Classutopia: A Serious Game for Conceptual Modeling Design

Felipe Larenas[a], Beatriz Marín[a], Giovanni Giachetti[b]
[a]Facultad de Ingeniería y Ciencias, Universidad Diego Portales, Chile
{felipe.larenas, beatriz.marin}@mail.udp.cl
[b]Universidad Tecnológica de Chile INACAP, Chile (ggiachetti@inacap.cl)

Abstract— **One of the more complex topics to teach to software engineering students is the conceptual modeling design, which has several concepts that students must learn in order to specify the structural, behavioral and interaction views of software systems. Learning the design of class diagrams is of paramount importance since these diagrams are used to guide concrete development tasks such as programming and software testing, and -consequently- to avoid defective software products. Applying novel teaching/learning techniques in this topic may help students to reduce the defects that are committed at the moment of designing a class diagram. One interesting technique is the use of serious games, due to the fact that they provide learning environments free of risks and pressure for students, allowing the students to know the topics that they must learn in a fun way. Serious games have been widely used in programming courses. Thus, we aim to investigate the feasibility to replicate this experience for conceptual modeling of class diagrams at software engineering courses. In this paper, we present a role-playing game specially focused in the class diagram, which is called *Classutopia*. This serious game provides modeling challenges, comprehension and correction of diagrams with different complexity levels for learning conceptual modeling design.**

Keywords— Gamification, serious games, class diagram, software engineering.

I. INTRODUCTION

Conceptual modeling is one of the more complex topics that must be learnt by software engineering students since they need to abstract concepts from reality and express them in computational terms. The conceptual modeling design is of paramount importance in the development of software projects because it specifies the different views of a software system, which guide the programming and the testing tasks. The class diagram is one of the most representative design approaches for software modeling [1], which serves as a guideline to develop the structures and the methods of a software product. A faulty class diagram misleads the software development teams, leading to mistakes in coding and delays in project implementations, and subsequently causing project failures.

Some common defects encountered in the development of class diagrams [2–5] are as follows: confusion amidst elements such as aggregation and composition; overspecification; errors in the cardinality, i.e., creating infinite recursive associations; poor choices of names for classes and attributes; class replications; lack of inheritance between classes of the same type; lack of classes, attributes, and lost methods.

The defects mentioned above can be overcome if the students actively participate in the learning process. The use of games and simulation-based experiences has been of great help for teaching and offering learning environments without any risk. These games are designed for a different purpose than to solely entertain and are known as serious games [6,7]. Moreover, the use of videogames characteristics such as points or rewards in a game is known as gamification. Currently, there are several systematic reviews in the literature [8,9] that have reported the positive results of serious game-based learning, wherein the use of this type of game has been verified in various domains of learning in topics related to health, education (school and university), culture, social skills, and vocational training, among others. Within the domain of education, specifically in the area of computing [10], it has been used to teach topics such as programming, project management software, and operating systems.

We believe that by implementing serious games, specifically focused on learning conceptual modeling, it is possible to motivate and entertain students by modifying their behavior in a positive manner when coping with class diagrams. Therefore, the design of a serious game for learning and understanding conceptual modeling design is presented in this paper. The game is a Role-Playing Game (RPG) wherein a character hero (characterized by a robot) is assigned to a player who must complete an adventure by overcoming three types of challenges to defeat the villains (characterized as evil wizards). As the game progresses, the difficulty of the challenges will increase based on the predefined problems.

Therefore, the contribution of this work is the design of a serious game for conceptual modeling learning of the class diagram and the implementation of a mobile application for Android that provides support for running the game on tablets. This contribution is useful for students that want to learn how to do a class model design avoiding defects, and it is also useful for researchers that want to add new challenges to the *Classutopia* serious game.

The rest of the article is organized as follows: Section 2 presents some relevant related works. Section 3 presents the design of a serious game developed for learning conceptual modeling of the class diagram. Section 4 presents the

DOI reference number: 10.18293/SEKE2018-145

evaluation of the game. Finally, Section 5 presents the main conclusions and future works.

II. Related work

Even though there are several serious games published in literature, as we can observe in the systematic literature reviews presented in [8,9], we did not find literature of serious games that had been applied to teaching/learning class diagram. Nevertheless, some of the games that are focused on teaching object-oriented programming concepts are closely linked to the modeling of classes, so that we review these documents in this section.

Java Ninja [11] evaluated the effects of utilizing serious games focused on teaching specific concepts of modeling and programming, wherein the students receive help in understanding the inheritance in object-oriented programming and show positive results in their learning processes and motivation.

In [12] a game-theme instructional model is presented, which demonstrates the concepts of object-oriented programming, specifically encapsulation, polymorphism, and inheritance. The game uses virtual-reality to engage students with 3 problems related to the programming topics mentioned above. The student must answer correctly in the inheritance situation in order to construct a class model diagram. Results indicate that 80% of students agreed that the module developed their interest in programming.

In [13] a game focused in object-oriented programming concepts is presented. In this game, the players can select an object and the class diagram is displayed in order to show the inheritance relationships of the selected object.

A serious game focused in the education of object-oriented analysis and design and artificial intelligence is presented in [14]. The game is about the Sokovan problem, where five types of relationships can be identified: association, composition, dependency, inheritance and aggregation. Authors state that to facilitate applying serious game a promising strategy would be accumulating and opening game-based materials. So that, they provides this game with a puzzle style, but no information is provided about the usage or empirical evaluations.

In summary, we didn't find serious games specifically focused in learning conceptual modeling design, nevertheless clear examples of the use of class diagrams in challenges of serious games have been obtained. Considering the importance and difficulty of conceptual modeling learning in software engineering, a serious game has been created specifically for class diagram learning, which is shown in the next section.

III. Design of Classutopia serious game

The design of a serious game for the conceptual modeling of class diagram learning has been developed using the engine Unity, aiming toward it being executed in a mobile application for devices with Android operating system. The conceptual framework of the MDA (Mechanics, Dynamics and Aesthetics) [15] is used to explain the design of the game, which allows the implementation of the game by taking into account the player and the developer viewpoints.

A. Aesthetics

For making the game attractive and fun for the player, the game has a minimalist graphic style, displaying easy-to-interpret buttons, lists, and menus. In a similar way, common visual representations of RPG games are used, such as health bars, characters' dialog windows, and sprites for the same. The visual feedback of the player's interaction with the elements on the screen is based on the changes of hues and approaches.

Moreover, an interesting context has been created to motivate playing the game. *Classutopia* is a technological utopia wherein human beings have reached the pinnacle of civilization thanks to the help of technology. Amidst all significant technological advances achieved, five are considered the most important, called *The Pillars*, in which all this social prosperity is underpinned: a satellite plant of solar energy, a synthesized food system, an organ cultivation center, an anti-pollution multi-industrial system, and a robots factory for unwanted tasks.

All the basic needs of humanity (health, food, and energy) have been met. The villain in this story is the last descendant of a caste of sorcerers, who is bothered by the fact that humanity has reached fullness thanks to technology and not magic. Therefore, he has initiated an attempt to sabotage *The Pillars*, making use of his magic tricks to modify this systems software by altering the class models to overthrow the utopia. However, a maintenance robot has become aware of these failures; the reason why it pursues an original adventure to repair the damages and ruin the sorcerer's plans. This context works as a guideline for the creation of the diagrams arranged for the missions since each mission represents the robot's attempt to save one of *The Pillars*. A diagram is modeled to represent the satellite plant of solar energy, synthesized food system, and organ cultivation center; these diagrams are presented in the missions 1, 2 and 3, respectively.

Classutopia is available for download at https://www.dropbox.com/s/sfcjjz0jlp6a911/Classutopia.apk?dl=0

B. Mechanics

The game is based on three types of mechanics: modeling the character, correcting defective diagrams, and understanding diagrams.

1) Character Modeling

The main character is a robot with three basic attributes: attack, defense, and speed. Attack relates to the damage the protagonist can cause to the enemy, defense is the resistance posed to the damage received from the enemy, and speed is the management of the limited time to solve the problems. These attributes can be modified and enhanced through modeling of class diagrams, which describe the features and components of the robot. Initially, the robot needs to be modeled correctly, and subsequently, as the game progresses, new improvements can be unlocked, which should be properly included in the class model of the robot so that the changes take effect. These improvements, in addition to modifying the character's attributes, also unlock some special powers that can be used in the clashes between the protagonist robot and its enemy (the magician).

2) Correction of Defective Diagrams

During the game, the hero battles against the enemy; the disputes are based on the challenge of correcting defective class diagrams. In these challenges, given a defective class diagram, the player can tap the screen of the tablet to highlight the various elements that make up the class diagram, which have been considered to be defective. Therefore, *Classutopia* offers options to correct the possible defects. Once the correction has been made, feedback on whether the player has made a mistake or done it correctly is not instantaneous but the player must "Attack" to be able to assess the correction. This has been designed in this manner to give an option to the player to evaluate one or more modifications made to the diagram by pressing a single button, in which mistakes result in harming the protagonist (robot) and the successful corrections affect the enemy (magician). This is graphically presented on the characters' health bars that indicate the status for each character. In addition, *Classutopia* delivers feedback in the class diagram; it indicates in green the modification that was applied correctly and in red where it has been incorrectly modified. It also includes a button to undo the recent applied modifications.

3) Understanding Diagrams

During the battles, the player can activate special powers in the robot that allow him to perform various effects such as enabling damage modifiers to apply more force on the attacks, reducing corrections in the diagram to finish it on time, or healing the protagonist. These modifiers have limited use and are activated by means of a challenge in understanding class diagrams. By pressing the button which activates a certain power, the player must answer a question with respect to the concepts that can be used in a class diagram. This question displays a small diagram or some conceptual construct and offers between two and four possible answers. If the player answers correctly, the power is effective; if answered incorrectly, the effect will not be applied. The powers are arranged in the battle according to the previously modeled characteristics for the robot.

C. Dynamics

The mechanics previously described have been presented to the player in a smooth, consistent, and understandable way during the execution of the game. To obtain the expected gameplay, a series of screens, menus, and systems are implemented, which are explained below.

1) Control Systems

Like most mobile applications available today, the game is controlled via touch-screen by simply tapping on buttons and menu items displayed on the screen. In some game mechanics, multi-touch gestures are used to zoom in and out and scroll through the proposed class diagrams.

2) Start screen and save game handling

When the player runs the application, a screen is shown with the name of the game, a representative image, and a Start button to initiate the game. It is a simple landing screen like that of most mobile games. Once the player has clicked on the Start button, the button disappears presenting to the player two new options within the same screen (see Figure 1): one to start a new game and the other to load a previous game. The game uses a system of unitary games, i.e., you can only save one game at a time, therefore, creating a new game automatically deletes (if any) the previous game. For this reason, as a precaution, a warning window to confirm deletion is displayed to avoid accidental loss of the game. The games are saved automatically, indicating this with a symbol at the end of each challenge. If the player presses the button to start a new game, the stored data are reestablished and the game moves on to the main menu. If the player presses the button Load Previous Game, it goes directly to the main menu preserving the conditions recorded during the last auto-save.

Fig. 1. Game selection. Upper text: Classutopia. Lower left text: START NEW GAME. Lower right text: LOAD PREVIOUS GAME

3) Main menu

The main menu displays three buttons (see Figure 2): one to access the construction challenge of modeling the main character, one to access the missions' menu, and one to return to home screen. In addition, it displays a summary table with the current attributes of the character robot. The first time the player accesses this screen (when he/she starts a new game), the missions button is disabled in order to force the player to model the character on the construction screen before accessing the missions' menu.

Fig. 2. Main menu. Upper right text: CURRENT ATTRIBUTES: ATTACK 1, DEFENSE 1, SPEED 1, HEALTH 100. Left text: BUILD UP, MISSIONS.

4) Build and upgrades

On the construction screen (see Figure 3), the mechanical modeling of the character is presented, as has been previously described. Initially, the player must connect the classes that represent the parts of the robot correctly, for example, sensors, extremities, core, and weapons. The player can see, as he/she models the robot appropriately, how its attributes (attack, defense, speed, and unique skills) modify according to the characteristics of the diagram. Defects in modeling during the

construction of the diagram result in the non-modification of the attributes, i.e., the classes do not apply their effects and the features of the robot are not improved. There is a button to save the changes made, continuing to the main menu. Within this screen, the list of improvements for the robot is presented, those that get unlocked, as a reward, each time a mission is completed. These represent the improved components of the robot or new components that, by being adequately included in the model, increase the attributes and activate new powers.

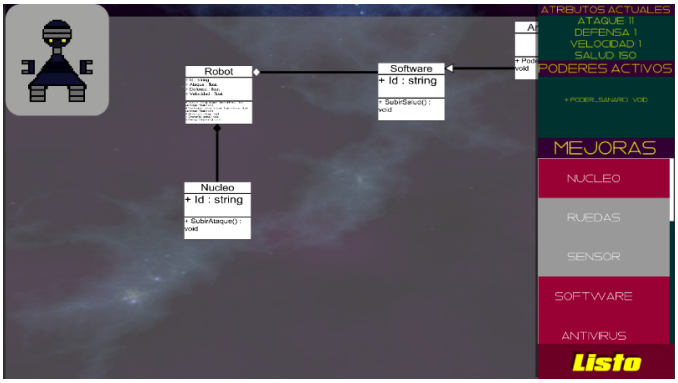

Fig. 3. Building screen.

5) Missions

After modeling the robot for the first time, in the main menu, the button to access the missions' menu is enabled. The missions correspond to each of the clashes the robot must complete by defeating the magician (see Figure 4). These tasks are presented as a list and are sequential, i.e., the second mission is enabled when the previous mission is completed, and the third is activated upon completion of the second mission. Each mission is shown with a name and its level of difficulty. The difficulty of the missions is incremental. In the first instance, the game has three missions: the basic level, mid-level, and advanced level. The missions can be repeated for practicing or unlocking new rewards (those corresponding to the accomplished level). To start a mission, the player must tap on it, by doing so it gives way to the battle.

Fig. 4. Missions menu.

6) Battle

The battle screen shows the class diagram to be corrected (as explained in Mechanics Section), which can be navigated using tactile gestures (see Figure 5). The avatars that represent the parts in conflict are shown at the top of the screen with their respective health bars and with the remaining time for the mission. The confrontation ends when the time runs out, when one of the parties loses all their health or when all defects are corrected. If the player (robot) wins, the level is completed, regards are awarded, progress of the game is saved, and the missions menu appears. If the player (robot) loses, the missions menu appears without rewards but with the option to reattempt to complete the mission.

Fig. 5. Battle screen.

Each level focuses on the correction of different groups of defects in class diagrams. The basic level focuses on learning the essential elements within a class (names and attributes). The medium level focuses on learning the correct specification of services in a class. The advanced level focuses on learning the associations and cardinalities. Table 1 shows the injected defects in each of the levels.

TABLE I. DEFECTS INJECTED IN CLASSUTOPIA LEVELS

Level	Defects
Basic	• Classes without name • Classes without id • Classes with repeated attributes • Classes with attributes without data type
Medium	• Classes without creation service • Classes with service without return • Classes without visualization of attributes
Advanced	• Associations of wrong type • Association without cardinality • Minimum cardinality of 1 at both ends • Descending cardinality at one end of the association

The more advanced levels include the types of defects of the levels of lesser complexity. These defects are applied randomly at the beginning of a confrontation on a diagram without defects, i.e., the game can generate any of these defects in any of the elements of the class diagram presented in a battle and generate another type of defect, completely different, in the same elements when repeating the mission.

7) Special Powers

On the battle screen, there are buttons to activate the special skills obtained in the construction of the robot (see Figure 6). By tapping on a skill, the understanding problem is presented, which must be answered correctly for its activation. While this occurs, time is not paused. Once the problem is addressed, the

battle continues and if answered correctly, the effects of the skill are applied; otherwise, the effects do not apply. Whether it is answered correctly or incorrectly, the selected skill is disabled for the rest of the combat. The types of effects that the skills have are as follows: Multipliers of special powers, Health recovery of the protagonist, Mission's time manipulation or Reduction of remaining defects.

Fig. 6. Special powers panel.

IV. Evaluation of Classutopia

An exploratory empirical evaluation of *Classutopia* has been conducted with the aim of assessing the understanding of the game and the perceived utility to support the conceptual modeling learning in the Software Engineering course. For the design of this activity, the guidelines previously defined in [16] were used. The following research question was defined:

RQ1: Do the students find the use of Classutopia for learning the conceptual design of class diagrams beneficial?

The context of the experiment is included in the Software Engineering course, which is a fourth-year course of the Engineer degree in Computer Science and Telecommunications at the Diego Portales University (Chile). The subjects are students of this course that have already attended classes on conceptual modeling. These classes have been conducted using traditional techniques of teaching/learning such as PowerPoint slides and inspection exercises in diagrams printed on a sheet.

In the experiment, the subjects receive a small description of what is *Classutopia* and then they must play. They begin building the robot and go through the missions. Finally, the subjects must respond to a questionnaire in order to obtain the perceptions of the usefulness of the game as a new technique of teaching/learning of conceptual modeling in Software Engineering. The sentences of the questionnaire are presented at Table 2, and subjects must answer using a Likert scale, where 1 is totally disagree to 5 totally agree.

TABLE II. Questionnaire to Assess Classutopia

	Sentence
1.	Classutopia facilitates the understanding of a class diagram.
2.	Classutopia helps determine how to correct a class diagram.
3.	Classutopia helps to understand the concepts of the class diagram.
4.	I find Classutopia easy to understand.
5.	Classutopia allows you to learn about the design of a class diagram in an easier way than by using text.
6.	In general, I found Classutopia to be useful.

A. Operation of the empirical evaluation

The research was conducted in first semester of 2017 with students who have completed the course of Software Engineering in the second half of 2016. Thirteen students participated voluntarily. Participation in the experiment was not related to the score of the students but the students were urged to make their best effort. The start and end of each student for each mission and the times they had to conduct a mission to win it were registered. It is important to mention that each student should complete mission 1 (M1) to unlock mission 2 (M2) and further complete M2 to unlock mission 3 (M3).

The students had 30 minutes to conduct the activity. However, most subjects wanted to continue playing to defeat the magician. The students realized that each time they earned a mission, they unlocked new features that could improve the robot and they were going to build the robot to improve their attributes. In addition, the subjects used the easier mission to unlock the features thanks to they win many times to build the robot to defeat the most powerful wizard in the last mission.

B. Results

Results are shown in Table 3. It can be observed that 7 subjects were able to correctly build the robot and detect all defects injected in the diagrams.

TABLE III. Results

Subject	Time M1	Num. of M1	Time M2	Num. of M2	Time M3	Num. of M3	Win
S1	5	2	13	3	14	7	yes
S2	2	1	25	8	-	-	no
S3	3	1	27	6	22	9	yes
S4	4	3	7	2	15	5	yes
S5	5	3	22	5	-	-	no
S6	1	1	12	5	7	3	no
S7	3	2	10	4	6	3	yes
S8	11	11	7	2	46	16	no
S9	11	5	37	12	20	8	no
S10	2	1	1	1	6	3	yes
S11	4	2	28	7	21	6	no
S12	4	2	17	5	22	8	yes
S13	3	2	8	2	30	10	yes

Of these 7, 3 subjects exceeded the game in 30 minutes but 4 subjects were engaged in the game to overcome it. Four subjects correctly build the robot but did not exceed M3 that had defects in the associations of the classes. Two subjects did not exceed M2 that had defects in the services of the classes and did not continue to play after 30 minutes. It is important to note that the four subjects who did not win had to stop playing the game although they were eager to continue playing.

The results indicate that all students (13 subjects) could learn how to build the robot and to detect defects in the class attributes (basic level). 11 of them were able to learn how to build the robot and to detect defects at basic and medium levels. From these 11 students, 7 students could learn how to build the robot and to detect defects in the class diagrams at basic, medium, and advanced levels. This allows to observe whether it is feasible to use a serious game to learn about the conceptual modeling software, the game improves the

students' understanding of class diagram, and encourages us to improve the testing to evaluate the effectiveness in learning.

Regarding the survey, Questions 1, 2, and 3 are related to the perception of the subject matter of learning using the serious game. The results (see Figure 7) indicate that the students perceive that Classutopia helps in the comprehension of the class diagram, in the understanding of how to construct a class diagram, and that Classutopia helps to understand the concepts used in the class diagram (the majority of the subjects agree or totally agree with the statements of Q1, Q2, and Q3). Regarding the ease of use of the game (Q4), there are two subjects that did not find it easy to understand how *Classutopia* works. However, more than 50% of the students agree or totally agree that it is easy to understand how the game works. Regarding the utility of the game, the majority of students agree that *Classutopia* allows to learn more easy than using texts (Q5), and they are totally agree that in general found *Classutopia* useful (Q6). The results of this survey allows to answer the research question since the students find it beneficial to use a serious game for learning conceptual modeling, in particular, for learning class diagrams.

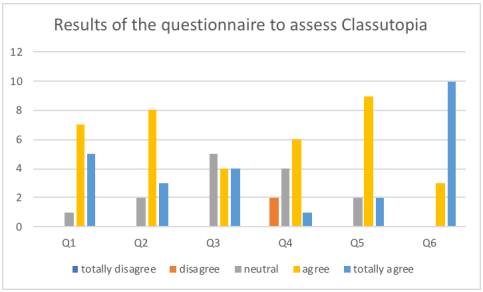

Fig. 7. Results of the questionnaire.

V. CONCLUSIONS

Novel teaching techniques in computer courses are required, particularly for the advanced levels of Software Engineering, because in many cases, the emphasis of research has been placed on the most basic courses such as programming. The design, implementation, and use of a serious game to learn conceptual modeling in software engineering courses are presented in this study. From this work, it can be concluded that it is feasible to implement new teaching techniques with gamification and serious games for Software Engineering courses.

In the development of computer projects, modeling a correct class diagram is of vital importance for the reduction of problems and delays in deploying a software. In this work, we provide a solution for the problems caused by the misunderstanding of class diagrams and their semantic components. Thus, *Classutopia* has been presented, which is a serious game that supports the conceptual modeling learning, and it can be a good solution to correct the most common problems that students have to familiarize themselves with class diagrams, offering them a space free of pressure and consequences to learn and understand how to model and correct class diagrams.

In addition, there has been an exploratory empirical evaluation to verify the perception of students with respect of this new method of teaching/learning in Software Engineering. Results indicate that *Classutopia* provides benefits for learning conceptual modeling design. We are aware that with a limited number of subjects, it is not possible to apply statistical techniques to increase confidence in the results. One of the limitations of the empirical assessment performed is the lack of evidence of the effectiveness of *Classutopia* in the learning process. Therefore, further work is referred to conduct experiments to evaluate the effectiveness of *Classutopia*. Finally, there are plans to improve the Classutopia graphics aspects that will help to enhance the gaming experience.

ACKNOWLEDGMENT

This work was funded by CONICYT project ENSE REDI170020, 2017-2019.

REFERENCES

[1] OMG, Unified Modeling Language (UML) 2.4.1 Superstructure Specification, 2011.

[2] D. Giordano, F. Maiorana, Object Oriented Design through game development in XNA, 3rd Interdisciplinary Engineering Design Education Conference (IEDEC), pp. 51-55, 2013.

[3] J. Cabot, Common UML errors (I): Infinite recursive associations. http://modeling-languages.com/common-uml-errors-i-infinite-recursive-associations/ (last access 14.03.18)

[4] Guidelines for UML Diagram Development. http://eng.umd.edu/~austin/nsf-crcd/uml-guidelines.html(last access 14.03.18)

[5] B. Marín, G. Giachetti, O. Pastor, A. Abran, Identificación de Defectos en Modelos Conceptuales utilizados en Entornos MDA, XII Ibero-American Conference on Software Engineering (CIbSE'2009), pp. 109-114, 2009.

[6] C. Abt, Serious Games, University Press of America, 1987.

[7] M. Zyda, From visual simulation to virtual reality to games, IEEE Computer, vol. 38, issue 9, pp. 25–32, 2005.

[8] T. M. Connolly, E. A. Boyle, E. MacArthur, T. Hainey, J. M. Boyle, A systematic literature review of empirical evidence on computer games and serious games, Comput. Edu. Vol. 59, issue 2, pp. 661–686, 2012.

[9] A. Calderón, M. Ruiz, A systematic literature review on serious games evaluation: An application to software project management, Comput. Edu. vol 87, pp. 396–422, 2015.

[10] J. Vargas-Enríquez, L. García-Mundo, M. Genero, M. Piattini, Análisis de uso de la gamificación en la enseñanza de la informática, XXI Jornadas de la Enseñanza Univ. de la Informática (JENUI 2015), pp. 105-112, 2015.

[11] J. Zhang, E. Caldwell, E. Smith, Learning the concept of Java inheritance in a game, 18th International Conference on Computer Games (CGAMES), pp. 212-216, 2013.

[12] S. Sharma, J. Stigall, S. Rajeev, Game-theme based instructional module for teaching object oriented programming, International Conference on Computational Science and Comp. Intelligence (CSCI), pp. 252-257, 2015.

[13] J. Livovský, J. Porubän, Learning object-oriented paradigm by playing computer games: concepts first approach, Central European Journal of Computer Science, vol. 4, issue 3, pp. 171–182, 2014.

[14] Z. Li, L. O'Brien, S. Flint, Object-oriented Sokoban solver: A serious game project for OOAD and AI education, IEEE Frontiers in Education Conference (FIE) 2014.

[15] R. Hunicke, M. LeBlanc, R. Zubek, MDA: A formal approach to game design and game research, AAAI Workshop on Challenges in Game AI, 2004.

[16] C. Wohlin, P. Runeson, M. Host, M. Ohlsson, B. Regnell, Experimentation in Software Engineering, Springer, 2012.