ISSN 2007-9737

SimulES-W: A Collaborative Game to Improve Software Engineering Teaching

Elizabeth Suescún Monsalve[1,4], Mauricio Toro[1], Raúl Mazo[1,2], David Velasquez[1], Paola Vallejo[1], Juan F. Cardona[1], Rafael Rincón[1], Vera Maria Werneck[3], Julio Cesar Sampaio do Prado Leite[4]

[1] Universidad Eafit, Grupo de Investigación Desarrollo e Innovación en Tecnologías de la Información y las Comunicaciones (GIDITIC), Medellín, Colombia

[2] Université Panthéon Sorbonne, Centre de Recherche en Informatique (CRI), Paris, France

[3] Universidade do Estado do Rio de Janeiro, Departamento de Informática e Ciência da Computação, Brazil

[4] Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Informática, Rio de Janeiro, Brazil

{esuescu1, mtorobe, rimazop, rrincon, dvelas25, pvallej3, fcardona}@eafit.edu.co, vera@ime.uerj.br, julio@inf.puc-rio.br

Abstract. There is empirical evidence concerning the effectiveness and benefits of game-based learning (GBL). Our mainly interest is to present a tool that can be used to complement teaching software engineering in a motivating and didactic way. This paper studies the use of a GBL tool called SimulES-W (Simulation in Software Engineering), to teach Software Engineering in an undergraduate engineering program. SimulES-W has three characteristics: it is based on real software cases, it can be customized during the learning process, and it is a collaborative game. These characteristics are important because they help us understand and propose a new learning scenario, and to research with this the learning processes in their environments According to it, the first characteristic of SimulES-W makes it a motivating and engaging game, which brings up cases, which usually are only present in real software projects. Thanks to the second characteristic, the educators can use SimuelES-W to customize the education material, and tune the game for specific software engineering courses. The third characteristic is related to the proposed game as activity that involves group discussions and decision-making. This paper presents SimulES-W a digital version of SimulES and reports the results of an evaluation from a pedagogical perspective, where game adequacy for teaching a subject and positive potential impact in student's academic performance are investigated.

Keywords. Software engineering, game-based learning, games, pedagogy of software engineering.

1 Introduction

Computer games and simulations have been applied to explore reality in many educational areas. Conolly et al. [1] produced a systematic review of computer and serious games showing that games can be considered for motivation and effective impact on the learning environment. GBL literature reports that games are effective for learning with entertainment and to simulate situations that occur in real software projects. Games simulating software engineering projects allow students to take the role of a project manager and deal with software engineering management

issues, which are difficult to simulate during traditional lectures.

In this paper, we present a Software Engineering (SE), collaborative board game, called SimulES-W, used to teach concepts enacting a software engineering process. In SimulES-W players perform different roles, such as software engineer, technical coordinator, project manager and quality controller. Players deal with budget, software engineers hiring and firing, and construction of different software artifacts. The game simulates a development process where players must deal with:

i. The complexity and size of a software product,
ii. The concept of product quality based on verification through inspection,
iii. The risk of having poor quality products,
iv. Budgeting,
v. Admission and dismissal of software engineers,
vi. Human resources as a matter of cost, productivity and maturity, and
vii. Construction of several different artifacts required for project completion.

In addition to different roles, SimulES-W allows players to have a strategy-oriented game, where each player could (i) pose problems to other players and (ii) use concepts to improve their ability to solve these problems. In SimulES-W, both a competitive process and a collaboration process are used. The competitive process, implemented as a special round, is enacted as each player chooses which other player will receive a "problem card".

The collaborative process occurs since players who have a "concept card" may use it as a counter measure for the "problem card", but only if the other players agree with the argumentation of why that card could "solve" the problem given by the opponent. This discussion, mediated or not by an instructor, is a way of discussing concepts, problems and possible resolution schemes (of the corresponding cards), which becomes an opportunity to learn the fundamentals of SE.

This paper is organized in six sections. Section 2 introduces SE education and reviews some games and proposals that are used to teach this discipline. Section 3 gives an overview of how the previous SimulES versions were improved, the SimulES-W development process and it also explains the game components, rules and its dynamic play process. Section 4 describes how SimulES-W can be used to teach concepts of software engineering. Section 5 reports on an experiment of using SimulES-W in the classroom evaluated by different criteria, including exam impact. Section 6 presents our conclusions.

2 Teaching Software Engineering

Software engineering is an area of computer science that offers methods, techniques and tools for building software with quality. Teaching SE just based on traditional lectures and providing emphasis on theoretical aspects fails to pass to students the major challenges/problems a software engineer faces in real projects. Therefore, students need to face practical aspects and project decisions, which are difficult to identify in a theoretical way. According to Boehm [2] game-oriented SE education is forecasted as one of the future trends of the SE field.

2.1 Related Work

In the past, it was already clear the need to simulate the processes of SE, as it is presented by Lin et al. [3]. Their work examines trade-offs of cost, schedule, and functionality, and to test the implications of different managerial policies on a project's outcome and enable software managers to gain a better understanding of the dynamics of software project development and provides a learning environment through simulation where the implications of different policies on a project can be studied, and insight can be gained into the causes of project dynamics.

In this same direction, Kellner et al. [4] focused their work on simulation; it offers a set of simulations that can be selected and serve as guides to practical application, so it uses a set of working hypotheses through models. There are also works that show how to simulate processes in requirements engineering [5, 6]. They show, as the major goals behind the modeling and simulation effort are to assess the issues associated with the social and behavioral aspects of the EasyWinWin process, and to explore how these issues affect the overall outcome of the process. Pfahl et al. [7]

present concepts of a computer-based training (CBT) and argue that university education needs to provide to their computer science and SE students not only technology-related skills but also a basic understanding of typical phenomena occurring in industrial (and academic), software projects. Thus, in a recent study, Chen and Chong [8] show that some aspects in SE education can be taught through simulations by offering comprehensive training that explores subjects such as collaborative software development, team projects and the social aspects of software development. It allows students to experiment real problems rather than mere academic exercises.

Wangenheim et al. [9] identified the potential to teach SCRUM in concrete situations through an educational game and complement the learning process through theoretical lectures. According to authors, SCRUMIA was created based on experience in class and using instructional design. This game has been applied several times in two undergraduate project management courses. They also evaluated the competency, understanding and motivation. SCRUMIA explores a specific situation through a case study.

Other authors such as Qin et al. [10], Barros and Araújo [11] and Alvarez et al. [12] Carried out some experiments in which they taught SE with games in small software projects. Bollin et al. [13] highlight the importance of using real (usually large) projects to teach SE in addition to soft skills such as economic planning responsibility, ethic and project evaluation. These issues should be better taught by means of simulations or games, which caught partial situations of real projects.

For example, the AMEISE system [13] allows students to exercise their software project management skills and reflect about their decisions. These decisions are based on incomplete and uncertain information and feedback, which are provided by the system. Certainly, GBL could be used efficiently to supply and simulate real processes.

In the same way, GBL (of board, cards and computer), simulators and others are being explored to support teaching in different areas of knowledge [14, 15, 16].

These approaches mainly support the idea that GBL should be combined with aspects such as motivating the students, being enjoyable and providing software, as well as providing an environment where students do not feel like they are learning [14]. GBL also is an important didactic proposal to explore collaboration. For instance, Alvarez et al. [12] show the benefits of including GBL tools in the classroom to foster collaborative learning and active student participation. The authors believe that games can be used as a tool supporting teaching methods that are effective towards the educational objectives.

Ebner and Holzingerb [14] suggest that there is evidence, which shows the learning result of using games is at least equivalent to the results from learning using the traditional method. For that reason, it is important to consider games as a modern and useful method for learning as they do not disturb or not offer disadvantage for the learners. Quite the contrary, the study carried out points out that GBL is one of the most preferred modes of learning from the point of view of students.

There are others approaches to teach SE by means of the implementation of computer game-development course. The students perform different roles in different areas of SE, the aim is to improve abilities, skills and prepare software engineers for industry demands in an interesting way as they use game-inspired exercises [17]. In addition, Qin and Mooney [10] proposed to use game-oriented projects as a promising choice to make the learning environment as close to the real-world software development environment as possible.

Claypool and Claypool [18] suggested building today's entertainment applications, which use computer games, coupled with the software engineering discipline, arguing that it presents an opportunity to use computer games as a means to better train software engineers. Project-based modules can be used to illustrate all aspects of the software process, tapping into a broad range of SE disciplines as is required to build current applications while enticing students to grasp and apply software engineering to such disciplines by using games as a powerful motivator.

There are quite a lot of papers providing evidence that games and SE have attracted attention of software engineer's educators [19, 20, 21]. Hainey et al. [22] Describe problems that are associated with traditional approaches to teaching

requirements elicitation and analysis, with the authors suggesting that this is very difficult for a traditional course teaches students some of the skills that are required of professional software developers, and that traditional lectures do not adequately tackle the software development process.

Nevertheless, there are advantages and disadvantages of GBL as compared to traditional teaching as listed by Hainey et al. [22]. For instance: lectures have as advantages control of the learning experience and provide models of how to address problems; as disadvantages there are no mechanisms to ensure that students are intellectually engaged, it is possible to lost the attention after a short time, and it is assumed that students are at the same level of understanding even if they are not engaged.

Just like GBL has advantages providing a safe environment to increase practice experience, GBL helps the students can naturally transfer what they have experienced in class to real life because they receive immediate feedback. Nevertheless, there are disadvantages: it has to be well planned, monitored and in some cases, it could put pressure on learners and could result in embarrassment. On the other hand, the authors also contribute to the empirical evidence in the context of teaching requirements elicitation and analysis, as they report on a five evaluation experiments comparing a GBL approach to a role-playing and paper-based approach. This evidence is collected thought of evaluations took place in Higher Education (HE) and Further Education (FE). Moreover, the results showed a significant increase in knowledge at both FE and HE level after the experiments had been applied. They also believe that the GBL approach to teach requirements elicitation and analysis may be more suitable to HE level than FE level. Finally, they suggest that initial knowledge should be taken into account when considering a GBL approach for different educational levels.

Paraskeva et al. [23] describe online computer games and its teaching potential, highlighting some studies that have been focused on classifying types of teaching, which are supported by games. The authors also show the potential to draw together players from different contexts to communicate and collaborate. More than that, they suggest that with GBL it is possible to address ways that the student does strategies, hypothesis testing, or problem solving, preferably with higher order thinking rather than rote memorization or simple comprehension.

Paraskeva et al.'s work shows that the successful games have to be related to characteristics like rules, goals and objectives, outcomes and feedback, conflict (and/or competition, challenge, opposition), interaction, and representation of story. Finally, this work also examines *the educational value of games doing*. Along of a literature review this *value* is described, showing the games' potential for player engagement, justifying with research reports on factors such as game use (frequency of game use, gender differences, identification with the characters, game preferences), and some psychosocial factors that may influence learning (academic performance, self-esteem, computer self-efficacy).

Boyle et al. [24] and Connolly et al. [1] present two systematic review of empirical research. They examined different aspects related to engaging and enjoyable activities in games. On the one hand, Boyle et al. focused in showing aspects of experiences with entertainment games like motivation for playing games; game usage and time spent playing games and the impact of playing on life satisfaction. Boyle et al. show that understanding game usability has had priority over understanding game enjoyment, cognitive and emotional involvement. The authors describe that subject, as real world dissociation, challenge and control are aspects little explored.

On the other hand, Connolly et al. [1] focused on computer games and serious games showing empirical evidence about the positive impact of learning and how they are still growing, as noted earlier. They also quote modern theories that suggest that learning is most effective when is active, experiential, situated, problem-based, and provide immediate feedback. Connolly et al. [1] show in their related work how games are integrated into the learning experience indicating aspects as feedback, strategies for varying difficult level, and availability of support memory that are keys to the success of the games-based approach. For that reason, the authors suggest that to encourage the use of games in learning it is important to develop a better understanding of the

Table 1. Summary of Software Engineering game features

Game name	Game goal	Player goal	Game modeling
1. Problems and Programmers (PnP)	Teach software engineering.	Simulate the process of development in waterfall.	Does not have.
2. SESAM	Teach project management.	Create a model of software development process and run it using a simulation system.	Documentation related to specification, architecture, design, definition, and implementation of work environment.
3. SimVBSE	Teach software Engineering value.	Identify the stakeholders in the system with what they perceive as critical success factors and values, all of that in a simulated setting.	Development based on prototypes.
4. SimSE	Teach software engineering process.	Complete a software engineering project.	Modeling as it has different version of the game.
5. Planager	Teach some project management concepts.	Simulate some of the processes used in project management, mainly planning processes.	With object-oriented UML.
6. Scrumming	Teach through simulation agile project management practices, mainly SCRUM.	Make a simulation assuming the role of SCRUM Master.	With object-oriented UML.
7. X-MED v1.0	Teach software metrics.	Simulate a measurement program aimed at project management. All of that aligned to maturity level 2 of the CMMI-DEV.	Based on its architecture, ie. it was developed from a layered architecture.
8. SCRUMIA	An educational game for teaching SCRUM in computing courses.	Strengthen the understanding of SCRUM concepts and to exercise the application of the SCRUM process.	The game has been systematically developed following the Instructional Systems Design Model.

tasks, activities, skills and operations and, to be fair, examine how these might match desired learning outcomes.

Although authors like Caulfield et al. [25] expressed that more evidence and research is required to prove the efficacy of games in teaching SE, we believe that teachers should consider using games as part of their courses because they could become useful and interesting add-ons.

2.2 Using Game Tools in SE Education

Given this context of GBL, we will focus on SE education on a game named SimulES and its evolution with different versions until it becomes a digital version named SimulES-W.

We agree with Liu et al. [26] that if students learn computational problem solving with games they will be more likely to perceive a flow learning experience than in traditional lectures. They also confirm that there is a close association between the students' learning experience and their problem solving strategies.

Our study with SimulES-W, analyzed feedback, examined the differences between the game and traditional lecture and problem solving behaviors. We found that these behaviors are strongly related to: a) understanding the problem, b) devising a

Table 2. SimulES to SimulES-W evolution

Version	**1 (Board game)**
Characteristics	- Evolution from PnP. - It integrates evolution concepts in the game. - It incorporates the first board game. - It uses a flexible development process.
Experience of Use	It was used in PUC-Rio when it was built. Until now, it is used in UFMG. Both to teach Software Engineering.
Version	**2 (Board game)**
Characteristics	- It includes a modeling based on scenarios. - It incorporates the individual board. - Proportion of defects in white and gray cards was balanced. As well as *time points.*
Experience of Use	Interactions with undergraduate and graduate students in PUC-Rio.
Version	**3 (Board game)**
Characteristics	- It includes an intentional modeling based on i* framework.
Experience of Use	Interactions with graduate students in PUC-Rio and Universidade do Estado do Rio de Janeiro (UERJ).
Version	**4 (Web game)**
Characteristics	- Digital version named SimulES-W. - It includes an intentional modeling based on i* framework. - Proportion of defects in white and gray cards was balanced. As well as *time points.* - It is customizable according to issue that will be taught. - Online references can be consulted and discussed.
Experience of Use	Interactions with undergraduate and graduate students in PUC-Rio and UERJ.

solution plan and c) testing the plan as was mentioned in Liu et al. [26]

Fully understanding the game is important to meet learning aims because perception of the game gives players “the true game” [27]. In other words, when a player has a much better understanding of the game than the other player, the first can enjoy more with less effort and consequently achieve aims in an appropriate way.

The results presented in Hanaki et al. [27] show that players might feel good without fully understanding highly complex strategic environments. The authors thought their results suggested that players might have a good understanding about the dynamic of the game, but it is necessary to maintain a level of mystery enveloping the game strategies as to live space for discovery. In the same way, it is necessary that

players have a reasonable knowledge about their playoffs and opponents' playoffs.

In the case of SimulES-W, payoffs can be higher if the game is in a more advanced stage when players have more artifacts, such that players can improve their game and damage opponents' games, despite the SimulES-W randomness provided by throwing dice. It is interesting to notice how the players enjoy and feel satisfaction with the game's payoff and how they use them.

There are simulators to teach software engineering such as SESAM [28], SimVBSE [20], SIMSE [21], and others presented in Table 1.

SESAM [28] works as a simulation system that is able to execute models. It is focused on teaching software management. The basic idea of the game is to create a software process model with particular data, which is then simulated by the system. Quantitative data is generated based on the user selections for the specific project. As a result of the simulation, it is possible to analyze the process and the user choices.

SimVBSE [20] is a game for students, which help them to better understand value-based software engineering. It departs with what the stakeholders have defined as critical factor of its success and the preference values, this last one is a concept predefined in the game and that allows to be analyzed in detail. The users (students) should identify what stakeholders regard as critical success factor (choose this concept) and determine a strategy to balance the critical factors with the other preference values.

SIMSE [21] is an interactive educational software that is used with a single player and simulates a software engineering process. Therefore, it guides students through the different software processes, in which students have to deal with budget, project time and other difficulties that arise when the simulation is running.

At the same time, students must make decisions that could affect positively or negatively the project. The idea in this game is to finish the project within the stipulated time and budget. These games are similar to SimulES because they were evolutions of the game "Problems and Programmers", also known as PnP [29].

Approaching these games from the point of view of how they were specified or modeled, we observed the following. (a) SESAM game does not have any pre-defined method related to documentation on specification, design and architecture. (b) SimVBSE was built by prototyping method based on the different evolutions of itself.

And (c) SIMSE has been modeled for each one of its versions, i.e. each version of the game has its own specification, modeling and architecture with an individual approach to each. PnP was the source of inspiration for the different versions of SimulES up to SimulES-W.

3 SimulES-W

SimulES-W incorporates the concepts of SimulES, therefore, it contains the concepts of the PnP game [29]. The difference between SimulES and the PnP game is that the first one does not impose an order to build software artifacts or a particular software development process. A brief compilation of the game's history, SimulES was conceived after an analysis of advantages and disadvantages of PnP, as detailed by Figueiredo et al. [30].

This analysis led to the creation of the first version of SimulES. This version included important differences between both games, for example: SimulES integrated the concept of evolution by incorporating a new board in the game, as well as improved and reformulated the cards.

But, the most representative improvement has been that it allowed the players to use a flexible development process. This means, when the players (students), develop a software product, they can choose a developing strategy according to their convenience and ability.

This resembles the reality of real projects whereas each project may decide which process to enforce or may decide not to enforce a specific software process like PNP using phases to develop a software product, which is still taught in some software engineering courses. Figueiredo et al. [30] also provided some challenges for future research, which included automation and more usage of the game as to gain more insight of its potentialities as well as its drawbacks.

The elements used to improve and evolve SimulES are the results, feedback and observations of the users [31, 32, 33]. Table 2

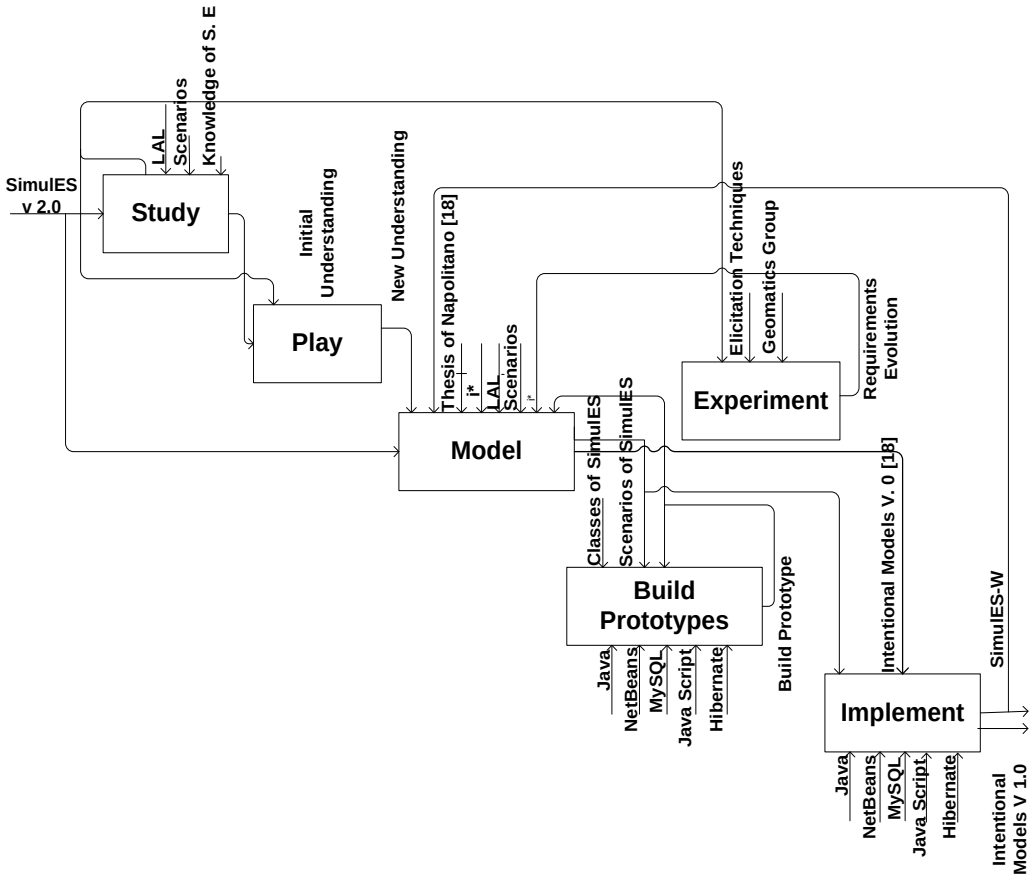


Fig. 1. SADT diagram – The process to build SimulES-W

summarizes the SimulES versions characteristics and experience of use.

Version 2.0 of SimulES, described in [34], included different improvements. One of the improvements was the usage of a scenario-based specification [35] to describe the game behavior and its rules this as part of the game analysis and design artifacts. Another improvement was the redesigned of the main board as to better guide the players, this element allow player organize public information related to the game. This version also reduced the proportion of defects (bugs), and changed the value of time points on the white and gray artifact cards as to create a better balance among card types.

This version was used in the works of Serrano et al. [34] and Napolitano [32], which profited from the experience with the game as it was played by different students and researchers.

Napolitano [32] used the i* [36] language to model SimulES. This work was the seed for SimulES version 3.0.

As depicted in Table 2, we presented the evolution of SimulES to SimulES-W [37]. This last is a digital version of the game created on top of a refinement of the goal model presented by Napolitano [32].

The design of SimulES-W was a combination of several elements. Firstly, we studied other online games. After, Monsalve played SimulES to better understand it, and we continue observing how players interacted with the board game. This experience was reported in [38] and [39].

Monsalve [38] presents the results of an experience in which an interdisciplinary group plays with SimulES.

Using observation techniques and questionnaires, information was collected. This information was important for providing insights and ideas, which helped tuning the SimulES-W design process. An example is the case of initial training before the game, also the contents in the cards should been revised because some were difficult to understand by the average students and some rules of the game needed to be sharpened.

The experience was also important to confirm the acceptance of SimulES by students. They reported that SimulES was a positive, motivating and entertaining experience.

Monsalve et al. [37] provide extended observations on the use of SimulES-W as a learning tool for SE students, mainly because it is fun and motivates a healthy competition. The work also reports on new feedbacks that led to the improvement of SimulES-W. Next section stresses these new features of SimulES-W; some control actions are now delegated to SimulES-W, freeing students to better focus on the goals of the game.

3.1 SimulES-W Development

The development process of SimulES-W was performed by six steps as illustrated by the SADT diagram in Fig. **.** The first step was based on a review of the literature related to educational games as explained in [38]. In addition, we studied the available documentation about the previous versions to gain an initial knowledge about the game. This theoretical knowledge was allied with the practice of playing the game, still in its manual version was used when playing the game.

As such, we gained a new understanding about the game and the intentional model of the game was useful for that [32]. Intentional modeling, with i* [36] for instance, was chosen because it represents the interaction among players; other representation approaches researched did not consider this interaction. The modeling language used in [32] follows the ERi*c method [40] which helps to create i* models.

During the elicitation process we used i* models to represent the interaction among players, and to show the game dynamics.

For the creation of these models we adopted the ERi*c method [40] that builds these models using six steps, interconnected by means of requirements baseline. These steps are: (i) goal and actor elicitation, (ii) SDsituations identification, (iii) goal modeling for each actor, (iv) rationale modeling for each actor, (v) SDsituations specification, and (vi) analysis of SD and SR models as described by Monsalve et al. [38].

Each round of the game is represented in SDsituation, which represents in a structured diagram the situations of strategic interdependence between actors sharing a goal common within an organizational context. Monsalve et al. portray the round Play Round to Start in [38]. In Figure 2, the oval-shaped represents a goal that must be met and in which two actors interact, such as: Player in turn interacts with SimulES.

In the syntax i* this means that the actor Player depends on the actor SimulES (the system) to Game be started, the system available resources, publishes movements. In like manner SimulES depends on Administrator (role) to Entry of player be closed and type of cards be chosen. Also, Adversary depends on Player in turn to Project be accepted, as Player in turn depends on SE to be hired. In addition, the box-shaped figure represents resource; they may be read as follows: Player in turn depends on SimulES to available Dice. In Figure 2, the cloud-shaped represents a soft goal or a non-functional requirement and can be read as follows: Player in turn depends on SimulES to the game is availability.

Figure 3 presents the description about the model view controller architecture resuming the heuristics used to identify the elements of modeling that were subsequently used in the implementation. Summarizing, the behavior found in each SDsituation, presented in Figure 8, was implemented as user interfaces or related modules.

Symbols in Lexicon, which are a starting point to describe the vocabulary of the application, were modeled in i* diagrams as resources or actors and then they were mapped to the diagram class (Figure 4).

Goals and tasks represented in i* diagrams which were categorized as verb type symbols were implemented as methods.

ISSN 2007-9737

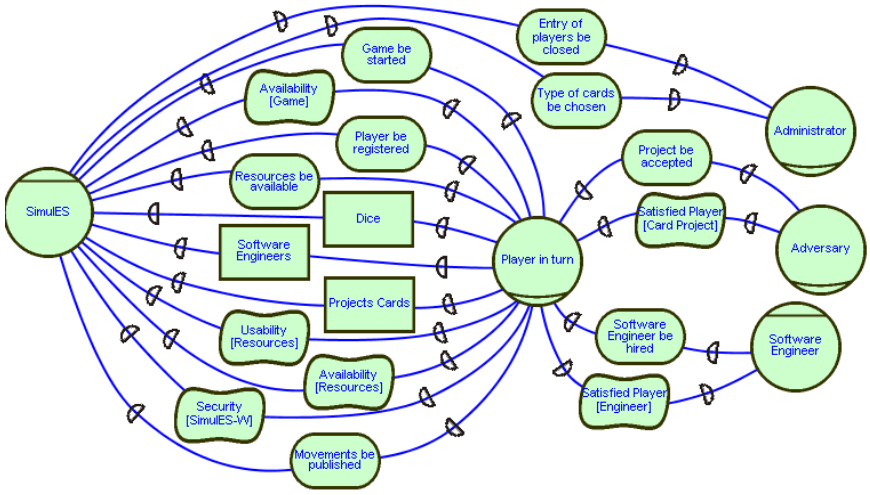


Fig. 2. SDsituation: Play round to start

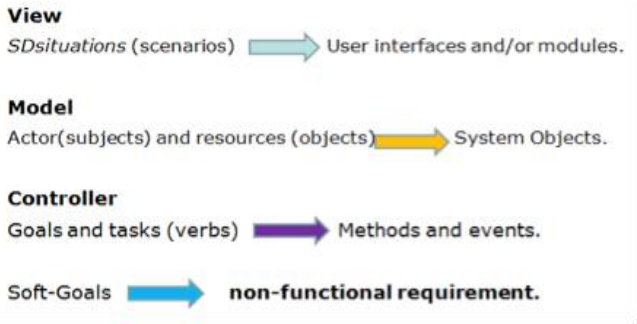


Fig. 3. General heuristics

Simultaneously, as we were using these models to develop SimulES-W, we did one experience with an interdisciplinary group what had interacted with SimulES. We used elicitation techniques, such as the observation of the players and questionnaires, to evolve the requirements.

SimulES-W was developed mainly in Java with the support of other software tools and frameworks such as MySQL, Hibernate, and Javascript. The base used for the implementation was the ERi*c's models which were mapped to a general MVC (Model View Controller) architecture [41] and the code was instrumented with scenarios describing i* tasks. This process is detailed in [38].

The development environment used was NetBeans version 6.5. Figure 5, shows the packages that separated each component of the MVC pattern (Control, Model, and View).

In addition, two additional packages were created in the group Servlets and Util Classes. The project was written on Java and Javascript.

As source code metrics, we presented Code lines by Packages: View: 18034, Control: 4292, Model: 1806, Util: 506, Servlet: 367.

Classes by packages: SimulES: 79, Model: 22, Control: 18, Util: 8, Servlet: 3. Total methods: 1326. Classes with the biggest number of methods: Index: 96, SourceCardsPage: 85, ProblemsByPlayerPage: 71, SubmitProblems Page: 66, SessionBean1: 59. Frameworks used jMaki Ajax, Visual Web JavaServer Faces, JavaServer Faces and Hibernate.

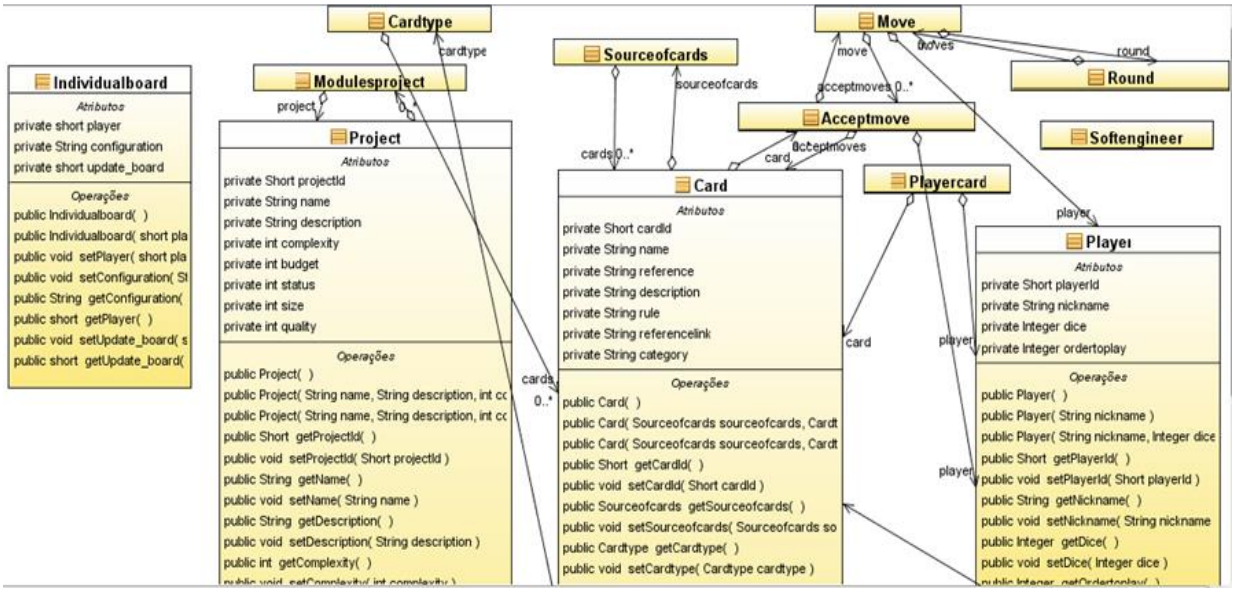


Fig. 4. Partial UML Class diagram of SimulES-W

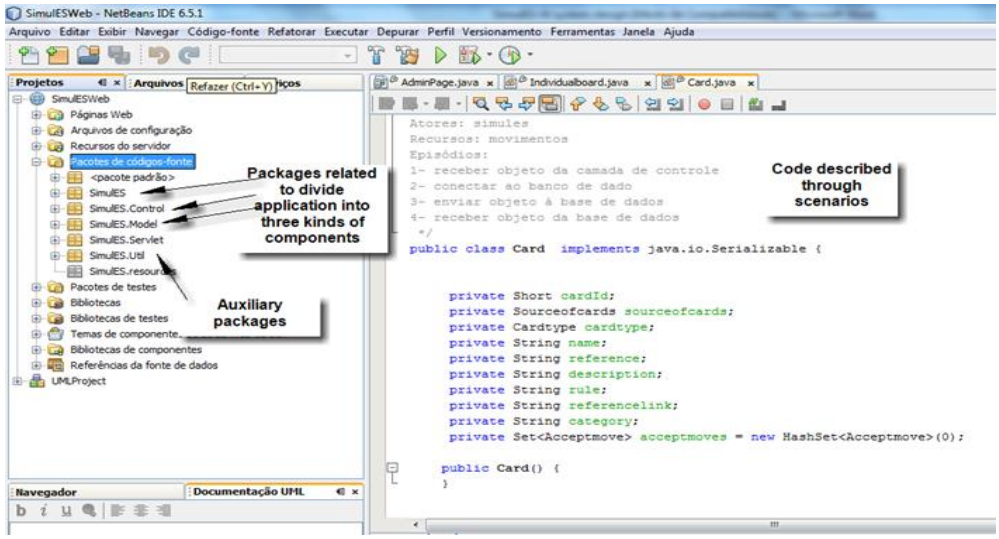


Fig. 5. SimulES-W packages

This code was documented using scenarios [35] and we used MySQL as a database server.

Figure 6 shows the main page of the game. This page exhibits the messages exchanged between players and system messages according to moves made in the game. These last messages are displayed when a player makes some moves in the game.

Furthermore, the chosen project and its modules are also displayed. Apart from that, a list of moves accepted between players is displayed in the bottom area of the screen as well as players that are online who are already registered in the game.

The version of SimulES-W presented in [38] is being packed and will be released as soon as open

ISSN 2007-9737

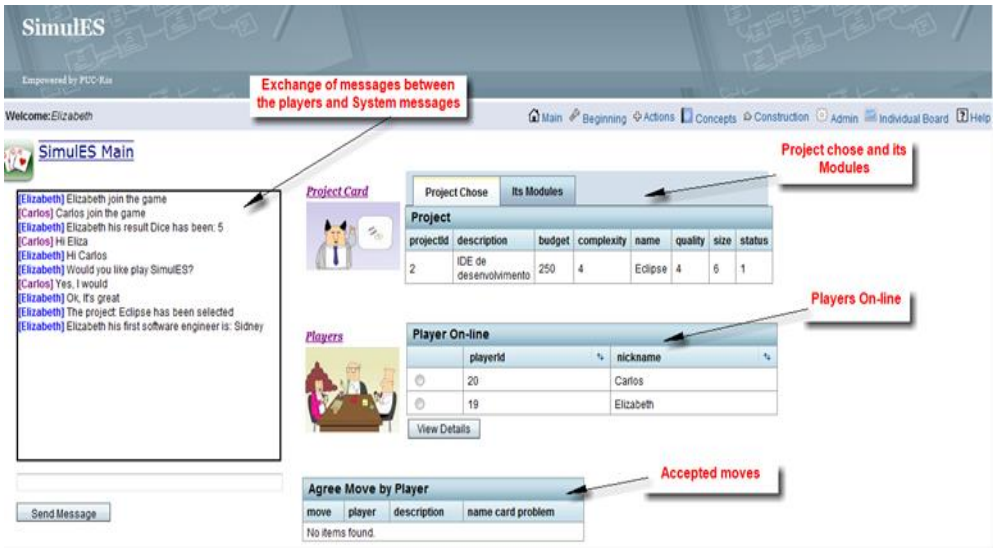


Fig. 6. Main Page of SimulES-W

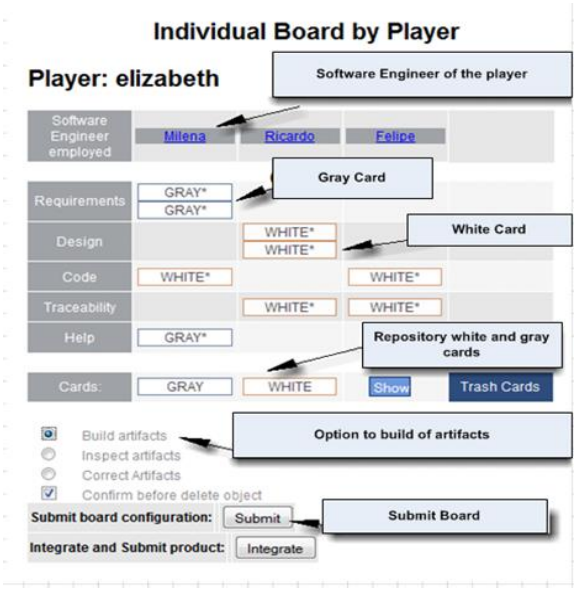


Fig. 7. Example of an individual board

source software. Being open software allows anyone to evolve it.

Furthermore, it will work as a web application and therefore it will not require downloads, installations or special configurations. However, the main advantage of SimulES-W is that it can be easily customized. That means, categorization can be incorporated; it will happen if the instructor chooses to approach the issue from a different angle or just want to stress specific topics.

For example, if the instructor identifies that the topic, which should be used in the training, is

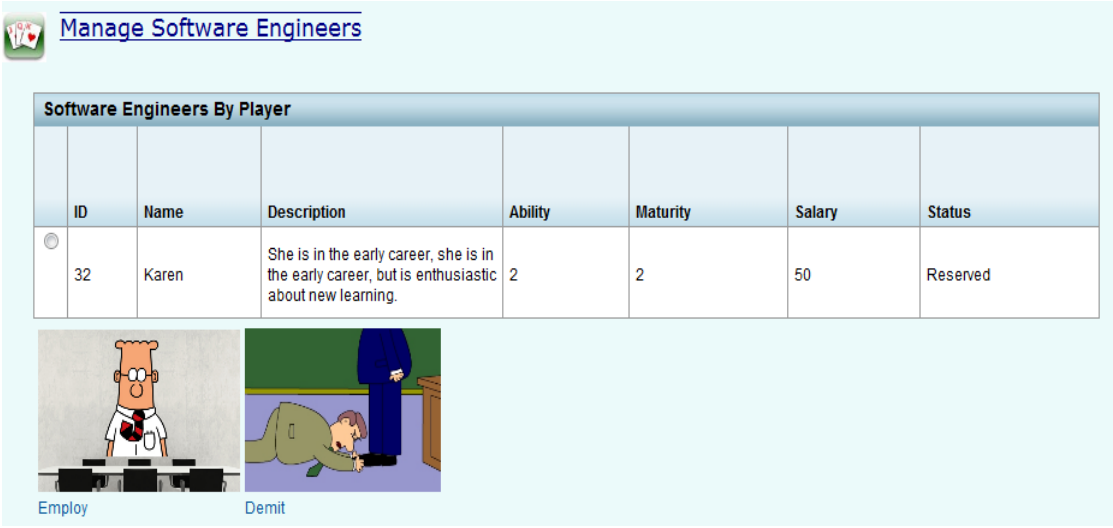


Fig. 8. Example of Project Card

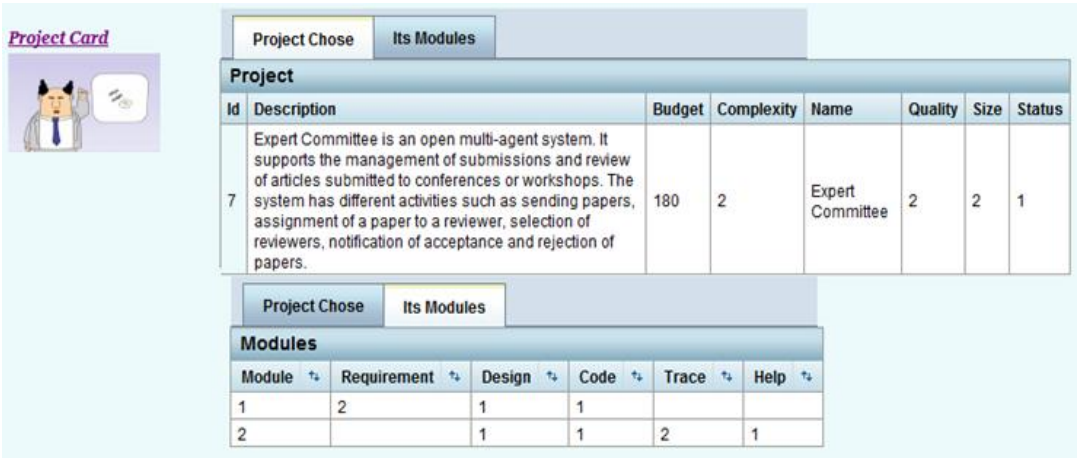


Fig. 9. Graphical interface to manage of software engineers cards

Requirements Engineering Management or *Software Verification and Validation,* then the instructor could produce cards to address the topic, and consequently the discussion generated when the players are interacting with the game will be the targeted one. In fact, the customization allows cards to be edited and stored in named files, so the game can be played with different emphasis as well as with different philosophies.

Other customization may be enacted, for example: changing the rate of white and gray cards or different calculations for time points, and analyze different players and tune the game accordingly with the observations found. Another advantage of SimulES-W is that concept cards and problems cards do handle links to web material, thus making possible direct access to bibliography or other resources. Having a direct access to supporting material is supposed to improve student performance, since the bibliography would be easily accessed in the right context.

This enables an even more targeted education, with a combination of cards and proper educational material.

ISSN 2007-9737

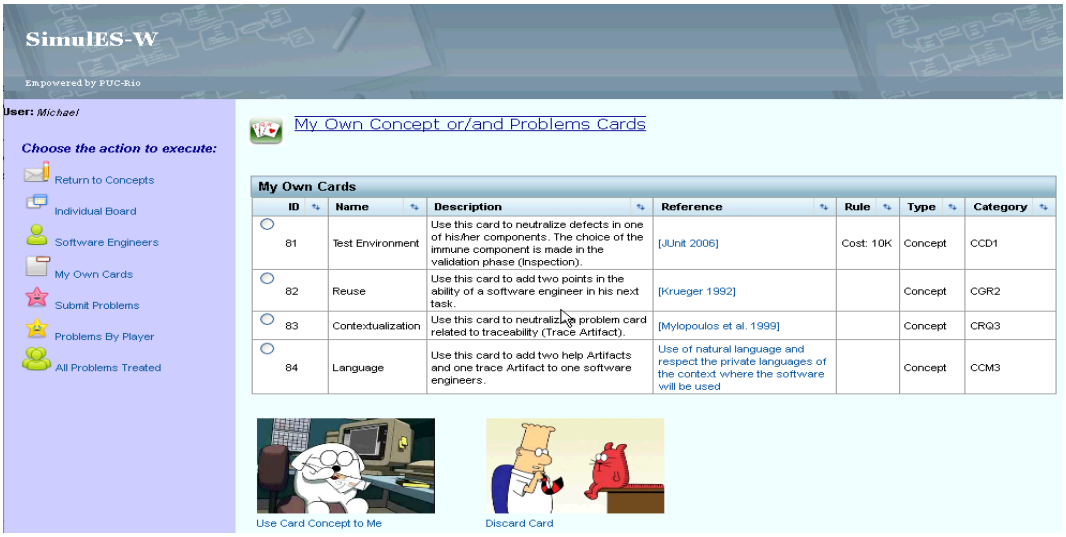


Fig. 10. Graphical interface to manage of Concept Cards

3.2 Game Components

The most important idea in SimulES-W is that it simulates the SE process where each player takes the role of a software project manager in his own game, consequently he has to: deal with budget, deal with SE employment, build different software artifacts and control its quality by inspecting them before submitting the final product.

During the game, other players, named adversaries, such that new problems may be assigned to disrupt the player´s situation in the game, can establish some situations.

It is also the case that the player analyses his own game and the others player's games to create a strategy of development, which allows one's player game to become a better game than of his adversaries.

This strategy is based on the analysis of the player's resources (problems cards and concept cards), the project budget and the skill that its software engineers have.

If he has a good and big team he can develop quickly and inspect all the artifacts latter or he can mix constructing artifacts and inspecting them.

The resources used during the game are: the main board (Figure 6), the individual boards (Figure 7), the project cards (Figure 8), the software engineer cards (Figure 9), the concept cards (Figure 10), the problem cards (Fig.), the white and the gray artifact cards (Figure 7) and the dice.

The Individual Board is the base for playing, this is the area of the game in which each player places their Software Engineers in the columns and the artifacts in the rows.

Each project requires constructing artifacts of the following types: Requirements, Design, Code, Traceability and Help. Figure 11 shows the Individual Board in a game scenario with three Software Engineers.

The artifacts are placed in cells of the board, below the software engineer who produced them and in rows for their types. These types represent the same type of real software project artifacts that must be built. At first when the game starts, the project has to be established and it will be available for all players.

This definition is made randomly so all players have to roll the dice and the player who got the highest dice result is the one who execute the action to get the project card and starts the game. This movement is important so all the players start to get involved in the game and starts to know SimulES-W interface and how the project characteristics will influence their moves. Figure 12 exemplifies a Project Card and its features, like description, complexity, size, quality, and budget and modules descriptions.

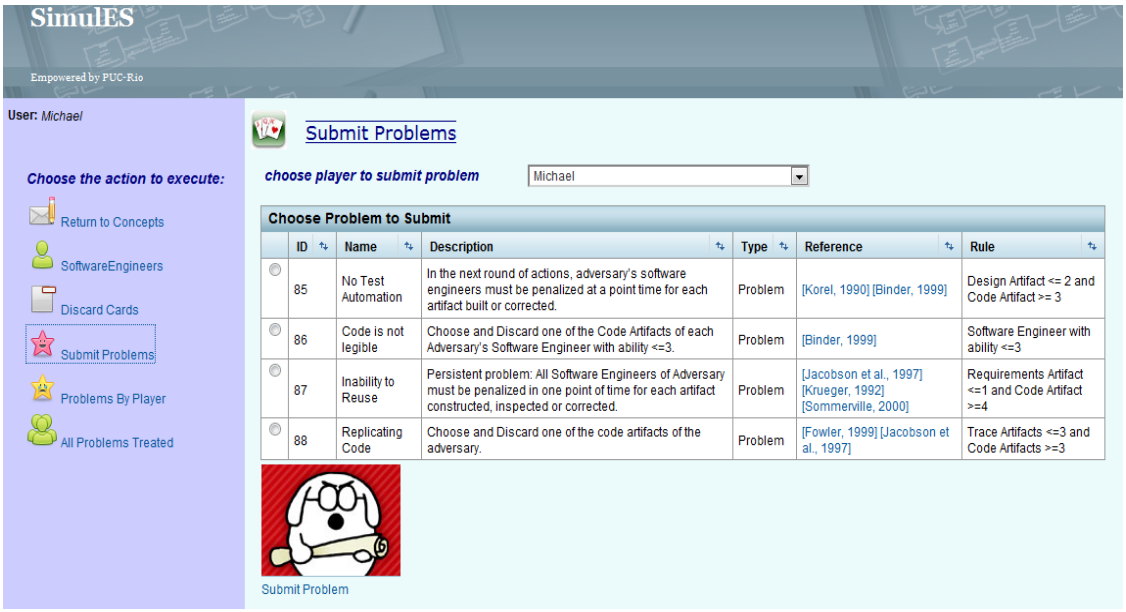


Fig. 11. Graphical interface to manage of Problems Cards

The project complexity is related to the white cards value, so the white cards will be worth the value, which is indicated in the complexity, and gray cards will be worth half of this value. The size denotes the number of modules to be built in each project and the quality shows the amount of error free modules that the final product must have to win the game.

The budget is the total project cash and defines the amount of money available to be spent.

Finally, the modules description describes the quantity and the type of modules to build on the individual board.

There is a round where each player throws the dice once. The dice result allows the player to draw white cards, gray cards and software engineers, if the dice upshot is less than three, then he will draw only concepts and problems cards according to this result. On the other hand, if the dice upshot is more than three he will draw concepts and problems cards and software-engineer cards.

The quantity of software engineer cards will be the difference between three and the dice result (dice upshot – 3). As an illustration, if Mary threw the dice and its result was 2, then Mary would draw 2 cards (problems and concepts). Nonetheless, if Mary threw the dice and its result was 4, then Mary would draw 3 cards (problems and concepts) and 1 software engineer card, these cards will be kept until that player needs them to make his move in the game, namely the player applies concepts or submits problems or places a software engineer card on the individual board.

Placing a software engineer (Figure 13), on the board (Figure 11) means that the player has hired a software engineer, but that action is limited to the project budget, since each hiring subtracts the salary of a software engineer from the total budget, so a new hiring is dependent on the number of software engineers already on the board and their salaries.

Fig. portrays a characteristic software engineer card and its features like name, description of his personality, salary, which is related to the budget, an ability, which is related to the project complexity, and a maturity, which is occasionally used in problem cards conditions.

A software engineer can execute one or more actions in the individual board (Figure 11) such as build artifacts, inspect artifacts, correct defects and integrate artifacts in the module. He does these actions according to his ability, a number (see Fig.), which determines how many moves he can do in each action and is related to the project complexity.

ISSN 2007-9737

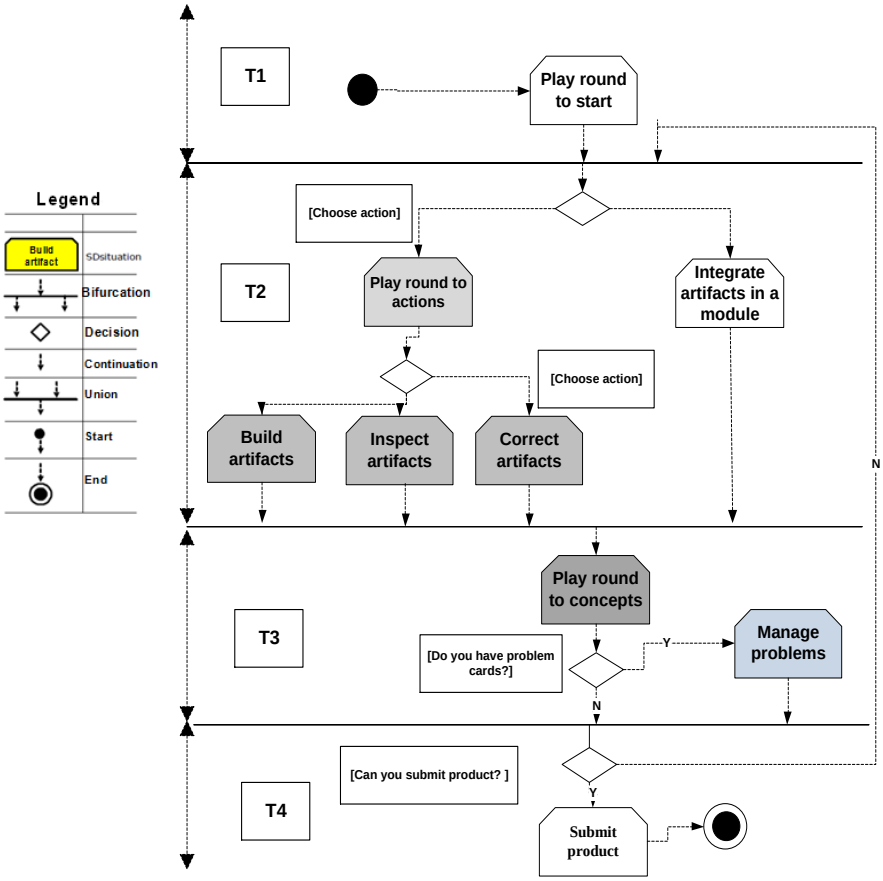


Fig.12. SimuES-W SDsituations - the rounds in the game [39]

It is also known as time points. For example, if the software engineer has an ability level of 2 and the project complexity is 2 he can build in each round one white card artifact.

The ability is the number of points of time (productivity) that the software engineer has to spend on each round, so it defines the number of white and gray artifacts cards that can be produced by him. In Figure 12, white artifact cards costs 2 and gray cards costs 1, so Karen (Figure 13) with the ability of 2 points of time and with the project complexity (2) can build 1 white artifact card or 2 gray artifact cards. Therefore, if the player has engineers with higher ability, then the player will have more productivity (will build more artifacts, and as such may finish earlier).

Each artifact is built either with a white or gray card. As shown in Figure 12, the quality attribute determines the number of modules that will have to be bug free, and the budget attribute defines the amount of money available to be spent in hiring software engineers.

Concept Cards and Problem Cards have a name, description and a reference. The reference shows the concept source and explains it in detail justifying its use. Thus, if the players are interested in some issues, then they could research more about it. The description in both type cards (Concept Cards and Problem Cards) describes how cards have to be used, some of them may also have a cost associated with it.

As a result, the player can use the Concept Card not only to block a Problem Card, but he can also use it to improve the player´s performance. Hence, both types of cards have the sufficient

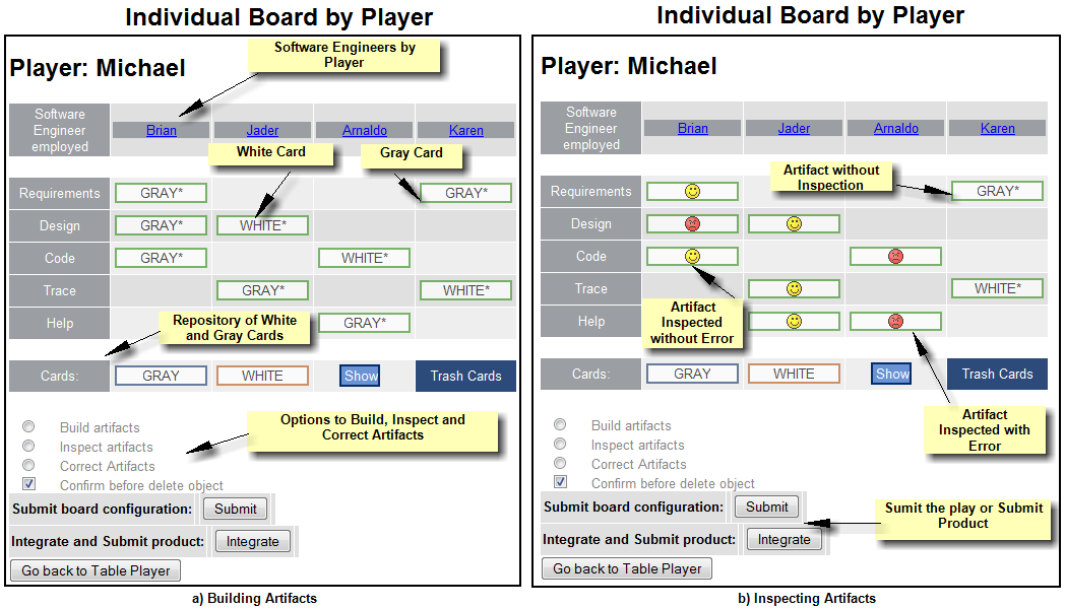


Fig. 13. Graphical interface to manage Individual Board

information to present a concept or highlight a common problem.

3.3 The Game Rounds

SimulES-W has different types rounds where players execute their moves such as: Start, Concept and Manage problems, Actions (Build, Inspect or Correct artifacts and Integrate Artifacts into a Module), and Submit product. Figure 12, an SDSituation diagram of the ERi*c method [40] illustrates these rounds in a time-oriented flow chart.

When the game starts, one project must be chosen from those available [T1] in Figure 12. The dice is rolled and the one who gets the highest dice result chooses the project and starts the game. Furthermore, the information about the project is displayed to all players.

After that, each player assembles an individual board and picks up one software engineer in the stack of SE cards.

In the “play round to actions” [T2], each player with the information of his/her software engineers (ability and salary) and the information in the project card (size, complexity and budget) uses a software engineer to: build artifacts, Inspect artifacts, correct artifacts and Integrate artifacts in a module (see [T2] in Figure 12). In the action build artifact, if the player builds with white artifact cards, he/she will spend the points of time as per the complexity in the project card, but if he builds with gray cards then he/she will spend half of the points of time.

However, white artifact cards (5 cards to 1 defect) have a lesser defect rate than gray artifacts cards (3 cards to 2 defects). Inspect artifact is an action of turning up an artifact card under the responsibility of a software engineer, disclosing its quality status (with or without a bug – see (Figure 7). The cost of inspection is fixed by 1 point of time per card if it is performed by the same software engineer that built the artifact and 2 if it is performed by another software engineer. Correct defect action has to be performed when the software engineer inspects an artifact card and finds a defect (“bug”).

By correcting a defect, he/she spends 1 point of time if it is performed by the same software engineer that built the artifact and 2 if it is performed by another software engineer. Integrate Artifacts in a Module action has to be performed before the player submits the product. This

situation happens when the player has built all types of artifacts required in a module (Figure 8).

The player can choose the artifacts that are available in his/her individual board, considering the artifacts types described in the project card to compose a module.

The artifacts can be originated from different software engineers (columns in the individual board).

In the "play round to concepts" [T3] in Figure 12, each player rolls the dice once. The dice result allows the player to draw concept/problem cards. These cards (concepts and problems) are shuffled together and piled upside down in the main board. If the dice shows a number greater or equal to 3, then software engineering cards may also be drawn. The quantity of software engineer cards will be the difference between three and the dice result (roll of the dice – 3). Thus, the greater the result from rolling the dice, the more resources the player will have.Here is where luck comes into play.

At this point, the player has to think about team composition: the number of software engineers is limited to the overall budget (see Figure 4), that is the sum of the salaries of the software engineers posed in the Individual Board. This implies the possibility of hiring and firing software engineers (project management skills). Note that there is an educational purpose of making students deal with real world issues (hiring/firing) by means of simulation.

In [T3] the player uses concept and problem cards. So during the game, the player can receive problem cards from the other players. These cards, when received, are to be used in the next round. The objective is to damage the game of the other. However, if the player has one card, which invalidates some problem cards (a concept card), the player will be able to use it and the action described in the problem card will not affect his game. Then he must discard both cards. On the other hand, if he does not have any card that invalidates the problem cards, this problem will be applied to his game. At this point in the game, the educational goal is that the players discuss both problem and concept cards claimed to pair. A player using a concept card has to build an argument as why that card neutralizes the problem card.

This argument can be discussed, but it will only take effect if all players agree. As mentioned before, this discussion can be mediated by an instructor. The "submit round" [T4] can be performed in the beginning of the player turn. When the player integrates all his modules, he can submit the product.

Then the other players have to inspect those artifacts that are not inspected (faced up). The product will be accepted, and wins the game, if there are *n* number of modules free of bugs, whereas *n* is indicated by the Quality attribute in the Project card Figure 8.

Figure 13, shows two different situations in The Individual Board: a) Build Artifact and b) Inspect Artifact. At the top the Software Engineers who is employed by the player. The Individual Board (Figure 13 a) portraits the White artifact Cards and the Gray artifact Cards. It illustrates cards, which have not been inspected yet. Alike, the part b of Figure 13 shows when artifacts have already been inspected and also the result of the inspection. This figure also shows when the artifact has a defect (bug) or not. This result is chosen randomly by SimulES-W and is based on the rate of defects mentioned before. Both sides part a and b at the bottom of this figure present the different operations, described above, that can be executed by the player based on his own board.

Each player would use his knowledge about SE in Play Round to Concepts and Manage Problems. This type of round is where he can throw problem cards (Figure 12) to another player, treat problems and improve his game if possible.

This round allows the player to use Concept Cards and Problem Cards which describe a typical process, good practice or problem in a specific area of SE.

Figure 10, portrays some typical processes and tasks of SE. These cards can be used by player to improve his game or block some problem that he has. After that the player analyzes his own game, looks if he has problems to deal with that have been submitted by other players and if some card could be used to treat or invalidate these problems. In similar fashion there are concept cards that can be used at any time and that can improve his game giving advantage over his adversaries. To illustrate it, there are concepts which enable to increase the budget, solve problems related to inspection, solve

a specific problem and improve the ability of engineers to better build the software product that is asked in a more efficient way.

Problems Cards, shown in Figure 11, represent typical problems in SE process. This is the view of the player when his adversaries have submitted problems to him. The objective is to damage the game of the other and to try to win the game. [T3] above describes the mechanics of this type of round.

By the end, if player understands what the cards means and has the opportunity to use them, then he will apply the Problems Cards to halt the adversaries' progress.

For this reason, the proper use of these cards could lead to the creation of a good strategy game that is reflected in the good performance of the player. This could lead to winning, but it is clear that there is a component related to the fate of the player and he has to have the right cards and know how to use them at the right time.

4 Learning Software Engineering with SimulES-W

There are two important issues that are taught through SimulES-W: one is the dynamics of software construction and the other is related to problems and concepts about SE in an interactive way.

When a student or player is building his software product on his individual board, he will be able to choose a way to address his game, namely he has different ways to organize his activities or to establish some tactic to conduct activities. In other words, he can choose his own strategy to assemble his product, which means that he can choose when to address quality issues of his cards (inspect and correct actions).

This situation is explained in [30] SimulES implement a dynamic software development process that mean, it is geared towards evolution, and does not preach a fixed process with well-defined and fixed stages. SimulES-W strategy also avoids the concept of complete requirements, which in reality is a misconception that is repeated in several teaching materials.

As a result, SimulES-W presents an open development process that is dynamic and not structured so the student can learn that different context implies different strategies.

Furthermore, it is possible to refine, inspect or correct any artifact or artifacts. In other words, the dynamic nature of the process in SimulES-W allows the student to realize the evolving nature of the artifacts in a software project, mainly the requirements. Therefore, requirements are present at any stage of the development process and they could be tackled whenever necessary.

This concept is taught to students in the game when they have to build their software product, showing this as an open option where they choose a way to build and their own strategy. However, the teacher should highlight the importance of the requirements in all stages of development. In fact, players can suffer a penalty if they do not regard the requirements in the development process.

5 SimulES-W Experience

5.1 A Proposed Process to Teach with SimulES-W Didactic Concepts

As described in [37] the proposed educational of SimulES-W is based on collaborative experience. This means that teachers address experimental learning to a specific issue and trainees or students should be able to learn through discussions and decisions making techniques which are represented by the game rounds. All the time, the players are interacting with each other. In the same way, these actions will be reflected in the final result of each student's game. We strongly recommend teachers or trainers to offer a previous training about the game dynamics and how to use the software.

Alike, when the experience has finished, evaluations should be applied, one to evaluate the software acceptance and other to evaluate concepts that were learnt.

The teacher is the one who decides these contents according to what issue is necessary to teach. In Figure 14, we propose a didactical process that we established in order to apply SimulES-W as an education tool. The following parts were identified: Prepare Train, Execute, Feedback and Analyze.

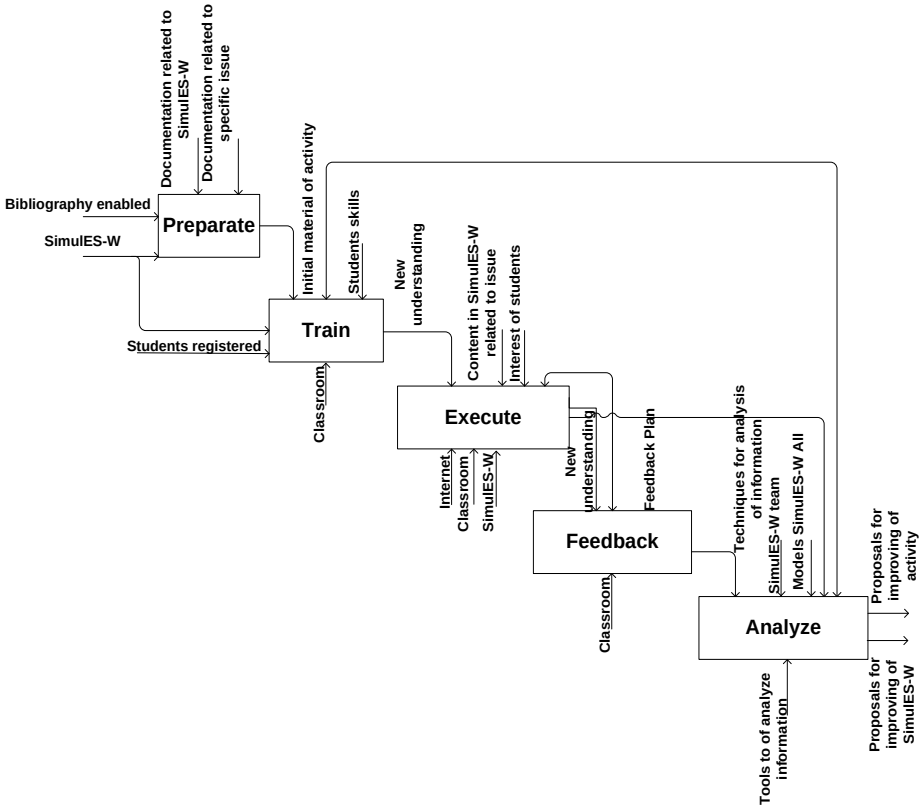


Fig. 14. A SADT model of the process we propose to use SimulES-W

(i) *Prepare***:** the teacher should analyze some features related to the students' skill and goals of the course, like interests, previous knowledge, and importance of the issue in the course. After that, the teacher chooses the specific issue, collects, and prepares the information related by creating specific concept and problems cards. The project and software engineer's cards can also be edited to specific classes characteristics. Afterwards, the contents are added into the SimulES-W database. Finally, some test should be made to check the contents. Of course, that courses could be reused, so in that case there will be needed for customization, but just the selection of previous produced material.

(ii) *Train*: students receive information related to SimulES-W origin of the game, historical review, basic rules, dynamic rules, goals and main screens. After that, students use the tool. Instructions about the use of the tool, including navigation, interface features and execution of actions, are explained. All in all, the activity (play the game) could begin.

(iii) *Execute:* we recommend the activity to take place in a classroom situation with a teacher or instructor who guides the students and who answer questions related to the activity. SimulES-W allows to at least two players and no more than five players play online at the same game. That means that if there are more than that amount, students should be arranged within groups not exceeding five, because many online players could slow down each round and, consequently, the activity. During the activity the teacher emphasizes the concepts and problems addressed in the game and discuss with the students each round of all players. Each student should participate in the turn of their

Table 3. Closed Questions Related to SimulES-W as a Tool

1) How does the game SimulES-W is usable from perspective of a software system?

(1) Easy to use (2) Usable, (3) Reasonable (4) Little usable (5) Hard to use

N	Mean	Median	Standard Deviation	Upper 95% Mean	Lower 95% Mean
14	2.1428571	2	0.9972489	2.73589	1.5498243

2) Accordingly to student's point of view, how SimulES-W game is?

(1) Motivating (2) Interesting (3) Neutral (4) Tiring (5) Nothing to do

N	Mean	Median	Standard Deviation	Upper 95% Mean	Lower 95% Mean
14	1.9285714	2	0,7300459	2,3500874	1.5070555

3) In your role as student, how SimulES-W game is?

(1) Teaches (2) Informs (3) Neutral (4) Distracts (5) Disturb

N	Mean	Median	Standard Deviation	Upper 95% Mean	Lower 95% Mean
14	1.5714286	1,5	0.6462062	1.9445369	1.1983202

4) According to your own opinion, does SimulES-W show the dynamics of a software project?

Value	N	Percentage
Yes (1)	11	78.571%
No (0)	3	21.429%
Total	14	1.00000

N	Mean	Median	Standard Deviation
14	0.6923077	1	0..480384

adversaries to agree or not based in contents of problem and concept cards.

(iv) *Feedback:* we recommend two kinds of feedback of information to be collected. One related to the tool, such as user experience, expectations, improvement suggestions, navigation aspects, appearance, tasks execution and strengths and weaknesses and the second one related to concepts addressed in the activity, and specific questions about topics taught. In this manner, information collected could be used to improve not only the process around the activity but also the tool.

(v) *Analyze*: the teacher should address the feedback and clear some misconceptions or loose ends identified. With respect to tool feedback, this should be directed to the developers to further improvement.

5.2 An Assessment Scenario

In this section, we present a comparison of two education strategies for a part of a SE

undergraduate course. The focus of our assessment relies on the part of the course that deals with Software Risk Management (SRM). One approach was just based on lectures, and the other was the use of the SimulES-W game to present the SRM material together with lectures. We report on the information collected and lessons learned from this experience.

This activity took place in the second semester of 2014; at the *Universidade do Estado do Rio de Janeiro*, Brazil. 37 students attended this activity divided into two groups. The division was made alphabetically. The first group (14 students) had one class using SimulES-W.

After that, in another class they answered a survey and attended to a SRM lecture. Another group (23 Students) only attended the SRM lecture. Each activity lasted 90 minutes. Both groups were tested by a software engineering exam with one question on SRM.

At first, to prepare the SRM contents in SimulES-W, two co-authors read [42] Chapter 5 and studied how SRM involves identifying risks and developing plans to minimize the risk effect on software projects, stressing the tasks of risk analysis, risk planning, risk identification and monitoring. These tasks involve other activities like: identify project, product and business risks, evaluate the probability and consequences of risk, prioritized list of risks and also if risk could be known, predictable, unpredictable; risk type like project, product and business and related to technology, people, organizational, tools, requirements and estimation.

Once knowledge had been obtained on the subject, we divided the activities to create problem cards and concept cards to SimulES-W based on software engineering contents. Three people (one teacher and two graduate students) were responsible for the activity. The students made the problems and concept cards to the game; these cards were created and separated by subject in the book. After that, the teacher according to his experience and the reference book revised and validated the cards. They were refined and a final version was available. Finally, these contents were incorporated in the SimulES-W database.

Then the contents were analyzed by means of tests. The SimulES-W was ready to the activity with the students.

5.3 Data Collection (Feedback)

To assess the results of the experience a survey was designed using closed and opened questions to measure perception. In the first group of questions (closed) the students were asked to give a grade to some issues about SimulES-W usability, motivation, role in learning and if SimulES-W shows the dynamic of software development process.

Table 3 summarizes the answers by giving the statistical numbers and Figure 15, shows in a graph how the answers were distributed to the different questions. Questions 1, 2 and 3 used a Linkert scale and question 4 was a binary question.

For the first question proposed to the students: how do you perceive SimulES-W usability? As presented in Figure 15, the most of them report that SimulES-W is perceived as easy to use (57%), 22% of them claim that they find it easy to use. A lesser proportion of students claim that it is reasonable (15%) while 7% declare that it is hard to use. The results suggest that the game according to the appreciation of the players can become usable therefore; this could be a proposal to support software engineering class.

The statistics of Table 3 suggest positive results, given the median, the mean and the standard deviation, so: second question of Table 3 is related to motivation, that is, if the student found motivating or tiring the game. Figure 15 shows that most of them find SimulES-W interesting (72%), motivating (21%) and 7% of them report that it is tiring. Even with the positive result, we believe that it is necessary to better understand this question. For that reason, new studies are necessary. In a similar fashion, the median and the mean confirm the trend, thus, median with value 2 points to the answer Interesting, the mean shows the inclination for that answer (1,929) and standard deviation shows how far the lie from the mean.

Question 3 (Table 3) portrays the result related to how students perceived the role of the game, if it is oriented to teaching or otherwise if it disturbs. The result shows that most of the students agree that SimulES-W (Figure 15) teaches (50%) or informs (43%). Nevertheless, there are 7% who think that SimulES-W is neutral.

We think that this experience with SimulES-W was positive, and the result supports this thought.

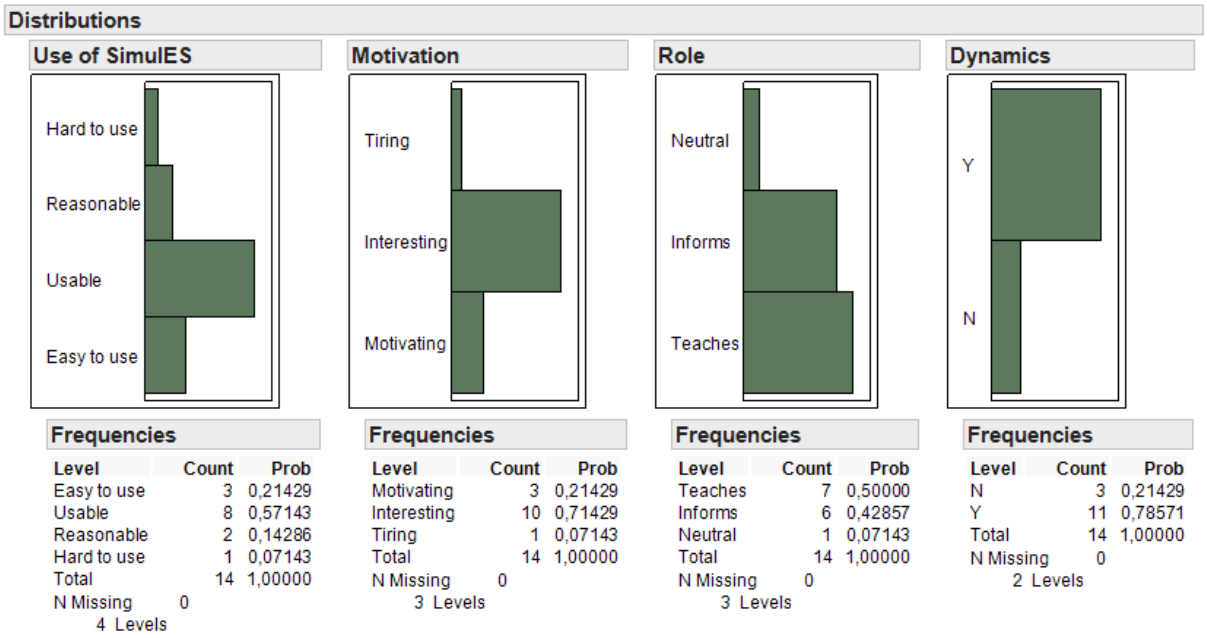


Fig. 15. Answers of Closed Questions Related to the use of SimulES-W by UERJ Students

In a similar fashion, the median with value 1.5 refers to middle value between teaches and informs also mean shows the intermediate trend for those answers (1.571) and finally the value 0.646 as standard deviation shows how far the values lie from the mean. Finally, Question 4 (Table 3) enables us to identify if students recognize the dynamic of a software project in SimulES-W. The majority of the answers shows that the students recognized the existence of software process as the basis for the game (Figure 11), (79%), followed by a group who reports that they do not identify the notion of process in SimulES-W (22%). For this question we have interviewed some of the respondents and found out that of those who answered yes, found that the notion of a software project is clear, as well as the process for producing artifacts; for those who answered no, there was a consensus that the notions are still provided in a superficial way and it gives just a rough idea.

The second set of questions (open) was designed to gather student's opinion in a free style text. With these questions we collected information related to problem and concept cards, strengths and weaknesses of SimulES-W and information about aspects of navigation, appearance and task execution. These questions are shown in Table 4.

In the fifth question, "What do you think about problem cards?" most of respondents answered that the goal is clear and that they represent real problems in software engineering.

Even thought, it is necessary to increase the number of problems cards so the game will be more interesting.

In the sixth question, "What do you think about concept cards?" most of respondents think that the goal is clearer than problem cards. "Concept cards are clear, when they did not apply to specific problem, they were useful to exemplify situations." On the other hand, some students claimed that the game was too fast, and that for that reason it was impossible that some of them could recognize the cards quality and their details, as well as claims that some cards were hard to understand.

There are many positive aspects that were collected with question number seven, "In your opinion, what are the strengths of SimulES-W" Some of them were: "SimulES-W encourages me to learn software engineering.", "it was a challenge that SimulES-W gets to simulate a software engineering environmental."

Table 4. Open Questions used in Activity with SimulES-W to evaluate the tool

Open Questions Related to SimulES-W as a tool
5) What do you think about problem cards? Is the cards goal clear?
6) What do you think about concept cards? Is the cards goal clear?
7) In your opinion, what are the strengths of SimulES-W?
8) In your opinion, what are the weaknesses of SimulES-W?
9) What aspects of navigation, appearance and task execution would you like to see performed better?

Table 5. Survey related to Contents and Learning Elements.

Survey related to Contents and Learning Elements
1) According to your own opinion, what concepts 2) did SimulES-W teach?
2) How were these concepts approached by the game?
3) According to your own opinion, what did conditions 4) Lead to lose or win the game?
5) According to your own opinion, does SimulES-W 6) Simulate a real situation?
5) Identify the risks of a software project.

"With SimulES-W it was possible to play in group and discussions are brought up.", "it is possible to recognize the value of quality and the quality of software engineers in the software project.", "this game tries to show stages in a software project.", "with this kind of game is possible that class in software engineering becomes more practical."

The eighth question, "In your opinion, what are the weaknesses of SimulES-W?" allowed us to identify aspects to improve in SimulES-W. Naming only a few: "there are many actions that are based on good luck", students think that many actions in game should not depend on the dice, that it is required to improve performance and response time of the software, students suggest developing some strategy to improve the time that students wait between each round. Students propose to spend more time in training so they would understand better the software. It is necessary to improve the software navigation, interface and layout, create more alert messages being activated when users execute some action.

Students have also been asked to tell us more about contents and learning elements that were passed on SimulES-W (Table 5).

The goal with this part of the survey was to target a more depth on student´s understanding that is if SimulES-W communicated the contents planned.

In the first question, most of the respondents agree that concepts were taught related to typical problems in software development project combined with requirements, prototyping, and risk analysis. In a similar fashion, respondents think that SimulES-W teaches how the different tasks in software construction are important and some techniques to approach these tasks. It is also the case that students understand the importance of software engineers to the projects. One student claimed that he did not remember what was thought and other said that SimulES-W does not achieve the goal proposed.

In the second question, most of the respondents answered that the concepts were passed through the concept and problem cards,

and by the Project manager tasks of hiring and budgeting.

The third question, “According to your own opinion, what conditions lead to lose or win the game?” most of respondents said that luck was the main condition to win or lose the game, other students said that it was necessary more time to end the game. In addition, students also believe that inexperience, shallow knowledge about project management and knowledge on decision making were important factors.

The fourth question, “According to your own opinion, does SimulES-W simulate a real situation?” most of respondents said yes (64%), furthermore they complemented the answer saying that it is necessary in software projects some kinds of strategy and to be aware that problems can occur. They also mentioned that the tasks of software engineers are clear, and also that some of the problems that may occur in real projects are presented, as in the case of dealing with personnel and budgeting. Students who answered negatively (29%) had problems with the problem cards, either because they failed to apply them or because they did not see them fit in their game. They also pointed out that the game is superficial since they did not use the concept or problem cards as expected, they consider that the game does not provide a real SE situation.

5.4 The Software Engineering Exam

For the sake of assessment of using SimulES-W we compared two groups of students, one using SimulES-W and the other with just regular lectures. An exam was used to test students of both groups.

As such, in this part, we included an analysis related to the exam applied after SimulES-W activity. The idea with this analysis was to look through the exams and find useful information related to the usage of SimulES-W. That means, our analysis is basically centered on identifying the student´s performance on questions related mainly to risk analysis, which was the emphasis given in the SimulES-W activity as well as their general performance. Finally, we compared the performance between students who attended the activity (playing with SimulES-W) with those who did not participate.

The exam had five questions and each question had different values.

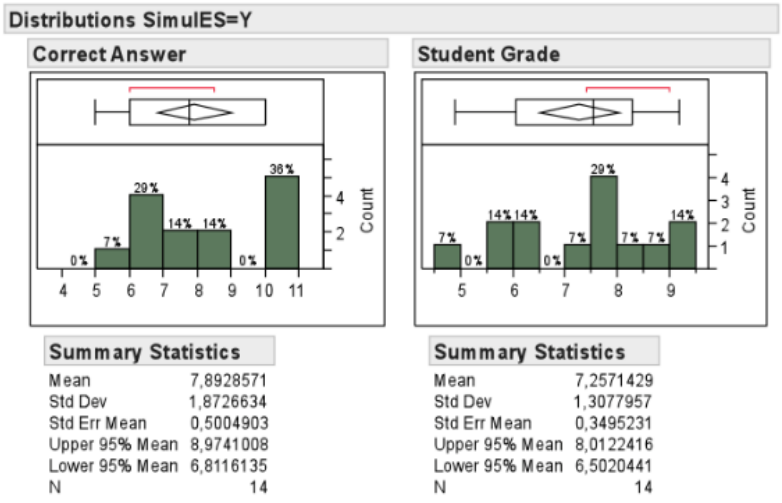
1. The first question the students suggested a software architecture for one specific problem, describing and listing the criteria which justified the choice. The value question was 3.0 points.
2. The second question was related to testing and students had to explain a sentence describing why *testing is not sufficient to* ensure software quality. To be fair, they had to describe techniques, which could be used towards quality control. The value question was 2.0 points.
3. The third was addressed to students classified a list into the corrective, adaptive and evolutionary maintenance of software. The value question was 1.0 point.
4. The fourth, students were asked to explain why the quality software depends on the quality of software development process. The value question was 2.0 points.
5. And finally, the fifth question was related to risk management. Here students had to identify two risks, give two examples and explain strategies that could help to mitigate risks. The question valued 2.0 points.

Although just the last question was geared towards SRM, questions two and four are topics dealt by the dynamics of SimulES-W with type of artifacts and verification by inspection. The exam was a general one, but will be used in our assessment since part of its questions was dealt by the game. It is also the case that we are not doing a direct comparison of learning with or without the game.

We understand this as an advantage of our assessment, since we using the exam as a confirmation of the survey result and not as the prime result, which would be required a more complex assessment structure.

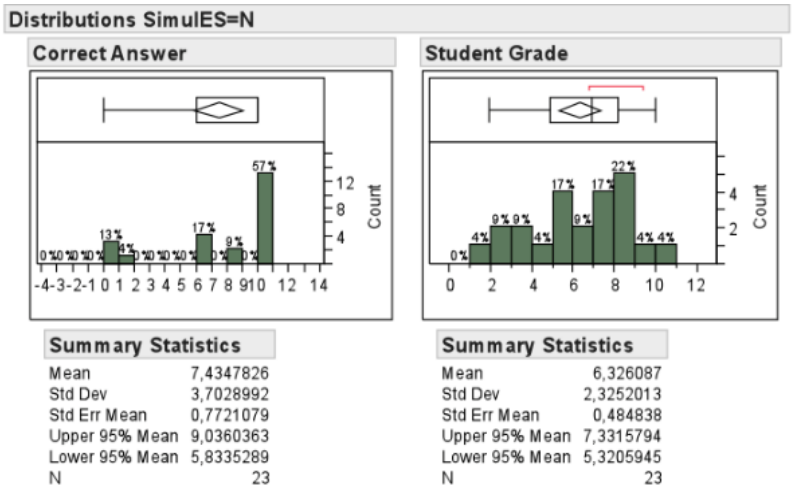
As such the hypothesis we are trying to check with the exam is if the students who used SimulES-W had at least equivalent learning results as the ones that used only traditional methods [14].

So, if there are no disadvantages for the learners who used the game, then we may assume



a b

Fig. 16. Results related to Exam Applied after SimulES-W Activity (SimulES=Y)



a b

Fig. 17. Result related to Exam Applied after SimulES-W Activity (SimulES=N)

that using SimulES-W does not influence learning in a negatively way.

All the students in the class took the exam. As noted earlier, the exam was composed by five questions and one question was about SMR, the target issue of this SimulES-W cards.

A total of 37 students took the exam, in which 23 had not played the game and 14 who did. Figure 16, a) (SimulES=Y) shows the success rate of the risk management question for the students who played the game. The figure portraits that five of the fourteen students achieved 100% of the answer this represents 36% of this group, one achieved 85% and this represents 7%, one achieved 80% this represents 7%, two students achieved 75% and this represents 14%, four students achieved 60% and this represents 29%, finally one student achieved 50% and this

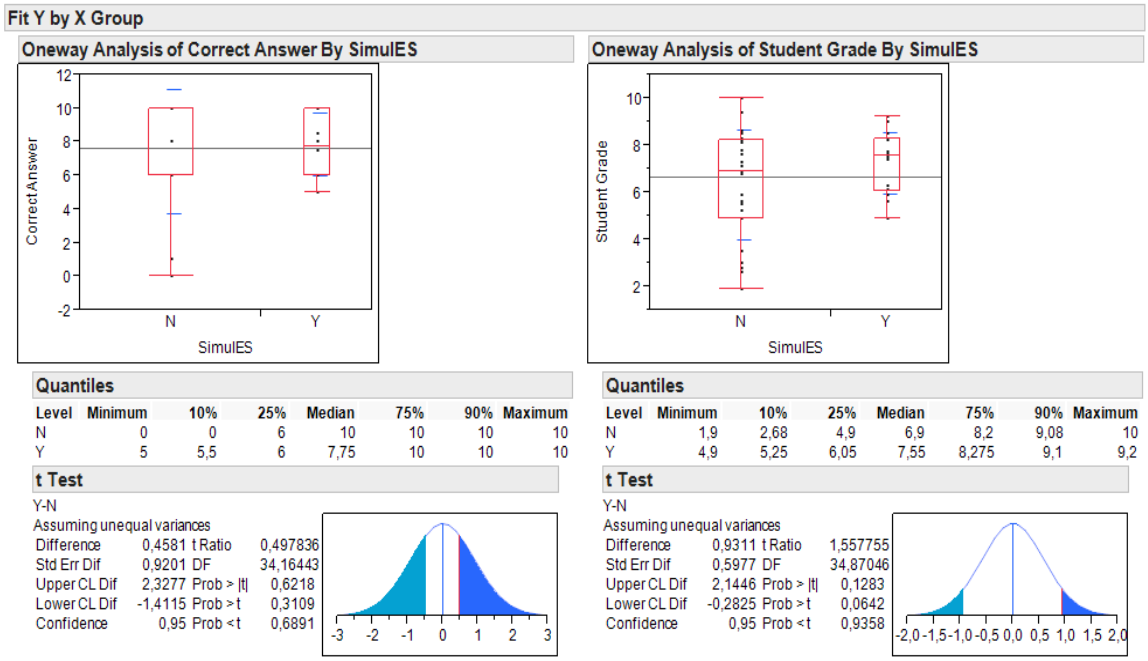


Fig. 18. Result related to Exam Applied after SimulES-W Activity

represents 7%. So, the majority of students got at least 60% of the questions.

Figure 17, part a (SimulES=N) shows the success rate of the risk management question for the students who did not play the game; of those (23), thirteen achieved 100% of the question and this represents 57% of them, alike two students achieved 75% and 70% this represents 9%, four students achieved 60% and this represented 17%, one student achieved 40% and this represented 4%, finally three students failed to adequately answer the question (13%).

Comparing the mean of the two groups we see that students who played the game got a little better degree than students who did not play the game because none of them get lower than the average (50%).

However, the mean number are very close (7,893 and 7,435) although the standard deviation has great differences from 1,873 to 3.703, showing grades that were more consistent within the group that played the game.

None of the students who played the game had a performance below the average (5.0) in risk management in software development. In contrast, the other group had four students below the average which represents 17% of the group. This is an important observation since not only SimulES-W did not negatively interfere on the learning process, but, it also showed that the players of the game had better grades overall.

If the whole exam is taken into account, Figure 16, part b (SimulES=Y) shows that most of the students (64%) who played with SimulES-W got a satisfactory score (Grade more than 7). However, 28% of them got a medium score (5.5 and 6) and only one student who did not answer the question of risk management in an adequate way (50%) got a grade (4.5) representing 7% of the sample.

In the same way, the Figure 17, part b (SimulES=N) shows that only 47% of the students got satisfactory score (Grade more than 7), 9% of the students got medium scores (6.9 and 6.8), 17% students got grades in the limit of the average (5) and 26% of the students fail the exam getting scores lower than five.

Comparing the mean of the two groups we see that students who played with SimulES-W got a little better degree than students who did not participate because the mean numbers for each

group were 7,257 and 6,326 and the standard deviation have great differences from 1,308 to 2,325. From this data, it is clear that there was no disadvantage for the students who used the game.

In Figure 18 we present the results of a paired-samples t-test with a Confidence Interval of 95% (α=0.025). The t-test was conducted to compare the hypothesis that the two means for the correct answer of the Risk Management question and the

Exam results were different for the two groups of students (students who played SimulES-W and students who only attended to traditional classes). The first result is that both groups can be considered to have the same distributions scores for the answer to the question about Management Risk because the t-test in a confidence range of 95% proved the hypothesis that the two groups have the same average and so there is no difference between the values answer questions of these two groups because the t-Ratio is equal 0,498 that should be less than 0,689. Otherwise considering the Grades of the Exam t-test rejects the hypothesis of having the same means for the two groups, the t-ratio 1,558 is greater than 0,936. So, this experiment evidences that students Exam Grades that used the SimulES-W had greater grades then the ones who did not use the game.

6 Conclusion

We presented in this paper a game-based approach for teaching SE and an assessment of its impact. In this game, the player explores different roles such as software engineer, technical coordinator and project manager and quality controller. Players have to deal with budget, software engineer employment, and construction of different software artifacts into the different scenarios named rounds. The elements are presented in the form of five different cards (artifacts, software engineers, project, concept and problems) and are available on two kinds of boards, the main board, which shows the project to be developed in this game, and the players' moves and individual board that represents each player team and products developed by the player.

This paper reports on how SimulES-W could be used in a SE course as a tool to improve learning. The results of the quasi-experiment carried out with two groups of 14 students each show a positive but statistically inconclusive improvement of students' grade performance, this experience also confirmed what others have reported on how games establish a motivating and enjoyable environment for students. Literature also has pointed out that games do generate knowledge as well as problem solving abilities and create strategies in a broader context. For example, if students understand the need and importance of software engineering in software projects they also will appreciate and use this knowledge when they face real projects. In addition, in this case with SimulES-W, the results indicate the trend that students could improve their grade if they are motivated by the use of games.

So, we have shown that the use of SimulES-W at least does not compromise the ability of students acquiring knowledge and can be a further form of learning in software development. In Education, some tools can confuse students rather than help them to improve their knowledge, so in this experiment we had evidences that SimulES-W improves general software engineering knowledge of students.

According to the work of [14] and [24], there are enough empirical evidence concerning the effectiveness of games-based learning. We think that SimulES-W is a case of that nature. Although, the results shown in this paper confirm our previous experiences [37], further evaluations and experiments have to be performed. In terms of future research directions, this study is still a small-scale investigation but we expect to perform further quantitative/qualitative evaluations with SimuleS-W based on educational levels, and specific subjects to produce further empirical evidence associated with the game. We also hope to further evaluate learning in different roles such as software engineering, technical coordination, project management and quality controlling. Improvements on SimulES-W are underway as to incorporate it in our undergraduate and graduate classes.

References

1. **Connolly, T. M., Boyle, E. A., MacArthur, E., Hainey, T., & Boyle, J. M. (2012)**. A systematic

literature review of empirical evidence on computer games and serious games. *Computers & Education.* Vol. 59, pp. 661–686. DOI: 10.1016/ j.compedu.2012.03.004.

2. **Boehm, B. (2006).** A View of 20th and 21st Century Software Engineering. *Proceedings of the 28th International Conference on Software Engineering*.
3. **Lin, C. Y., Abdel-Hamid, T., & Sherif, J. S. (1997).** Software-Engineering Process Simulation model (SEPS). *Journal of Systems and Software,* Vol. 38, pp. 263–277, DOI: 10.1016/S0164-1212(96) 00156- 2.
4. **Kellner, M. I., Madachy, R. J., & Raffo, D. M. (1999).** Software process simulation modeling: Why? What? How?. *Journal of Systems and Software,* Vol. 46, No. 2–3, pp. 91–105. DOI: 10.1016/S0164-1212(99)00003-5.
5. **Stallinger, F. & Grünbacher, P. (2001).** System dynamics modelling and simulation of collaborative requirements engineering. *Journal of Systems and Software,* Vol. 59, pp. 311–321. DOI: 10.1016/ S0164-1212(01)00071-1.
6. **Suescún–Monsalve, E., Vallejo, P., Mazo, R., & Correa, D. (2007)**. Transparency as a learning strategy to teach Software Engineering. *12th Colombian Conference on Computing.*
7. **Pfahl, D., Klemm, M., & Ruhe, G. (2001).** A CBT module with integrated simulation component for software project management education and training. *Journal of Systems and Software,* Vol. 59, No. 3, pp. 283–298. DOI: 10.1016/S0164-1212 (01)00069-3.
8. **Chen, C. Y. & Chong, P. P. (2011).** Software engineering education: A study on conducting collaborative senior project development. *Journal of Systems and Software,* Vol. 84, No. 3, pp. 479–491. DOI: 10.1016/j.jss.2010.10.042.
9. **Von-Wangenheim, C. G., Savi, R., & Borgatto, A. F. (2013).** SCRUMIA—An educational game for teaching SCRUM in computing courses. *Journal of Systems and Software,* Vol. 86, No. 10, pp. 2675–2687. DOI: 10.1016/j.jss.2013.05.030.
10. **Qin, S. & Mooney, C. (2009).** Using game-oriented projects for teaching and learning software engineering. *20th Annual Conference for the Australasian Association for Engineering Education*, pp. 6–9.
11. **De Oliveira-Barros, M. & De Araujo-Mendes, R. (2008).** Ensinando construção de software aplicada a sistemas de informação do mundo real. *Anais do FEES08-Fórum de Educação em Engenharia de Software.*
12. **Alvarez, C., Alarcon, R., & Nussbaum, M. (2011).** Implementing collaborative learning activities in the classroom supported by one-to-one mobile computing: A design-based process. *Journal of Systems and Software,* Vol. 84, No. 11, pp. 1961–1976. DOI: 10.1016/j.jss.2011.07.011.
13. **Bollin, A., Hochmüller, E., & Mittermeir, R. T. (2011)**. Teaching software project management using simulations. *24th IEEE-CS Conference on Software Engineering Education and Training (CSEE T).*
14. **Ebner, M. & Holzinger, A. (2007).** Successful implementation of user-centered game based learning in higher education: An example from civil engineering. *Computers & Education,* Vol. 49, No. 3, pp. 873–890. DOI: 10.1016/j.compedu.2005. 11.026.
15. **Roubidoux, M. A., Chapman C. M., & Piontek, M. E. (2002)**. Development and Evaluation of an Interactive Web-Based Breast Imaging Game for Medical Students. *Academic Radiology,* Vol. 9, No.10, pp. 1169–1178. DOI: 10.1016/S1076-6332(03)80518-4.
16. **Gomes, A. S., Castro-Filho, A. J., Gitirana, V., Spinillo, A., Alves, M., Melo, M., & Ximenes, J. (2002).** Avaliação de software educativo para o ensino de matemática. *Convergências Tecnológicas--Redesenhando as Fronteiras da Ciência e da Educação: Anais, SBC.*
17. **Cagiltay, N. E. (2007).** Teaching software engineering by means of computer-game development: Challenges and opportunities. *British Journal of Educational Technology,* Vol. 38, pp. 405–415. DOI: 10.1111/j.1467-8535.2007.00705.x.
18. **Claypool, K. & Claypool, M. (2005).** Teaching Software Engineering Through. *Game Design, SIGCSE Bull,* Vol. 37, pp. 123–127.
19. **Drappa, A. & Ludewig, J. (1999).** Quantitative modeling for the interactive simulation of software projects. *Journal of Systems and Software,* Vol. 46, No. 2-3, pp. 113–122. DOI: 10.1016/S0164-1212 (99)00005-9.
20. **Jain, A. & Boehm, B. (2006).** SimVBSE: Developing a Game for Value-Based Software Engineering. *19th Conference on Software Engineering Education Training (CSEET'06),* pp. 19–21. DOI: 10.1109/CSEET.2006.31.
21. **Birkhoelzer, T., Navarro, E., & van der Hoek, A. (2005).** Teaching by Modeling instead of by Models. *Proceedings of the 6th International Workshop on Software Process Simulation and Modeling.*

22. Hainey, T., Connolly, T. M., Stansfield, M., & Boyle, E. A. (2011). Evaluation of a game to teach requirements collection and analysis in software engineering at tertiary education level. *Computers & Education,* Vol. 56, No. 1, pp. 21–35. DOI: 10.1016/j.compedu.2010.09.008.

23. Paraskeva, F., Mysirlaki, S., & Papagianni, A. (2010). Multiplayer Online Games as Educational Tools: Facing New Challenges in Learning. *Comput. Educ.,* Vol. 54, No. 2, pp. 498–505. DOI: 10.1016/j.compedu.2009.09.001.

24. Boyle, E. A., Connolly, T. M., Hainey, T., & Boyle, J. M. (2012). Engagement in digital entertainment games: A systematic review. *Computers in Human Behavior,* Vol. 28, No. 3, pp. 771–780. DOI: 10.1016/j.chb.2011.11.020.

25. Caulfield, C., Xia, J. C., Veal, D., & Maj, S. P. (2011). A systematic survey of games used for software engineering education. *Modern Applied Science,* Vol. 5, pp. 28.

26. Liu, C. C, Cheng, Y. B., & Huang, C. W. (2011). The effect of simulation games on the learning of computational problem solving. *Computers & Education*, Vol. 57, No. 3, pp. 1907–1918. DOI: 10.1016/j.compedu.2011.04.002.

27. Hanaki, N., Ishikawa, R., & Akiyama, E. (2009). Learning games. *Journal of Economic Dynamics and Control,* Vol. 33, pp. 1739–1756.

28. Drappa, A. & Ludewig, J. (2000). Simulation in Software Engineering Training. *Proceedings of the 22nd International Conference on Software Engineering,* pp. 199–208, DOI: 10.1145/337180. 337203.

29. Baker, A., Navarro, E. O., & van der Hoek, A. (2003). Problems and Programmers: an educational software engineering card game. *25th International Conference on Software Engineering,* pp. 614–619.

30. **Figueiredo, E., Lobato, C., Dias, K., Leite, J., & Lucena, C. (2007).** Um Jogo para o Ensino de Engenharia de Software centrado na Perspectiva de Evolução. *Workshop sobre Educação em Computação.*

31. Serrano, M., Serrano, M., Napolitano, F., & Soares, B. (2007), Evolução do SimulES Versão 2.0 Monografia em Ciéncias da Computação, Departamento de Informática. PUC-Rio.

32. Napolitano, F. M. P. (2009). Uma Estratégia Baseada em Simulação para Validação de Modelos em i.

33. Monsalve, E., Werneck, V., & Leite, J. C. S. P. (2010). Evolución de un Juego Educacional de Ingeniería de Software a través de Técnicas de Elicitación de Requisitos. *Proceedings of XIII Workshop on Requirements Engineering (WER'2010).*

34. Serrano, M., Serrano, M., Napolitano, F., & Soares, B. (2007). Evolução do SimulES Versão 2.0. Monografia em Ciências da Computação. *Departamento de Informática.*

35. Do Prado-Leite, J. C. S., Hadad, G. D. S., Doorn, J. H., Kaplan, G. N. (2000). A scenario construction process. *Requirements Engineering,* Vol. 5, No. 1, pp. 38–61. DOI: 10.1007/PL00010342.

36. Yu, E. S. K. (1996). *Modelling Strategic Relationships for Process Reengineering*. University of Toronto, Toronto, Ont., Canada.

37. Monsalve, E. S., Werneck, V. M. B., &. Leite, J. C S. P. (2011). Teaching software engineering with SimulES-W. *Conference on Software Engineering Education and Training (CSEE T).* DOI: 10.1109/ CSEET.2011.5876102.

38. Monsalve, E. S. (2010). Construindo um jogo educacional com modelagem intencional apoiado em princípios de transparencia.

39. Monsalve, E., Werneck, V., & Leite, J. C. S. P. (2010). Evolución de un Juego Educacional de Ingeniería de Software a través de Técnicas de Elicitación de Requisitos. *Proceedings of XIII Workshop on Requirements Engineering (WER'2010).*

40. Oliveira, A. P. A. (2008). Engenharia de Requisitos Intencional: Um Método de Elicitação, Modelagem e Análise de Requisitos.

41. Krasner, G. E. & Pope, S. T. (1988). A Cookbook for Using the Model-view Controller User Interface Paradigm in Smalltalk-80. *J. Object Oriented Program,* pp. 26–49.

42. **Sommerville, I. (2006).** Software Engineering: 8th International Computer Science). *Boston, MA: Addison-Wesley Longman Publishing Co., Inc*

43. Sweedyk, E. & Keller, R. M. (2005). Fun and Games: A New Software Engineering Course. *SIGCSE Bull.,* Vol. 37, No. 3, pp. 138–142. DOI: 10.1145/1067445.1067485.

44. **Pressman, R. (2010).** Software Engineering: A Practitionerś Approach. *7 ed., New York, NY, USA: McGraw-Hill.*

45. Monsalve, E. S., Leite, D. P., & Sampaio, J. C. (2013). *Using for Transparent Pedagogy*. IStar.

46. Leite, J. C. S. D. P. & Franco, A. P. M. (1993). A strategy for conceptual model acquisition. *Proceedings of the IEEE International Symposium on Requirements Engineering.* DOI: 10.1109/ISRE. 1993.324851.

47. Baker, A., Navarro, E. O., & van der Hoek, A. (2005). An experimental card game for teaching software engineering processes. *Journal of Systems and Software,* Vol. 75, No. 1-2, pp. 3–16. DOI: 10.1016/j.jss.2004.02.033.

48. Lin, C. Y., Abdel-Hamid, T., & Sherif, J. S. (1997). Software-Engineering Process Simulation model (SEPS). *Journal of Systems and Software,* Vol. 38, No. 3, pp. 263–277. DOI: 10.1016/S0164-1212(96)00156-2.

49. Serrano, M., Napolitano, F., & Soares, B. (2007). *Evolução do SimulES Versão 2.0*. Monografia em Ciências da Computação, Departamento de Informática, PUC-Rio.

50. Lozano, S., Suescún-Monsalve, E., Vallejo, P., Mazo, R., & Correa, D. (2017). Comparando dos Estrategias de Aprendizaje Activo para Enseñar SCRUM en un Curso Introductorio de Ingeniería de Software.

Article received on 03/05/2017; accepted on 08/11/2017.
Corresponding author is Elizabeth Suescún Monsalve.