SPIAL: A Tool for Software Process Improvement Training

Daniela C. C. Kupsch (author), Rodolfo S. F. Resende (advisor)

Departamento de Ciência da Computação – Universidade Federal de Minas Gerais (UFMG) – Belo Horizonte – MG –Brasil

{cascini,rodolfo}@dcc.ufmg.br

Abstract. *The complexity of software development is increasing and at the same time, the markets require reduced costs and short time production. The Software Engineering (SE) disciplines provide an introductory overview of the area and do not sufficiently address the practical issues of software development organizations. With the aim of providing a more real experience of the software development processes, we created SPIAL, a simulation game that addresses the SE best practices. SPIAL presents a metaphor of a software development organization that is running a process improvement initiative.*

Resumo. *Os requisitos para desenvolvimento de software estão mais complexos e as soluções devem ser obtidas de forma rápida e barata. As disciplinas de Engenharia de Software (ES) que tratam desse conhecimento proveem uma visão introdutória da área e não abordam suficientemente as práticas das organizações desenvolvedoras de software. Com o objetivo de prover uma experiência mais real dos processos de desenvolvimento de software, nós criamos SPIAL, um jogo de simulação que aborda as melhores práticas de ES. SPIAL apresenta a metáfora de uma organização desenvolvedora de software que está executando uma iniciativa de melhoria de processos.*

1. Problem Description and Motivation

One of the biggest challenges faced by society and, in particular, by universities is the need to make the process of people training more effective and efficient. The expectation of a higher quality in education and training has not been satisfied in general and there is an increasing dissatisfaction of the industry related to the students' preparation in various economical sectors. In addition, it is difficult to practice the concepts presented in the classroom by means of the traditional teaching methodology, centered on the professor [Chen et al. 2008]. One way to improve this scenario is applying methods and technologies to support teaching, for example, activities developed in real projects, execution of residence programs, games and simulations.

Considerable changes in the educational process and in the universities are expected upon the use of new pedagogic and educational technologies. Face to face courses, centered on professors, which are commonly offered by worldwide universities, are being substituted by more dynamic courses, which offer a hybrid use of innovative technologies. Important examples include the various free of charge online courses (*MOOC – Massive Open Online Course*) which are offered by renowned universities.

Other examples include the use of simulation games as a complementary method for teaching some topics addressed in undergraduate courses.

Simulation is a powerful tool which is commonly used to teach processes and techniques impossible or too expensive to be trained in the real world. Its use emerged from the evaluation of the new teaching technologies' potential. Using simulation it is possible to increase students' interest and the results are often surprising. Presenting important projects' related issues within a simulator allows the application of concepts learned in the classrooms in a more realistic environment. Some researches show that game-based simulators can be used as a teaching support tool [Navarro 2006]. These tools deal with activities in large scale projects in such a way that a university within its small scale projects would not be able to.

In the Software Engineering field, we observed that it is very difficult to train students in the real situation of a software development organization, due to the very nature of software applications and the great diversity of organizational cultures. A number of simulation games were developed for enhancing learning and understanding of Software Engineering [Peixoto et al. 2011b]. The variety of simulation games suggests that this is an attractive research field with great expansion possibilities.

After analyzing the defects found in artifacts produced by students in a team-based Software Engineering project and interviewing instructors, we observed that there is an undesirable gap between what is taught in the classroom and what the students have to do in the team project. Considering these aspects, we designed SPIAL (*Software Process Improvement Animated Learning Environment*) [1], a graphical, interactive, customizable, simulation game. SPIAL allows students to practice some Software Engineering skills in an environment that resemble more closely those in industry. Specifically, students are able to:

- Practice concepts related to technical and managerial processes of software development.
- Reinforce the concepts learned in class, mainly, the Software Engineering lessons, carrying out a Software Process Improvement (SPI) initiative (considering, for example, that requirements deficiencies are the prime source of project failures [Glass 1998]). Specifically, the simulation rewards correct decisions (appropriate Software Engineering practices) and penalize unjustifiable ones.
- Follow the stages of a software development process in an organization.
- Learn some of the events that can occur during a Software Process Improvement project (e.g. resistance to change, high-level management commitment).
- Observe possible results of improvement actions taken during development (e.g. the reduction of defects after performing the Requirement Engineering activities).
- Analyze the effects of following an immature software development process (e.g. more restricted process visibility, higher number of defects). Therefore, reinforcing the possible advantages of having a defined process.

[1] The game can be downloaded from www.dcc.ufmg.br/~cascini

SPIAL is a single-player game in which players take on the role of a manager of an SPI group in a software development organization. SPIAL's scope covers both a development project and an improvement project. The player is given a process improvement task and he or she can interact with other stakeholders (high level management, project manager, team member, consultant, or customer) represented as non-player characters, i.e. a character controlled by the computer (see Figure 1a). In order to complete the task, the player can make investments for improving specific process areas of a software development project. A good investment strategy will result in improvement of process areas and a bigger budget for further investments. The player can visualize project estimations, indications of process areas capability level and decide which process area to invest. During the development project, the player can visualize the effects of his or her selections on the outcomes (productivity, defect, cost, and time-to-market measures) and if needed, change his or her investments (see Figure 1b). The final outcome is a score that represents how close the results are to the initial proposed target. During the game, the non-player characters communicate the effects of the player's actions through bubbles over their heads.

We evaluated SPIAL through a pilot experiment and an inspection using the Semiotic Inspection Method [Peixoto et al. 2010d]. The educational aspects addressed in the experiment included the capability of students to understand, remember and apply Software Engineering concepts in the context of a CMMI software process improvement initiative. This thesis advanced the knowledge of developing and applying educational games in a Software Engineering context.

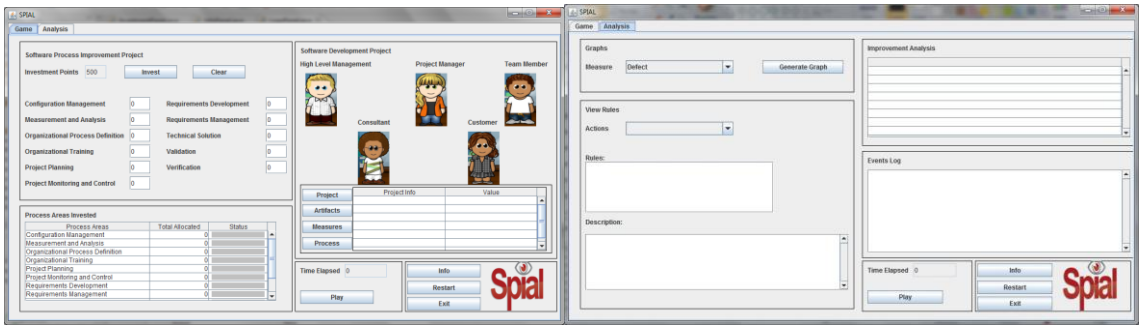

Figure 1. SPIAL Graphical User Interface: (a) Main tab sheet; (b) Analysis tab sheet.

2. Contribution

The main contributions of this research lie in the development and validation process that underpinned this simulation game creation, as well as the design and application of a simulation game in Software Engineering education. This process, which has a strong focus on interaction aspects, can provide some insights for simulation game developers.

The main achievements resulting from SPIAL development can be summarized as follows:

a. Theoretical aspects:

- Description of the main characteristics that make the adoption of simulation games successful. This information was consolidated in Peixoto et al. [2012a].
- Specification of an SPI simulation game with: (i) definition of roles, activities, and results; (ii) definition of a simulation model; (iii) definition of a reusable

framework; (iv) identification and characterization of the actual results of SPI initiatives; and (v) identification of the issues and challenges during the design of SPIAL [Peixoto et al. 2012b].

b. Practical aspects:

- Development of a set of components which can be reused in the development of new Software Engineering simulation games [Peixoto et al. 2012a].
- Integration of a specific SPI simulation model with the reusable components.
- Development of the simulation game itself, along with its SPI simulation model.

c. Empirical aspects:

- Communicability evaluation of SPIAL conducted by a specialist.
- Practical evaluation of SPIAL carried out by students through a pilot experiment.

In sum, the innovation introduced consists of an SPI simulation game that addresses several aspects of Software Engineering education, by allowing students to practice part of the Software Engineering skills.

3. Results

The main achievement of this research lies in the design, application and validation of SPIAL, an SPI simulation game. The novelty of this work is twofold. Firstly, it presents an SPI simulator, along with its simulation model and reusable framework. Secondly, it depicts the steps needed to create the simulation game, which includes the analysis of Human-Computer Interactions issues and SPI literature.

This research provided results on theoretical, practical and empirical level. The specific achievements can be summarized as follows:

Identification of students' mistakes: We investigated the common problems incurred by students in a Software Engineering course [Peixoto et al. 2010a]. The most striking observations are the difficulty that students have to bridge the gap between the theoretical lectures and the team project, the limited range of skills that they apply during the project development, and, usually, they do not follow the prescribed software development process during the team project development. They first elaborate the technical artifacts and then they complete the baseline with the managerial artifacts. Even if they are required, students usually do not inspect their artifacts. Unfortunately, we were even able to learn that they filled the defect report artifact with unreal data. This could be observed because of inconsistencies among artifacts.

Analysis of other simulation games: We identified and compared eight Software Engineering simulation games [Peixoto et al. 2011b]. Based on this analysis, we defined the essential design aspects for the development of our simulator. The assessment of these simulation games revealed that, although they are, in some cases, developed with different aims, they have a considerable number of common aspects, for example, "trial and error" strategy, goals, rules and some game features. They also have different features mainly regarding when the feedback is available, the type of feedback, and the adaptability characteristics.

Investigation of the SPI and Software Engineering domains: We investigated the domain of our simulation game. Since a simulation game should reflect what happens in the real world, we analyzed the main results reported by organizations in research papers

regarding their SPI initiatives. Through a systematic literature review, we gained an up-to-date view of the SPI area, allowing us to identify and characterize the actual results of SPI initiatives. In the SPIAL context, the results of this study provided the basis for the simulation model and requirements definition. We assessed 91 studies related to the results of SPI efforts [Peixoto et al. 2012c]. Most of the data came from large companies which are mainly focused on improving their project processes, requirement engineering processes, software review processes, and establishing organizational level processes. This encompasses mainly process areas of maturity levels 2 and 3 of CMMI. Considering all types of organizations, CMM/CMMI is the most frequently used improvement models. We observed that SPI initiatives do not bring "dramatic" changes (the works report improvements between 5 and 35%), and the main common reasons for launching SPI efforts are the reduction of the defect rates detected during development and after delivery, the improvement of productivity, the reduction of time-to-market, i.e. the time needed to deliver the product, and costs. A considerable percentage of data originated from interviews, observations or questionnaires revealing that most of the organizations have difficulties to measure and analyze their improvement quantitatively. Additionally, the studies have limitations in terms of rigor, credibility and validity in their findings. They do not provide enough information about the software development organization context and the correlation between the process improvement and the effect in the organization environment, preventing us of producing a better assessment.

In addition, we collected 123 Software Engineering rules from text books and other Software Engineering simulation games. This set includes rules that are dependent on values that vary from one situation to another and rules beyond the Software Engineering area, including a wider range of business processes. From this set, we selected the ones related to the SPIAL process areas, resulting in 57 Software Engineering rules. These rules are used to teach the best practices of Software Engineering to students, rewarding or penalizing their actions.

Development of SPIAL: Based on the steps previously carried out, we identified important requirements for our simulation game, which were used in the definition of the SPIAL behavior. We also evaluated the available components for simulation games development. Since we did not find components that were generic enough to be reused, we decided to develop our own framework, named FASENG [Peixoto et al. 2012a]. FASENG is a framework for development of Software Engineering simulation games. It was developed based on a set of selected requirements related to the application of learning theories.

Evaluation of SPIAL: Finally, we evaluated SPIAL from two viewpoints: specialist and player. An inspection using the Semiotic Inspection Method was conducted by a specialist and communicability breakdowns were identified. We evaluated the game carrying out a pilot experiment with undergraduate students of a Computer Science Software Engineering course as players. On average, students found the game quite enjoyable and they had fun during the game play. They agreed about adopting this game in a Software Engineering course as a complementary approach.

Refereed Conference Publications

Peixoto, Daniela C. C., Possa, Rodrigo M., Resende, Rodolfo S., Pádua, Clarindo Isaías P. S. (2012a) FASENG: A Framework for Development of Software Engineering Simulation

Games. In: 42nd Frontiers in Education Conference (FIE'12), Seattle, Washington, USA, October 3-6.

Peixoto, Daniela C. C., Possa, Rodrigo M., Resende, Rodolfo S., Pádua, Clarindo Isaías P. S. (2012b) Challenges and Issues in the Development of a Software Engineering Simulation Game. In: 42nd Frontiers in Education Conference (FIE'12), Seattle, Washington, USA, October 3-6.

Peixoto, Daniela C. C. (2012c) SPIAL: A Tool for Software Process Improvement Training. PhD Thesis, Universidade Federal de Minas Gerais, Brasil, setembro.

Meirelles, Lucas; Peixoto, Daniela; Monsalve, Elizabeth; Figueiredo, Eduardo; Werneck, Vera; Leite, Julio, Resende, Rodolfo; Pádua, Clarindo (2011a). Uso de Jogos para o Ensino de Engenharia de Software. In: IV Fórum de Educação em Engenharia de Software, XXV Simpósio Brasileiro de Engenharia de Software, São Paulo-Brasil, 26 a 30 setembro.

Peixoto, Daniela C. C., Possa, Rodrigo M., Resende, Rodolfo S., Pádua, Clarindo Isaías P. S. (2011b) An Overview of the Main Design Characteristics of Simulation Games in Software Engineering Education. In: 24th IEEE-CS Conference on Software Engineering Education and Training, Waikiki, Honolulu, Hawaii, May 22-24, pp. 101-110.

Peixoto, Daniela C. C., Batista, Vitor A., Resende, Rodolfo S., Pádua, Clarindo Isaías P. S. (2010a) Learning from Students' Mistakes in Software Engineering Courses In: Frontiers in Education, (FIE'10), October 27-30,Virginia, USA, IEEE.

Peixoto, Daniela C. C., Batista, Vitor A., Resende, Rodolfo S., Pádua, Clarindo Isaías P. S. (2010b) How to Welcome Software Process Improvement and avoid Resistance to Change In: International Conference on Software Process, (ICSSP'10), July 8-9, Paderborn, Germany, pp. 138-149.

Peixoto, Daniela C. C., Batista, Vitor A., Resende, Rodolfo S., Pádua, Clarindo Isaías P. S. (2010c) A Case Study of Software Process Improvement Implementation. In: 22nd International Conference on Software Engineering and Knowledge Engineering, (SEKE'10), July 1-3, Redwood City, San Francisco Bay, USA, pp. 716-721.

Peixoto, Daniela C. C., Prates, Raquel O., Resende, Rodolfo S. F. (2010d) Semiotic Inspection Method in the Context of Educational Simulation Games. In: Proceeding of the 25th Annual ACM Symposium on Applied Computing,(SAC'10) Sierre, Switzerland, p. 1207-1212.

Peixoto, Daniela C. C., Batista, V. A., Rocha, G. M., Pádua, C. I. P. S., Resende, R. S. F. (2008) Uma Experiência de Melhoria de Processo utilizando a Análise Causal de Defeitos. In: Simpósio Brasileiro de Qualidade de Software, Florianópolis. VII Simpósio Brasileiro de Qualidade de Software.

Peixoto, Daniela C. C., Pádua, C. I. P. S., Veloso, E. A., Resende, R. S. F. (2007) Prevenção de defeitos em Requisitos de Software: Uma caracterização do processo de melhoria In: VIII Simpósio Internacional de Melhoria de Processo de Software, São Paulo.

References

Chen, W., Wu, W., Wang, T. and Su, C. (2008) A Game-based Learning System for Software Engineering Education. In Proceedings of the 38th ASEE/IEEE Frontiers in Education Conference. p. T2A-12-T2A-13. Saratoga Springs: New York.

Navarro, E. O. (2006) SimSE: A Software Engineering Simulation Environment for Software Process Education. PhD thesis, Donald Bren School of Information and Computer Sciences, University of California, Irvine.

Glass, R. L. (1998). Software Runaways: Lessons Learned from Massive Software Project Failures. NL: Prentice Hall.