexity of software: The cyclomatic complexity metric provides a means of quantifying intra-modular software complexity, and its utility has been suggested in the software development and testing process. This work [29] proposes to measure complexity, which is based on cyclomatic *complexity* and the concept of interaction between modules through

Table 4.1 Relationship between design attributes and software metrics

Study	Authors	Quality attributes	Metrics used
Investigating object-oriented design metrics to predict fault-proneness of software modules	Santosh Sigh Rathore, 2012 [31]	Size, cohesion, coupling, complexity, and inheritance	CBO (Coupling Between Objects), RFC (Response For a Class), LCOM (Lack of Cohesion in Methods), CAM, DIT (Depth of Inheritance Tree), NOC (Number of Children), LOC (Line Of Code), WMC (Weighted Methods per Class), CC (Cyclomatic Complexity).
Coupling and cohesion metrics in Java for adaptive reusability risk reduction	M.Iyapparaja, 2012 [32]	Cohesion and coupling	EV (Explicit dependence), IV (Implicit dependence)
The prediction design quality of object-oriented software using UML diagrams	Vibhash Yadav, 2013 [33]	Size, cohesion, coupling, complexity, inheritance, and abstraction	CC, LCOM, WMC, LOC
Predict fault-prone classes using the complexity of UML class diagram	Halim, 2013 [34]	Complexity	CC, RC (reduced complexity), NC (Nick's class)

coupling. In this article [30], multidimensional metrics are identified and defined; in this case, complexity is one of the properties that are related to performance and efficiency, applied to health monitoring models at the system level with specific phases of the design. Under these conditions, attributes like performance could be assessed indirectly through, for instance, complexity and *coupling* [29].

Design software metrics have been developed to obtain information about the quality design of an object-oriented software application, which aims to quantitatively describe a system's design properties. Table 4.1 shows the studies that address the design approach based on the design attributes studied, about the metrics evaluated. It is possible to observe that the most used properties or attributes are complexity, coupling, cohesion, and inheritance. The metrics CC [31–35], CBO [31, 32, 34, 35], and WMC [31–33] are the metrics with the highest level of acceptance for the realization of the studies, followed by LCOM and LOC [31, 33].

4.4 Proposed Solution

Benzanilla, J. M. et al. [19] set the structural components for the design of the formal model of the game, which are objectives, rules, challenges, and interactions. In this case, the biggest problem is that the set of scenarios that can be expected in the scope of the system, to establish the challenges, has to do with the number of variables that intervene in the context and the number of possibilities to whom it is addressed, which makes for an infinite set of options, which can also only be specified up to the point where players interact with the game.

4.4.1 Didactic Requirements

The application is aimed at computer engineering students to acquire knowledge of traversing trees in dynamic data structures, through practical exercises in which they will be graded according to the score acquired. Any engineering and computer science student who wants to reinforce their knowledge through this application. The player must have previous knowledge of basic programming.

Table 4.2 detailed the functional requirements, and the application consists of three levels of interaction: beginner, intermediate, and advanced. According to this level, the theoretical content is related. The first level is the beginner, and the problems that arise are a function of the theoretical framework that corresponds to this level. Scenarios (games) are established according to the answers if they are correct, and incentives are obtained, concluding the game, and it is assumed that he already has the master of the said topic, promoting level promotions.

Hardware specification for development: AMD 9 processor, 8G RAM, 500G hard drive, RANDOM 5 video card.

Table 4.2 Serious game functional requirements

No.	Functional requirements
R1	The main goal of the application is made to solve practical exercises
R2	The application will have content about the theoretical foundations
R3	The application will have content about the use of game explanation, rules, levels, and scenarios of the game
R4	The game consists of three levels (beginner, intermediate, and advanced)
R5	A new gamer will be at a beginner level
R6	Nothing is saved for the player in the test game, nor are incentives given
R7	To propose problems of the topic
R8	Rating of the solution to see the score and go to another level
R9	At each level of the game, the score acquired is saved
R10	The incentives are awarded according to the grade of the exercises on the topic

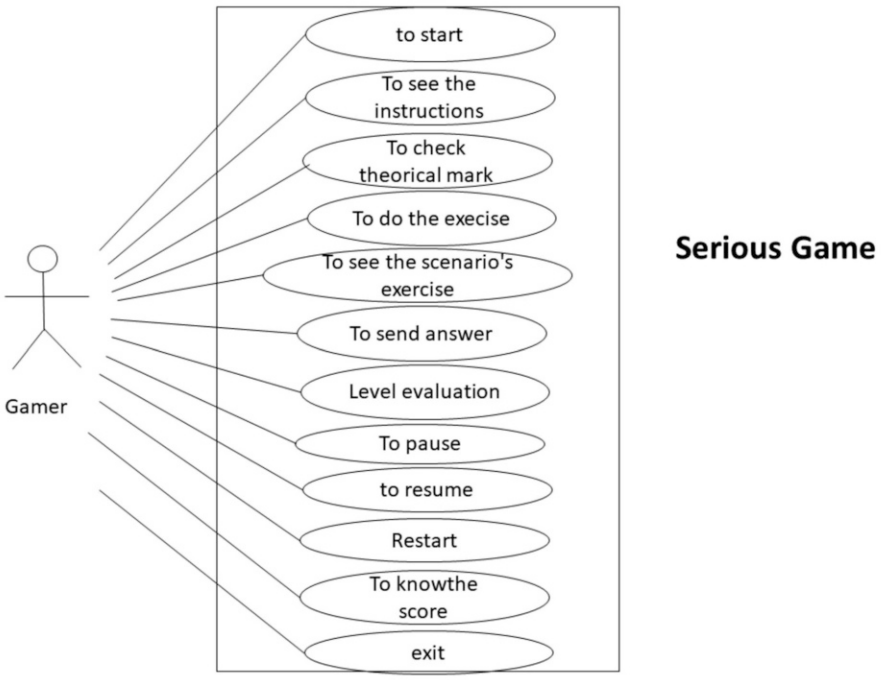

Fig. 4.1 Serious game functionality

Software specification for development: Windows 7, android-SDK-24-4-1-en-win, android-studio-ide-171.4443003-windows.

Specification for application: Have a smartphone that has an Android 4.2 (Jellybean) operating system, with 4G storage and 1G RAM.

The general functionality is shown in Fig. 4.1; the gamer (student) can select the play button, which shows him a screen where his initial score is presented, and the student can continue with the previous game or start a new game The serious game has three levels, before presenting the game scenario; first information about the topic to be evaluated is presented, and then the game scenario is presented, where several exercises accumulate points if they are solved satisfactorily. The player can pause the game, start a new game, or quit. In the case of just pausing the game, the accumulated points are saved. In other cases, the score is lost.

4.4.2 *Didactic Design*

In educational applications, defining the most appropriate pedagogical aspects for each project from the beginning is too important. In this case, the didactic design is specified by employing a descriptive letter of the didactic interaction in which the content of the game that is documented is specified. This application consists of three levels. In level 1 beginners, all the basic concepts of binary trees are addressed, which are necessary for the understanding of level 2 medium, where the path of binary trees is explained, so with this knowledge, you can go to level 3 advanced, and when you pass it, you can be evaluated as a student who masters the subject of binary trees in the data structure as shown in Table 4.3, the descriptive card of the game.

As a complement to the didactic letter, the navigation tree has been developed, using which it is possible to observe the proposed navigation for the player in the serious game. In Fig. 4.2, the player's activity in the three proposed levels is presented to accord letter of didactic interaction.

The serious game has been implemented in Android language. Design patterns are a technique that has demonstrated efficacy and reliability. The use of these in the software architecture allows for improvement of the distribution of device resources: memory and fast processing, the approach of setting set the needed objects at runtime. Design patterns have been successfully implemented in the architecture of our case studies; a serious game has been developed for the teaching of binary trees for the data structures subject of the computer engineering career. The Wrapper, Singleton, and MVC patterns are used in this work. They consider the functionality of the application. Better system evolution is another observed advantage of this approach; it is necessary throughout its useful life within the teaching-learning process.

In the proposed software architecture (Fig. 4.3), the *Wrapper pattern* allows to dynamically add functionality to the object to establish the scenarios, which allows only the base objects to be established in memory, and the pattern generates new combinations in the functionality in each new scenario so that the user has different views in each new game. Previous scenarios are dynamically removed from memory in this scheme. The *MVC pattern* (Model-View-Controller) allows for the separation of the operation of the user interface, the database, and the iteration between both; in our case, the database used was SQLite. The view shows the set of tools with which the player interacts, and the controller is responsible for communicating the *View_scenarios* actions and data where the game exercises are to the Wrapper pattern. Finally, certain global variables need to be kept in memory. When dealing with a dynamic schema, all the objects generated in memory will be eliminated except those that are handled by the singleton pattern. This scheme has allowed us to improve response times. The relationship of usability has been taken into account in studies that mark the most appropriate colors and texts [25].

When an object of type *Levels* is generated by the Exercise, in beginner level assigned zero points at the gamer through the *Assign_exercise*() method, according to Fig. 4.4. The gamer activity is evaluated by recording its responses, through the *Evaluate*() method, according to this action, the score is established, if the score's value is greater than 90%, a level rise can be granted through the *to_next_level*() method, this scheme operates when going from beginner to intermediate and from intermediate to advance via the *go_next_level*() method. When the player wants to pause, the *pause*() method is activated; if the gamer wants to leave temporarily and

Table 4.3 Descriptive letter of didactic interaction

Descriptive letter of didactic interaction					
Name of course: Data Structures		Duration: Variable		Modality: offline	
Instructor name:		Prerequisites: To the knowledge of basic programming			
Level 1 Beginner: Basic concepts of binary trees in data structure					
Topic	Content	Objective The participant:	Technique: Instructional	Activity: Instructor	Interaction: Mobile device
Rules game description	Game instructions	The student will know in a general way the composition of the game, the levels that it includes, and the learning that it will achieve	Through an explanation, the objectives of the game that will be used will be described to the student	The game allows interaction with the game rules description module	1. The student through the selection of options discovers the rules of the game 2. The application shows the academic content of each level and the challenges to solve 3. The student can listen to the general explanation of the game
Challenge Basic concepts	General concepts of trees: root, node, path, level, degree	Use the problem-based learning technique	The game will allow the student to listen to a reading of the basic concepts of binary trees The game provides questions related to the objectives of the problem	The application shows a reading with basic concepts	1. To show the student a fragmented image related to learning basic concepts 2. Randomly, the student is shown questions that, upon answering correctly, are given the corresponding image and he visualizes his puzzle little by little solved

Level 2 Intermediate: Binary tree traversal					
Rules' game presentation	Rules' game explanation	The student will know the rules of the game of level 2	Through the competence-based learning technique, through the explanation of the "rules of the game," the student will be able to solve the exercises by acquiring knowledge of the subject, based on their abilities, skills, and attitudes	The game provides audio where the rules of the game are explained	The game consists of an options menu where the player can select the instruction option where the student can listen to the general explanation of the game The game provides a video for each traversal where it explains the corresponding algorithm to traverse a binary tree
Binary tree traversal challenges	Tree traversal: Preorder, inorder, and postorder	The player will solve the binary tree traversal exercises provided by the application, thus being able to traverse the trajectory indicated in the game and go to the next level	It will be carried out with the competency-based learning technique	The game will provide a video with an explanation of each one of the routes that are carried out in the binary trees The application will provide the problems that the player will have to solve	The student will watch the video to learn how each of the routes of the tree is carried out, where each route that is correctly guessed will advance one square in the maze and will go to the next level once it reaches the goal square

(continued)

Table 4.3 (continued)

Descriptive letter of didactic interaction					
Name of course: Data Structures		Duration: Variable		Modality: offline	
Instructor name:		Prerequisites: To the knowledge of basic programming			
Level 3 Advanced: Balanced trees in data structures					
Know the rules of the game	Introduce the way of operating the game	The student will know the rules of the game	The discovery-based learning technique will be applied to solve the balanced tree exercises	The application will provide the rules of the game with audio explaining the goals that the player must meet as a measure of the knowledge obtained on the subject balanced trees in the data structure	The game describes the rules of the game
Balanced trees challenge	Balanced trees	The student will be able to acquire knowledge through the game; he will be able to discover the key and open the treasure chest, to solve the balanced tree exercises	Once the rules of the game are known, the learning technique based on discovery will be applied. The student must discover the knowledge and, in this way, guide him to the construction of his schemes. In this way, it organizes the information and relates it to previous knowledge	The game will provide a video of the academic knowledge of the subject so that the student can solve the exercises that the application will give him	The application shows a video with the content of the theme of the balanced tree; then it gives you the questions that you have to solve correctly to get a key; the more keys you have, the greater the probability of opening the treasure chest

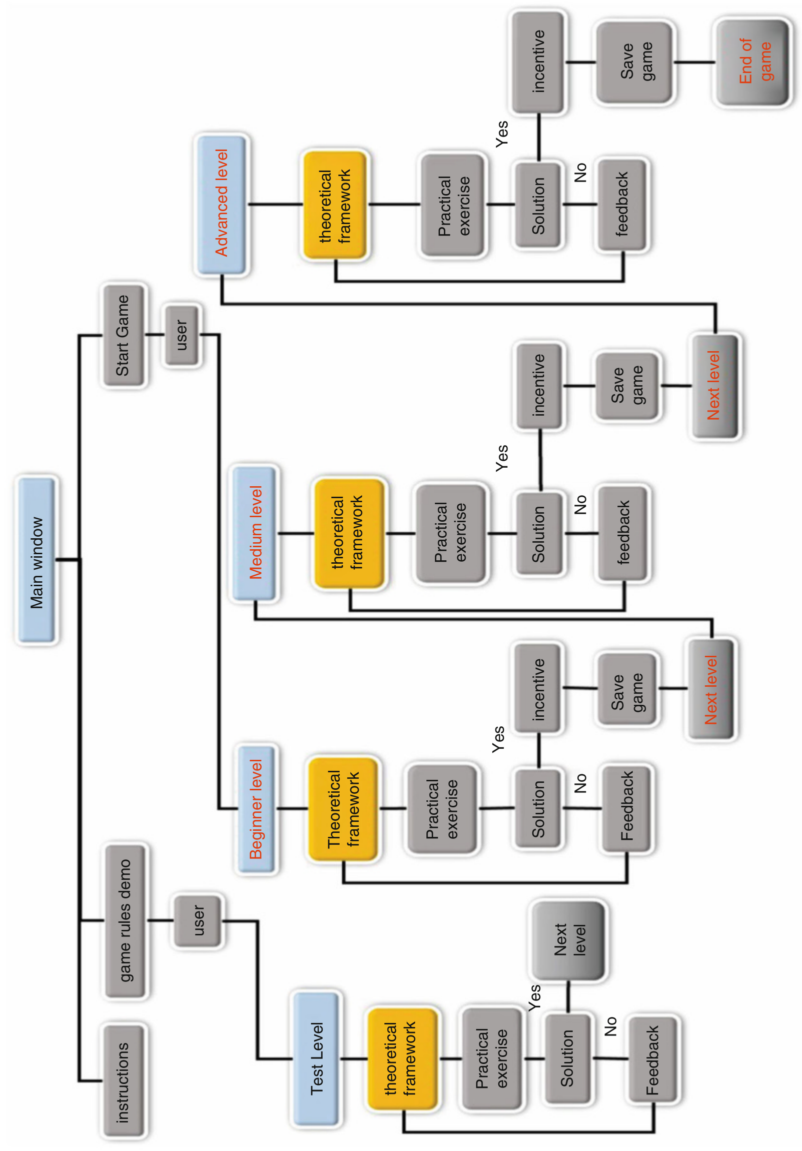

Fig. 4.2 Serious game navigation proposal

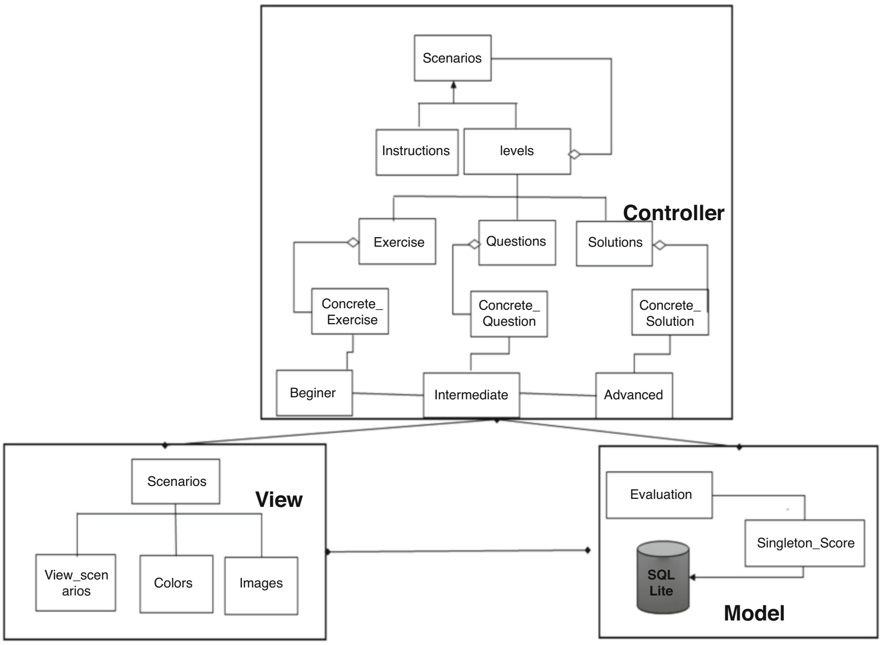

Fig. 4.3 Serious game proposal class diagram

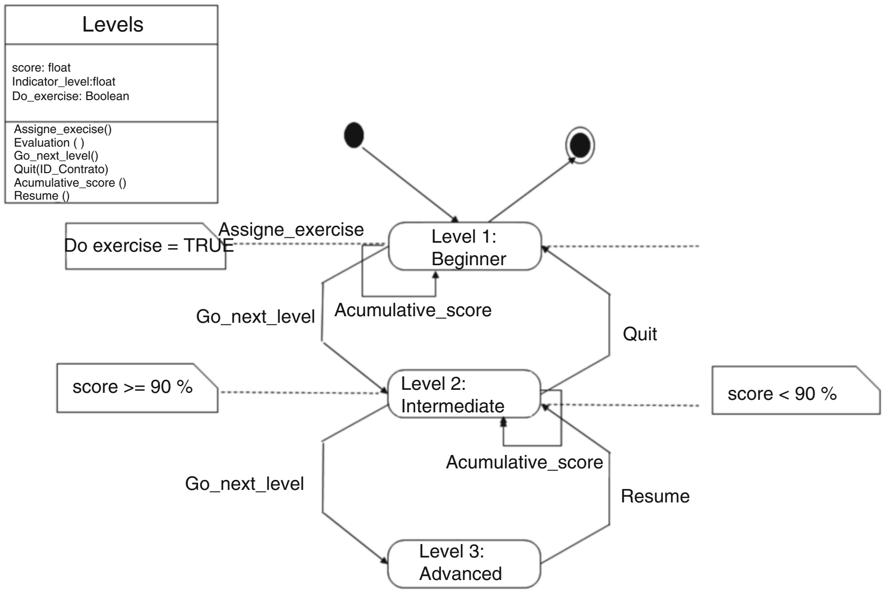

Fig. 4.4 State diagram levels class

then continue, his score is stored, and when he enters, he can *resume* from the level reached. Finally, if the gamer wants *to cancel* the game, the score is lost, and the game ends.

4.4.3 Serious Game Implementation

The serious game has been implemented in Android language. Figure 4.5 shows start-up interfaces and the beginner level. Regarding the usability of the game, the color chart has been selected that specifies the activity that motivates each of the colors according to [25]. For example, the blue color predominates, which generates calm and relaxation, which benefits the student to have concentration from the psychological didactic point of view. The green and yellow colors help the players in their retention and with greater ease, the theoretical contents of the game, to have more fluidity in the solution of the problems that each level of the game has.

Experimental Research on Design Evaluation
We have considered the proposal [17] to evaluate the quality architectural design of the proposed system. The proposal provides a predictive model to establish a quality scheme to assess the quality of oriented object systems based on the design stage using the software architecture analysis through graph theory and software metrics

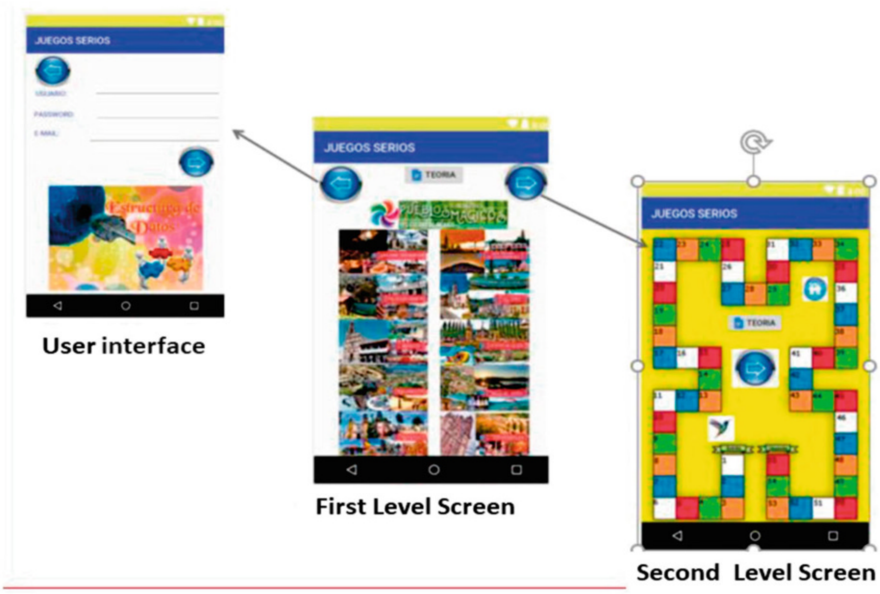

Fig. 4.5 Serious game interfaces

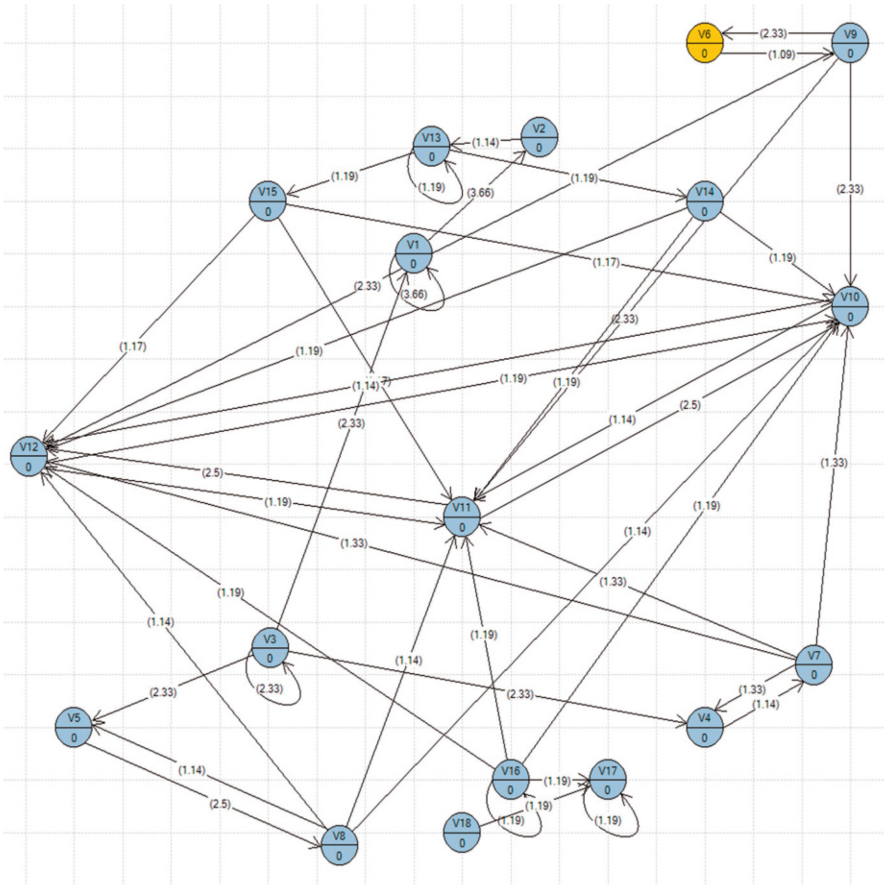

Fig. 4.6 ACG serious game proposal

of complexity and coupling. Based on the proposal, the following steps have been established:

Generation of the ACG. The architectural design of the system has been represented in Figs. 4.2, 4.3, and 4.4, so 18 components integrated the serious game, and the UML diagrams were the data gathering entry to establish the graph ACG$_{SG}$ (see Fig. 4.6). The relationship between the vertices of the graph and the components of the system is presented in Table 4.4. The first column has the name of the vertex that relates to the component, and in the following columns, they are the complexity and coupling components metrics. The plug-in CodePro Analytix [36] was used to get the CCM and CBO metrics on each component.

Setting the Complex attribute on ACG. The CCM metrics are ACG's weight, so the Floyd-Warshall algorithm was running, and 75 critical paths were estimated, thus the less complex paths are discovered at the beginning and the more complex

Table 4.4 The relationship between the vertexes of the ACG_{SG} and the components of the system

Vertex	Component	CCM	CBO
v1	Scenarios	3.66	2
v2	Instructions	1.14	3
v3	Levels	2.33	1
v4	Exercise	1.33	2
v5	Questions	2.5	1
v6	Solutions	1.03	2
v7	Concrete_Exercise	1.14	3
v8	Concrete_Question	1.14	3
v9	Concrete_Solution	2.33	1
v10	Beginner	1.14	3
v11	Intermediate	2.5	1
v12	Advanced	1.19	2
v13	Scenarios_view	1.29	3
v14	View_scenarios	1.19	2
v15	Colors	1.17	3
v16	Images	1.19	2
v17	Evaluation	1.19	2
v18	Singleton_Score	1.19	2

paths at the end, these are later, according to the projected functionality, that we designated critical paths, these are the ones that reflect the functionality of the system embedded in the entire design.

Setting the CBO metric on paths localized. Following the proposal, to set *quality factors* in critical paths localized, the Spearman correlation coefficient has been estimated with the CBO data group and CCM data group in each path localized previously. Some results of the paths identified are shown in Table 4.5; in the first column are presented the identification path; in the next column, the CCM and CBO metrics on the sequence are shown, and the last column is presented the quality factor results. To set quality factors, Spearman's correlation has evaluated the relationship between the CCM and the CBO as ordinal variables. The results obtained have shown spectrum values between -1 and $+1$. If the value closes to -1, it indicated a weak relationship between complexity and coupling, so it was set at a low-quality level on the path evaluated. The quality parameter values close to $+1$ have a stronger relation, and the quality is higher. The plot on the dispersion quality parameter of Fig. 4.7 shows us a dispersion set has a stronger relationship. From these results, it was possible to infer the quality of the system was acceptable according to the attributes that were evaluated.

In Table 4.5, the results are presented. The first column has the ID of the critical path, the second the components sequence and their CMM and CBO metrics values, and finally the last the quality parameter. In the table, only some results are presented as an example. The total results of the 71 critical paths are the input data of the scatter dispersion plot in Fig. 4.7.

Table 4.5 Set the critical paths evaluated through quality parameters on serious game

ID path	Sequence					Quality parameter
1	v1	v2	v13	v15	v10	1
CCM	3.66	1.14	1.29	1.17	1.14	
CBO	2	3	3	3	3	
2	v1	v2	v13	v15	v11	0.6488
CCM	3.66	1.14	1.29	1.17	2.5	
CBO	2	3	3	3	1	
3	v1	v2	v13	v15	v12	0.2236
CCM	3.66	1.14	1.29	1.17	1.19	
CBO	2	3	3	3	2	
4	v1	v2				0.8750
CCM	3.66	1.14				
CBO	2	3				
5	v1	v2	v13			0.6488
CCM	3.66	1.14	1.29			
CBO	2	3	3			
n	v1	...	...	...	vn	Quality_factor
CCM	...	...	...	...	...	
CBO	...	...	...	...	...	
71	v18	v17				1
CCM	1.19	1.19				
CBO	2	2				

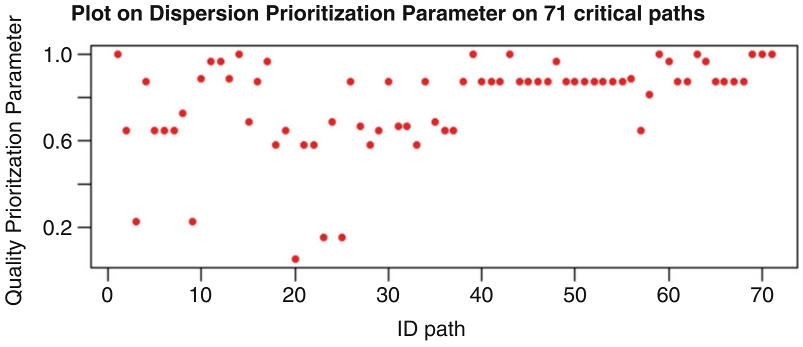

Fig. 4.7 Plot on dispersion quality parameter on serious game proposal

4.5 Discussion

In this proposal, the use of design patterns has been contemplated to improve the software architecture performance. Their holistic vision establishes an organization that requires optimization of the design, and the approach contemplates dynamic objects in memory generated at runtime [14–16]. This approach differs from others, where all the functions that the system can execute are loaded into memory, which in the case of serious games on mobile devices slows down the game dynamics.

To validate and quantify the design approach have integrated the software architecture evaluation method by design quality attributes complexity and coupling. The [17] proposal has been selected because the analysis has as its main axis a complexity approach strengthened with coupling. This relationship is related to efficiency [29–31], and the latter is another attribute that is desired in application performance. In the design stage, several studies are agreeing on the metrics Complexity [31–34], and Coupling [31, 32, 34, 35] are good options to estimate the quality and performance in oriented-object systems. In the design is not possible to carry out a dynamic evaluation, for instance on the testing, where it is possible because the code is available. The design doesn't have source code, and a static evaluation is made, taking the concept of concerns where the requirements are projected into the architecture of the system, and the approach is developed through UML diagrams, algorithms, and others [22]. The advantage of evaluating the design is cost reduction because it is 12 times lower than in the testing stage [28]. Another advantage is the defects localization and fixed before the code implementation, and finally, test prioritization is also available early, because the background is available to determine which components or their sequences that have a higher probability of failure.

The design architectural evaluation applied in this research provides a predictive model and formal method to assess a quality scheme on oriented object systems. According to the results obtained, it can be estimated that the quality parameter obtained through the evaluation model is based on the complexity and coupling attributes, so the architectural complexity graph (ACG) was established, through which a deterministic analysis is performed. This is the basis to set the quality parameter, and those component sequences with a quality parameter close to $+1$ have a stronger relationship between complexity and coupling attributes, also indicated at a good-quality level. With the implementation of the Wrapper and MVC design patterns in software architecture to the serious game, according to the scatterplot (Fig. 4.7), 86.6 percent of evaluated sequences close to $+1$, taking as an indicator the value of 0.5 of the dispensing graph, there are indicating a good quality level based on complexity and coupling quality design. The approach implemented to validate an indirect study of the efficiency of software design architectures using complexity and coupling of software for serious games is logically valid. Although the results are not enough to set quantified parameters for efficiency, it is necessary to estimate another analytical model to quantify the relationship.

4.6 Conclusions

The development of software for serious games in mobile applications implies a learning process like the process of incorporating new learning into memory, as well as retrieving and using it. This requires a software architecture designed to optimize memory and processing resources. The design patterns approach offers quality and performance. In the case of devices with limited storage, for instance, mobile phones, they allow the optimization of resources.

Answered the research questions, the *design patterns* contributed to improving the performance of software architecture on mobile serious games. The software development process to integrate *design patterns* on software architecture in mobile serious games involves an adequate abstraction of requirements employing a holistic vision projected in the design architecture, as well as the selection of quality attributes and design patterns that have the best performance. According to the purpose of the application that was developed at work, the Wrapper and MVC patterns focused on establishing scenarios at runtime. This approach optimizes the use of computing resources such as memory and processing. The process implemented to validate an indirect study of the efficiency of software design architectures through the complexity and coupling of software for serious games is logically valid. Although it is appropriate to determine an analytical model to quantify for efficiency, an additional study is required to complement the quantitative analysis to address the computational cost or to do traditional testing for this quality attribute.

As expressed by the study [23], our software engineering community must contribute studies and techniques to improve the performance of these applications from a formal point of view. The proposal describes a technique to perform and validate the design architecture. The advantage of evaluating the system in early phases like design is *cost reduction to remove defects* and better the software performance. The main limitation of the study is that there are attributes such as efficiency, which must be evaluated in phases after the design stage; however, validating that there is a good architectural design implies a better implementation and consequently a lower number of failures in its operation, which benefits the performance of the system.

The evaluation results set the pattern design implementation on design architecture to serious game software set a high quality related with Complexity and Coupling, allowing us to quantify the advantages of this approach which strengthens that the design techniques implemented are suitable in terms of software performance.

Acknowledgments This work was partially supported by the PNPC-CONACyT 2013–2016 and 2019 21 Master's thesis on Master of Degree Postgraduate of the Autonomous University of Mexico State.

Conflict of Interests We declare that we have no financial or personal conflicts of interest that could inappropriately influence the development of this research.

References

1. Gauthiera, A., Porayska-Pomsta, K., Mayer, S., et al.: Redesigning learning games for different learning contexts: applying a serious game design framework to redesign Stop & Think. Int. J. Child-Comput. Interact. **33**, 100503 (2022)
2. Daylamani-Zad, D., Spyridonis, F., Al-Khafaaji, K.: A framework and serious game for decision making in stressful situations; a fire evacuation scenario. Int. J. Hum.–Comput. Stud. **162**(2022), 102790 (2022)
3. Czauderna, A., Budke, A.: How digital strategy and management games can facilitate the practice of dynamic decision-making. Educ. Sci. **10**(4), 99 (2020)
4. Clark, E.M., Merrill, S.C., Trinity, L., Bucini, G., Cheney, N., Langle-Chimal, O., Shrum, T., Koliba, C., Zia, A., Smith, J.M.: Using experimental gaming simulations to elicit risk mitigation behavioral strategies for agricultural disease management. PLoS One. **15**(3), e0228983 (2020)
5. Mendonca, D., Beroggi, G.E., Van Gent, D., Wallace, W.A.: Designing gaming simulations for the assessment of group decision support systems in emergency response. Saf. Sci. **44**(6), 523–535 (2006)
6. Yamoul, S., Ouchaouka, L., Radid, M., Moussetad, M.: Implementing a serious game as a learner motivation tool, The 4th International Workshop of Innovation Technologies. Procedia Comput. Sci. **210**(2022), 351–357 (2022)
7. Jarnac de Freitas, M., Mira da Silva, M.: Systematic literature review about gamification in MOOCs. Open Learn. J. Open Distance e-Learn., 1–23 (2020)
8. Sailer, M., Homner, L.: The gamification of learning: a meta-analysis. Educ. Psychol. Rev. **32**(1), 77–112 (2020)
9. Connolly, T.M., Boyle, E.A., MacArthur, E., Hainey, T., Boyle, J.M.: A systematic literature review of empirical evidence on computer games and serious games. Comput. Educ. **59**(2), 661–686 (2012)
10. Kara, N.: A systematic review of the use of serious games in science education. Contemp. Educ. Technol. **13**, 2 (2021)
11. Graafland, M., Schraagen, J.M., Schijven, M.P.: Systematic review of serious games for medical education and surgical skills training. Br. J. Surg. **99**(10), 1322–1330 (2012)
12. Bellotti, F., Berta, R., De Gloria, A.: Designing effective serious games: opportunities and challenges for research. Int. J. Emerg. Technol. Learn. (2010)
13. Krath, J., Schürmann, L., von Korflesch, H.F.O.: Revealing the theoretical basis of gamification: a systematic review and analysis of theory in research on gamification, serious games, and game-based learning. Comput. Hum. Behav. **125**(2021), 106963 (2021)
14. Wedyan, F., Abufakher, S.: Impact of design patterns on software quality: a systematic literature review. IET Softw. **14**(1), 1–17 (2021)
15. Fletcher, J., Cleland-Huang, J.: Soft goal traceability patterns. In: 17th International Symposium on Software Reliability Engineering (ISSRE'06), pp. 363–374. IEEE, Raleigh, NC (2006)
16. Gamma, E., Henry, R., Johnson, R., Vlissides, J.: Design Patterns. Elements of Reusable Object-Oriented Software. Addison-Wesley, New York, NY (1994)
17. Dávila-Nicanor, L., Orozco Aguirre, H.R., Quintana López, M., Banda Madrid, A.: Enhancement to test case prioritization through object-oriented software architectural design. In: 10th International Conference on Software Process Improvement (CIMPS) 2021, pp. 131–138 (2021)
18. Galvis Panqueva, A.H.: Ingeniería de Software Educativo. Ediciones Uniandes - Universidad de los Andes, Santafé de Bogotá, Colombia (1992)
19. Bezanilla, M.J., Arranz, S., Rayon, A., Rubio, I., Menchaca, I., Guenaga, M., Aguilar, E.: Propuesta de evaluación de competencias genéricas mediante un juego serio. New Approach. Educ. Res., 44–54 (2014)

20. Gu, S.M., Wu, Y., Wu, W.Z., Li, T.J.: Knowledge approximations in multi-scale ordered information systems. In: Rough Sets and Knowledge Technology, Shanghai, China, 2014, pp. 525–534 (2014)
21. IEEE Std 1471-2000: IEEE Recommended Practice for Architectural Description of Software-Intensive Systems. IEEE Computer Society (2000)
22. Oliver, V., Ingo, A., Arif, C.K.: Timo Software Architecture. Springer Nature, New York, NY (2011)
23. Mizutani, W.K., Daros, V.K., Kon, F.: Software architecture for digital game mechanics. Entertain. Comput. **38**, 100421 (2021)
24. Dzaferagic, M., Kaminski, N., Macaluso, I., Marchetti, N.: Relation between functional complexity, scalability, and energy efficiency in WSNs. In: 13th International Wireless Communications and Mobile Computing Conference (IWCMC), Valencia, Spain, 2017, pp. 675–680 (2017)
25. Bansiya, J., Davis, C.G.: A hierarchical model for object-oriented design quality assessment. IEEE Trans. Softw. Eng. **28**(1), 4–17 (2001)
26. Jayalath, T., Thelijjagoda, S.: A modified cognitive complexity metric to improve the readability of object-oriented software. In: International Research Conference on Smart Computing and Systems Engineering (SCSE), Colombo, Sri Lanka, 2020, pp. 37–44 (2020)
27. Arvanitou, E.M., Ampatzoglou, A., Chatzigeorgiou, A., Galster, M., Avgeriou, P.: A mapping study on design-time quality attributes and metrics. J. Syst. Softw. **127**, 52–77 (2017)
28. Park, R.E., Wolfhart, B.: Goal-Driven Software Measurement. A Guidebook. Software Engineering Institute, Carnegie Mellon University, Pittsburgh (1996)
29. Madi, A., Zein, O.K., Kadry, S.: On the improvement of cyclomatic complexity metric. Int. J. Softw. Eng. Appl. **7**(2) (2013)
30. Lewis, A.D., Groth, K.M.: Metrics for evaluating the performance of complex engineering system health monitoring models. Reliab. Eng. Syst. Saf. **223**, 108473 (2022)
31. Singh Rathore, S., Gupta, A.: Investigating object-oriented design metrics to predict fault-proneness of software modules. In: 2012 CSI Sixth International Conference on Software Engineering (CONSEG), Indore, India (2012)
32. Iyapparaja, M., Sureshkumar, D.: Coupling and cohesion metrics in java for adaptive reusability risk reduction. In: IET Chennai 3rd International on Sustainable Energy and Intelligent Systems (SEISCON 2012), Tiruchengode, India (2012)
33. Yadav, V., Singh, R.: The prediction design quality of object-oriented software using UML diagrams. In: 3rd International Advance Computing Conference (IACC), pp. 1462–1467, Ghaziabad, India (2013)
34. Halim, A.: Predict fault-prone classes using the complexity of UML class diagram. In: International Conference on Computer, Control, Informatics and Its Applications (IC3INA), Jakarta, Indonesia (2013)
35. Kumar, G.P., Joshi, G.: QMOOD metric sets to assess the quality of the java program. In: International Conference on Issues and Challenges in Intelligent Computing Techniques (ICICT), Ghaziabad, India (2014)
36. Google Developers 2018. https://sites.google.com/a/strategiesinsoftware.com/site/commentary/googlecodeproanalytix (2022). Accessed Aug 2022