Teaching Software Engineering with SimulES-W

Elizabeth Suescún Monsalve[1]
Vera Maria B. Werneck[2]
Julio Cesar Sampaio do Prado Leite[1]
[1] *Pontifícia Universidade Católica PUC-Rio Rio de Janeiro, Brazil*
[2] *Universidade do Estado do Rio de Janeiro UERJ-IME Rio de Janeiro, Brazil*
emonsalve@inf.puc-rio.br
vera@ime.uerj.br
www.inf.puc-rio.br/~julio

Abstract

This work presents an educational board and card game named SimulES-W, as a tool for teaching Software Engineering. It encompasses 5 years of evolution, in which the game Problems and Programmers was the fundamental source. SimulES-W innovates in three distinct areas: it is a web based game, it relies on a broad view of the software process and it is customizable for content. SimulES-W is supported by collaborative software that implements the SimulES board game. The paper describes the game, stresses its strong points, provides initial data on its use and discusses its future.

1. Introduction

Game technology has been used as a teaching tool in many educational areas. Nowadays this technology has started to be applied to software engineering education [1], [2], [3], [6], [14], [15], [17], [18], [19]. Games with an educational purpose are conceived to balance entertainment with dissemination of knowledge, thus motivating students for learning as they play. Software engineering games, usually, allow students to take the role of a project manager and they play by addressing issues related to software engineering in the pursuit of finishing a project.

SimulES [9] is game in which the player performs different roles such as software engineer, technical coordinator, quality controller and project manager. Among the tasks of the game, 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) risk of having poor quality products, (iv) project budget, (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 the roles enumerated above, players simulate a software development team and they are also adversaries of other players. The adversary role is enacted such that each player poses problems or obstacles to the other players, as such changing the game of the other player. This characterizes SimulES as a strategy oriented game, where players have to be aware of competitors by defining a plan to develop with more productivity and quality than the others. The game is organized in different types of rounds, being one where problems and concepts cards are discarded. In this round, each player has the opportunity to choose which other player will receive a "problem card". This assumes that a player has a "problem card" to be used as an obstacle to the other player. Players may also have antidotes cards, called "concept card" which may block the effect of a "problem card". Decision, if an antidote card blocks an obstacle card, is achieved by consensus among players. This discussion, mediated or not by an instructor, is a way of discussing problems and possible resolution schemas according to the cards and, as such, an opportunity for learning. The

978-1-4577-0348-5/11/$26.00 © CSEE&T 2011, Waikiki, Honolulu, HI, USA

player who first produces the required software system, with the quality and budget established by the game objective, wins the game. The objective is set for each game play and is customizable.

This work aims at presenting SimulES-W and how the concepts of software engineering may be taught by this game. SimulES-W is a Web based implementation based on a series of versions that evolved SimulES from its first edition [4]. This paper is organized in six Sections. Section 2 gives an overview of SimulES and describes examples of use, reporting on its use in the classroom. Section 3 emphasizes what was learned in SimulES evolution. Section 4 describes the software SimulES-W, a Web based environment. Section 5 points out future works, and Section 6 presents our conclusion.

2. SimulES

SimulES [4] is based on the Problems and Programmers (PnP) game [1 and 14]. PnP is a card game and the players´ objective is to be the first to complete a common software project. Players use strategies, which are function of by their own preferences and available cards (software engineer cards, concept cards and problems cards). The concept and problem cards are the heart of the game providing problems or solutions to the players. The PnP allows the players to analyze their strategies and the other players' and allows a discussion about the usage of the concept and problem cards. As described in the Introduction, SimulES inherited these characteristics.

The main precondition for playing SimulES is being either a software engineering student or a person involved in software engineering with basic knowledge about the subject. SimulES is a multiplayer game and the player who wins the game is the one who first completes the software product with quality and budget defined in the project card (Figure 2).

SimulES has a main board and an individual board (Figure 1). The main board describes the rounds and moves. The center of the main board displays the project card (Figure 2) surrounded by the set of cards representing problems/concepts and software engineers. Artifacts cards are placed below the project card. Artifacts cards are white or gray and may have in their back a token to mark its quality, see bottom of Figure 1.

The individual board (Figure 1) is the area where the player places his/her software engineers in columns and the artifacts (white and gray cards) in the cells. These artifacts cards represent the software artifacts built by the software engineer, which may contain defects (bugs). The white artifact cards cost twice the price of gray cards, but they have less probability to have bugs. The rows of the individual board represent the different artifact types: requirements, design, code, trace and help.

Different from PnP [1 and 14], SimulES does not have any specific development process and the development process can be explored pedagogically during the game; for instance one player may use an agile approach where other uses a waterfall one. Exploring different development processes may be planned by the instructor as to prepare an after game discussion about development approaches.

The resources used during the game are: the main board, the individual boards (Figure 1), the project cards (Figure 2), the software engineer cards (Figure 3), the problem cards (Figure 4), the concept cards (Figure 5), the white and the gray artifact cards (Figure 1) and the dice.

The project card (Figure 2) that sets the game objective is composed of: name, description, complexity, size, quality, budget and modules. Each module has a minimal composition of artifacts of a given type. A project must have at least one module: the size defines the number of modules. The project complexity is related to the value of white and gray artifact cards, meaning how many points of time a software engineer needs to complete a white artifact by each round of game. The gray artifact is half of this value. For example, if the

project complexity is 2 so the software engineer spends 2 points of time to build a white artifact card; for the gray card, 1 point of time is enough, because the white card is more expensive but has fewer bugs than the gray one. In this example the project has size 2, so the final product has 2 modules. For module 1 the players have to build 2 requirements artifacts (2 artifact cards), 1 design artifact and 1 code artifact. Module 2 has 2 design artifacts, 2 traceability artifacts, 1 help artifact and 1 code artifact. Each artifact is built either with a white or gray card. The quality attribute shows the maximum number of defects allowed in a module. The budget attribute defines the amount of money available to be spent in hiring software engineers.

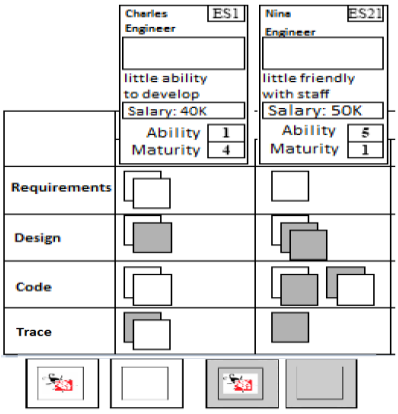


Figure 1. Individual Board.

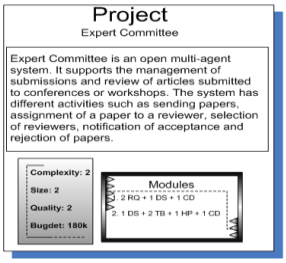


Figure 2. Project Card.

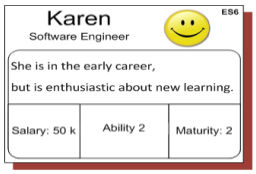


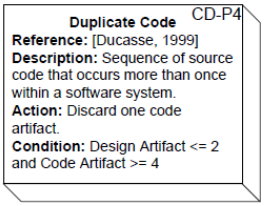
Figure 3. Soft. Eng. Card.

Figure 3 represents a typical software engineer card that has a name, a description of his/hers personality, a salary, which is related to the budget, an ability, which is related to the project complexity, and a maturity, which is used in concept and problem cards conditions. 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 this software engineer. In Figure 2, white artifact cards cost 2 and gray cards cost 1, so Karen (Figure 3) with the ability of 2 points of time and with this project complexity (2) can build 1 white artifact card or 2 gray artifact cards. So, 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).

The problem card (Figure 4) addresses typical problems in software engineering and they should be understood by players. The idea is that the players do reason about these problems and their relation with the conditions stated in the card. A concept card (Figure 5) is a way to neutralize the problem card or to get advantage over the other players. Concept cards have a reference to the literature, a description and usage instructions. Thus, the players can learn more about an issue in which they are interested. Thus, the player can use the concept card either to block a problem card or to improve the software engineer performance. Categorization (upper right corner of concept and problem cards) indicates that cards can be targeted to a specific topic. For example, if the instructor identifies that the topic which should be used in the training is Requirements Engineering Management then he could produce cards to address the topic, and consequently the discussion generated when the players are interacting with the game will be a targeted one.

SimulES has different 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 6 uses an SDSituation Diagram [12] to illustrate these

rounds and their sequence. When the game starts, one project must be chosen from those available ([T1] in Figure 6). All players roll the dice and the one who gets the highest dice result chooses the project and starts the game. Furthermore, the information about the project is displayed in the middle of the main board visible to all players. After that, each player assembles an individual board and picks up one software engineer in the stack of software engineering cards.



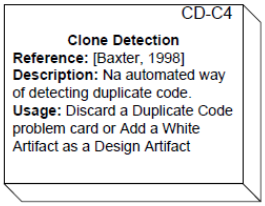


Figure 4. Problem Card. Figure 5. Concept Card.

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 artifact, Inspect artifact, Correct artifact and Integrate Artifacts in a Module (see [T2] in Figure 6). 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 1). 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 2). 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 6, each player rolls the dice once. The dice result allows the player to buy 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 bought. The quantity of software engineer cards will be the difference between three and the dice result (roll of the dice – 3). As an illustration, if Mary rolls the dice and its result was 2, then Mary would buy 2 problem/concept cards. On the other hand, if Mary rolls the dice and its result was 4 then Mary would buy 3 cards (problems and concepts) and 1 software engineer card. 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 2), that is the sum of the salaries of the software engineers posed in the Individual Board. This sum has to be lower than or equal to the project budget. 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 simulations.

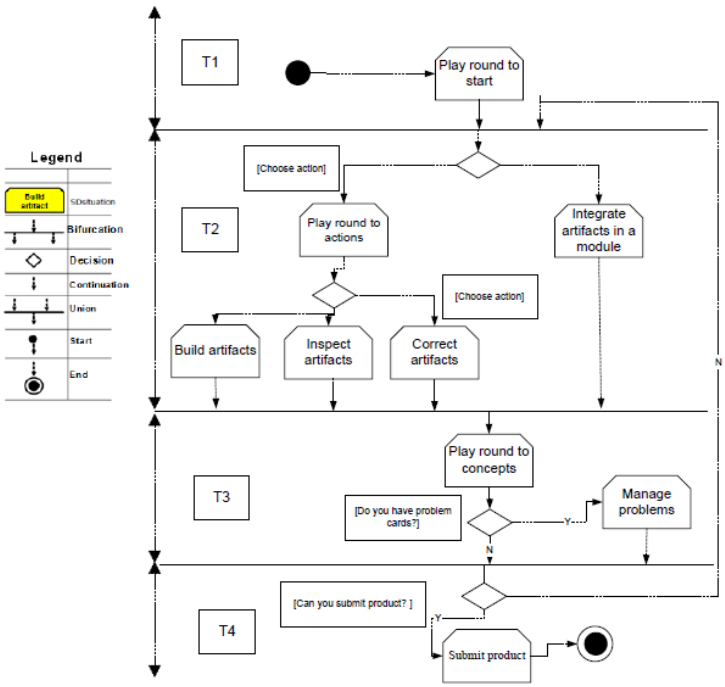


Figure 6. SimuES SDsituations [9].

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), he/she 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/she does not have any card which invalidates the problem cards, this problem will be applied to his/her 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 one module he/she can submit the product. Then the other players have to inspect those artifacts that are not inspected (faced up). The module will be accepted if the number of remaining bugs is lower than or equal to the quality attribute number in the project card (Figure 2). The player who finishes first all the modules within the quality required by the project card wins the game.

SimulES was conceived for teaching software engineering in general. Alternatively, it can be configured to focus on a particular subject of knowledge as well. Since cards can be configured, we can use problem and concept cards tuned to the topic of interest. We can also configure project cards as to only deal with certain artifacts. In the examples of problem and concept cards (Figure 4 and 5) we use cards related to code artifacts (CD). A player can assign the Duplicate Code (CD-P4) card to another player if that player has less than 2 design artifacts and more or equal 4 code artifacts. The argumentation in this case is that this software engineer has built code artifacts duplicating code, so more code artifacts were needed. However, the player who receives this problem can neutralize the card with the

concept card CD – C4, arguing that clone detection alerts about the problem. See that this concept card can also be used by itself saying that the software engineer avoids the problem by thinking ahead, so the player adds a white design artifact to one of his software engineers. This scenario aims at conveying how the educational purpose of the game is implemented. The key to learning is the understanding of why "duplicate code" is a problem. On the other hand, if proper tools are available you could avoid such problem.

SimulES, differently from the Problems and Programmers (PnP) game [1 and 14], does not impose an order towards types of artifacts, as such the player may choose, for instance, to start with design or code artifacts and produce requirements artifacts later. As such the game does not embrace a particular production process. Requirements (RQ), code (CD), help (HP), trace (TR, design (DE) artifacts must exist, but the player decides how to approach it, as mandated by the project profile (Figure 2), which lists the types of artifacts needed to complete the game. It is also important to stress that artifacts must be inspected [7] as well, and that trace artifacts [8] must be produced. In brief, SimulES has the necessary elements to emphasize or generalize knowledge to be transmitted and also may enact particulars of the software process as per concept and problem cards, making SimulES a powerful and useful tool, but also a fun way to teach.

3. Lessons Learned

The interactive process among SimulES, students and instructors has been reported elsewhere [4, 9, 10, 11, 16, and 20]. The improvements built into SimulES-W were based on an experimental case study of SimulES 3.0 in a Software Engineering course in a multidisciplinary graduate program, where the students interacted with each other and were observed by the instructor and an external observer to analyze the situation. Based on a survey with participants [10], we found that: a) the students should receive more training about SimulES before the game; b) contents in the cards needed to be revised because some were difficult to understand by the average students; c) some rules in the game needed to be better explained; d) concepts/problems should be tuned to the group´s background.

The experiment was also important to confirm game acceptance by students. They reported that SimulES is fun and motivates positive competition; as one of the players wrote: "…the game promotes the ability to develop artifacts with a consistent criterion for project requirements, monitoring and testing. It also encourages interaction between players and teamwork, creating healthy competition."

These feedbacks had supplied evidence to make a number of important changes. These changes are reflected on how the game is played now. For example, in [4] there is the first reported improvement of SimulES (version 1.0) that has included and reformulated resources such as the new individual board, and typed problem and concept cards. Version 2.0 [16] was specified using scenarios, the main board was created and the proportion of defects (bugs) were balanced. Version 3.0 [11] shows the first intentional modeling [13] of the game. Finally, version 4.0, SimulES-W, is a digital version of the game, which better supports game customization (tailoring of concept cards/problem cards), allows for non local playing and improves its own modeling [9]. As in similar proposals [6], automation improves play dynamics and control activities are delegated to software. Consequently, players can pay more attention to the important aspects of the game, as the discussions about the use of problem/concept cards.

4. SimulES-W

SimulES-W was developed based on a review of the literature on educational games, as

explained in [9], and SimulES usage feedback. It was identified [9] that educational games did not use any modeling, or the games were modeled without any particular method, or without explanations about the modeling, or models were very superficial. As such, Monsalve [9] paid special attention to modeling using the ideas of intentional modeling [13]. The intentional modeling approach was chosen because it explicitly represents the interaction among players; other representation approaches do not model this interaction, which we believe is beneficial for reasoning about the game even before it is designed. As a result, these models were used in the development of SimulES-W, which was implemented in Java. i* diagrams [13] were mapped to a general MVC (Model View Controller) architecture and the code was instrumented with scenarios describing i* tasks. SimulES-W is played on a browser over the internet, which allows for distributed cooperative playing. This is an advantage related to groupware, according to [5], since individuals cooperate to exchange information, organize and operate together in a shared space. This cooperation is supported by a common interface that allows players to play, make decisions and observe what is happening. Figure 7 presents the development environment and the software components organization.

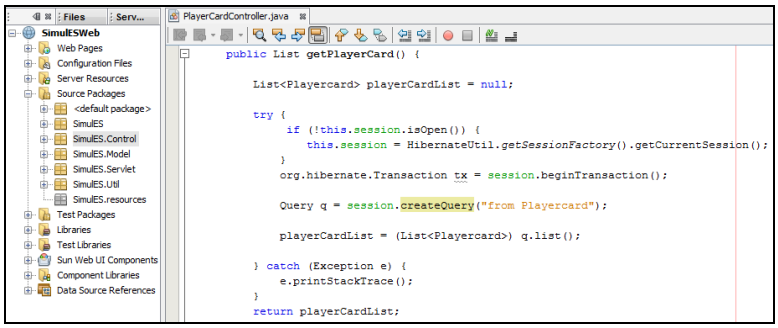


Figure 7. Development Environment of SimulES-W.

Figure 8 exhibits the main page of the game where the messages exchanged between players and system messages are displayed. System messages are displayed when a player makes moves in the game. The project data is also displayed. Figure 9 shows The Individual Board with two situations exhibited (a) Build Artifacts and (b) Inspect Artifacts. The software engineer employed by the player is displayed on the board and each software engineer has a link which displays information about him, for example: salary, ability, maturity and a description like the one presented in Figure 3. The Individual Board (Figure 9 a) also shows the white artifacts cards and the gray artifacts Cards which have not been inspected yet. The bottom of this Figure presents the different operations that can be executed on the board. Part b of Figure 9 shows when some artifacts have already been inspected and also the result of the inspection. The Figure portrays when the artifact has a defect (bug) or not. This result is chosen randomly by SimulES-W and is based on two premises: white artifact cards are less likely to present defects, and conversely, gray artifact cards are more likely to have a defect.

SimulES-W brings the advantage of customization, since cards can 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, as for example changing the rate of white and gray artifact cards, remembering that the game artifact cards (white and gray) have different defect probability. This customization would be part of a stage of game preparation when the instructor has to plan how to use SImulES in his/her class. Alternatively, the concept (white and gray cards) helps students to reason about the fact that, artifacts built with better quality are less prone to defects and that artifacts built with lesser quality is a risky business. Besides, the game stresses the importance of quality control, by means of a built in cycle of inspection [1] that should be applied to all kinds of artifacts. So, it is possible to play

with no inspection, but the game aims at teaching that this strategy involves risk. A player can choose to do inspection during the game or leave this activity to the end for random inspection, following the project card (Figure 2) and other player's choice. Choosing not to do inspection allows the player to quickly build without spending resources with inspection, taking the risk as to finish earlier. A player strategy should be based on an analysis of constraints and possible alternatives. For this, the player must interpret the current environment and analyze his/her situation, as well as the contender´s situation as well. In a realistic context, any person competing in the professional market must imagine future scenarios. Hence the creation of strategies may give players (students) an idea of the possible situations that occur in the future work environment.

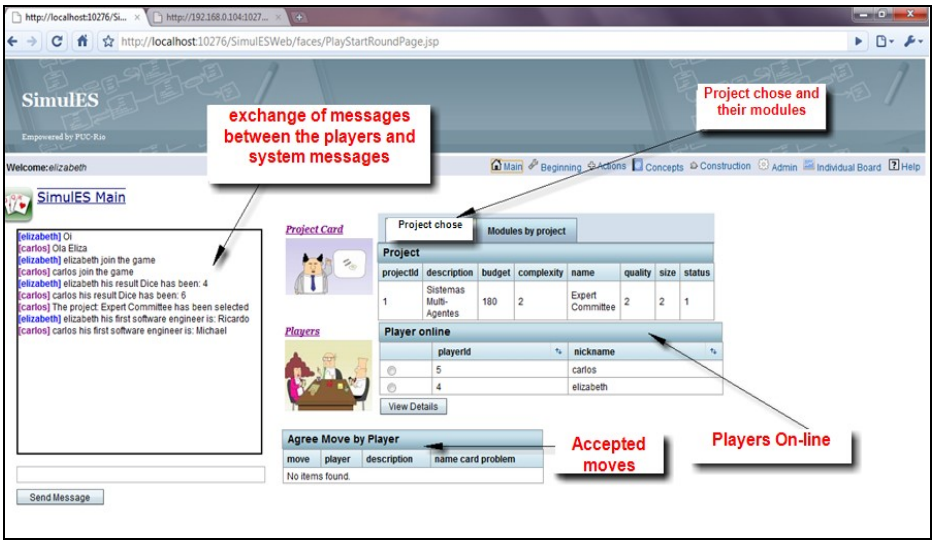


Figure 8. Main Page of SimulES-W.

The control of the game in SimulES-W was not all automated because this is a collaborative approach that aims at training the players in real project situations. The system should be a regulator not a controller of all the players´ moves. For instance, all the players have to pay attention to a move action so they have to know how many artifacts one software engineer has the ability to build. The player´s strategy is another example where he/she chooses a way to achieve the goal of the project. This approach is different from the other descendents of PnP [3, 6 and 18] in which much of the game is played by the software, where SimuES-W is, basically, a collaborative platform.

Another advantage of SimulES-W is that the references for concept cards are links to existing Web based material (Figure 10). Having a direct access to the bibliography or summary of the topic will certainly improve student performance, since references would be easily accessed in the right context. This enables an even more targeted education, with a combination of cards and proper educational material.

5. Future Work

The version of SimulES-W presented in [9] is being packed and will be released as open source software. This version of the game was implemented as open software and allows anyone to evolve it. Furthermore, it works as a Web application and exhibits its interfaces on a Web browser. Moreover, it does not require downloads, installations or configurations. This software is less intrusive than other applications available to download.

We will use SimulES-W as an experience that allows us to provide new ways to explore the reality in software engineering and different strategies to operate on it. In a way, SimulES-W will allow the group (students) to discover new facets of their knowledge about the project by finding alternatives so they will solve the problems and analyze the concepts proposed. In other words, our objective with SimulES-W is that the students interact with software engineering concept in a game that simulates problems that happens in a real project development. It is important to stress that our approach, as by the legacy of PnP, incentives a collaborative view as well as competitive view (problem cards) found on real projects. As part of SimulES-W evaluation process and to make it available to potential users, techniques based on usability IHC (Interaction Human Computer) for Web applications are being applied, focused especially on the analysis of consistency.

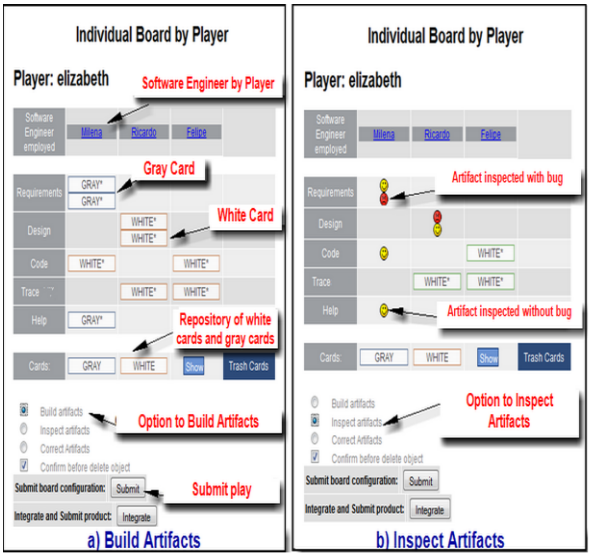


Figure 9. Individual Board of SimulES-W.

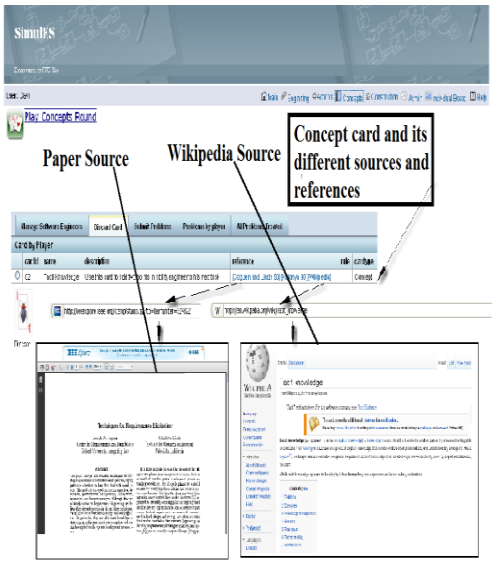


Figure 10. Concept Card of SimulES-W.

As we make the game widely available, we hope that this will impact positively on the quality of feedback. The pedagogical instructions will also be developed as to be an attachment to the distribution package. We are also revising the original material (concept and problem cards), which was last changed in version 3.0 of SimulES. Two sets should be available by the end of 2011: one centered on Requirements Engineering and the other on Software Engineering in general.

6. Conclusion

Educational games are a powerful learning approach for teaching Software Engineering where the students are encouraged to participate, helping educators to simulate real environments, improving students´ skills and stimulating individual and social groups' experiences. SimulES-W is a game that can achieve effective results of disseminating Software Engineering knowledge providing technological resources that can be customized by the instructor. The changes in different versions are the results of improvements identified by positive and negative feedbacks given by people who interacted with the game. SimulES-W has been used in several classes and we plan to conduct more experiences. In these studies we need to focus, not only on the advantages of automation, but mainly on how the lack of a live discussion around a game table will influence the efficacy of the game.

Acknowledgment

Leite acknowledges the partial support of CNPq and Faperj, Cientista do Nosso Estado; Monsalve acknowledges the support of Capes; Leite and Monsalve also acknowledge the support of the CNPq 557.128/2009-9 and FAPERJ E-26/170028/2008 grants for the Brazilian Institute for Web Science Research.

7. References

[1] A. Baker, E. O. Navarro, and A van der Hoek, "Problems and Programmers: an educational software engineering card game".In Proceedings 25th International Conference on Software Engineering, IEEE Computer Society Press, 2003, pp 614-619.

[2] J. Beatty and M. Alexander, "Games-Based Requirements Engineering Training: An Initial Experience Report", International Requirements Engineering, RE '08. 16th IEEE, Catalunya, Spain. (Sept. 2008). Pp. 211-216.

[3] T. Birkhoelzer, E. Navarro and A. Van Der Hoek, "Teaching by Modeling instead of by Models", Proceedings 6th International Workshop on Software Process Simulation and Modeling, St. Louis, MO, 4, 2005.

[4] E. M. L. Figueiredo, C. A. Lobato, K. L. Dias, J. C. S. P. Leite, C. J. P. Lucena, "Um Jogo para o Ensino de Engenharia de Software Centrado na Perspectiva de Evolução", Workshop sobre Educação em Computação (WEI – 2007), pp. 37-46.

[5] H. Fuks, A. B. Raposo and M. A. Gerosa, "Do Modelo de Colaboração 3C à Engenharia de Groupware", Simpósio Brasileiro de Sistemas Multimídia e Web – Webmidia 2003, Trilha especial de Trabalho Cooperativo Assistido por Computador, November. 2003, Salvador-BA.

[6] A. Jain and B. Boehm, "SimVBSE: Developing a Game for Value-Based Software Engineering". Proceedings 19th Conference on Software Engineering Education and Training, 2006, pp. 103 -114.

[7] J.C.S.P. Leite, J. Doorn, G. Hadad and G. Kaplan, "Scenario Inspections", Requirements Engineering Journal: Volume 10, Number 1, (January.2005). Springer-Verlag London.

[8] J.C.S.P Leite, G. Rossi, V. Maiorana, F. Balaguer, G. Kaplan, G. Hadad A. and Oliveros, "Enhancing a Requirements Baseline with Scenarios", Proceedings of the Third International Symposium on Requirements Engineering: IEEE Computer Society Press, 1997, pp. 44-53.

[9] E. Monsalve. "Construindo um Jogo Educacional com Modelagem Intencional Apoiado em Princípios de Transparência", (May. 2010), Dissertação de Mestrado (Masters Thesis), Departamento de Informática, PUC–Rio.

[10] E. Monsalve, V. Werneck, and J.C.S.P. Leite, "Evolución de un Juego Educacional de Ingeniería de Software a través de Técnicas de Elicitación de Requisitos", 13th Workshop on Requirements Engineering – WER 10, (April 12-13, 2010 – Cuenca, Ecuador), pp. 63-74.

[11] F. Napolitano, "Uma Estratégia Baseada em Simulação para Validação de Modelos em i*", Dissertação de Mestrado(Masters Thesis), (March.2009), Departamento de Informática, PUC–Rio.

[12] A. Padua Oliveira, L. Cysneiros, "Defining Strategic Dependency Situations in Requirements Elicitation", 9th Workshop on Requirements Engineering. WER 2006, (15 December 2006), pp.12-23.

[13] A. Padua Oliveira, "Engenharia Intencional: Um Método de Elicitação, Modelagem e Análise de Requisitos", Tese de Doutorado (PhD thesis), 2009, Departamento de Informática, PUC-Rio.

[14] Problems and Programmers. (April.2009) at http://www.problemsandprogrammers.com/.

[15] G. Regev, D. Gause and A. Wegmann, "Requirements Engineering Education in the 21st Century, an Experiential Learning Approach", The 16th International Requirements Engineering Conference (RE'08), 2008, pp. 85-94.

[16] M. Serrano, M. Serrano, F. Napolitano, B. Soares, "Evolução do SimulES Versão 2.0". Monografia em Ciências da Computação, 2007, Departamento de Informática, PUC–Rio.

[17] R. Smith, and O. Gotel, "Using a Game to Introduce Lightweight Requirements Engineering", in Proceedings of the 15th IEEE International Requirements Engineering Conference, 2007, pp. 379-380.

[18] R. Smith, O. Gotel, "Gameplay to Introduce and Reinforce Requirements Engineering Practices", in Proceedings of the 16th IEEE International Requirements Engineering Conference, 2008, pp. 95-104.

[19] Software Engineering Simulation by Animated Models (SESAM) – Stuttgart–Germany. (April.2010) <http://www.iste.uni-stuttgart.de/se/research/sesam/overview/index_e.html>.

[20] E. Monsalve, V. Werneck, and J.C.S.P. Leite, "SimulES-W: Um Jogo para o Ensino de Engenharia de Software", Fórum de Educação em Engenharia de Software – FEES 10, in SBES -10 (Brazilian Symposium on Software Engineering), 2010 – Bahia, Brasil.