USING SIMULATION TRAINING GAMES TO CREATE MORE ACTIVE AND STUDENT CENTERED LEARNING ENVIRONMENTS FOR SOFTWARE AND SYSTEMS ENGINEERING EDUCATION

Tucker Smith, Aaron Tull
The University of Texas at Dallas, 800 West Campbell Road, Richardson, Texas, U.S.A.

Kendra Cooper, Shaun Longstreet
The University of Texas at Dallas, 800 West Campbell Road, Richardson, Texas, U.S.A.

Keywords: Education, Games, Simulation training, Software and systems engineering.

Abstract: This paper contends that many of the bottlenecks and difficulties facing faculty and students in traditional lecture and textbook approaches to classrooms can be effectively addressed through the creation and application of simulation based games. In addition to augmenting an active, student-centered learning environment, simulation games can also cultivate soft skills such as communication, teamwork and self-reflection. From a software engineering perspective, an AOP-based architecture approach to developing a simulation game allows for greater flexibility and an increased ability to tailor the simulation for particular institutional and pedagogical needs.

1 INTRODUCTION

Software engineering (SE) and Systems Engineering (Sys) education have critical roles in preparing the future CISE workforce for careers in an increasingly technical, interconnected, and changing world. The specialized knowledge required to prepare students for a career in SE and Sys is rapidly changing with the advent of new techniques and technologies; their educational infrastructure faces significant challenges including the need to rapidly, widely, and cost effectively introduce new or revised course material; encourage the broad participation of students; address changing student motivations and attitudes; support undergraduate, graduate and lifelong learning; and incorporate the skills needed by industry. E-learning, as part of this infrastructure, has the potential to address many of these challenges and have a significant impact. In SE/Sys, current e-learning options include non-interactive slide-based or video/webinar courses and, very recently, a small number of research education games.

There is a growing use sophisticated simulation games in higher education. For example, in the field of supply chain management, there has been growth in simulation games (Horn and Cleaves, 1980; Riis 1995; Chen and Samroengraja, 2000; Anderson and Morrice, 2000; and Mustafee and Katsialiki). With respect to software engineering, two groups have initiated games to teach SE concepts: SimSE (Wang 2004) and SESAM, Software Engineering Simulation by Animated Models (Ludewig, 1992).

We believe the following SE curriculum guidelines (Diaz Herrera 2004) could be better met using a simulation game approach, rather than traditional, lecture based approach:

- Software engineering must be taught as a problem-solving discipline.
- The curriculum should have a significant real-world basis.
- To ensure that students embrace key ideas, care must be taken to motivate students by using interesting, concrete and convincing examples.
- Software engineering education in the 21st century needs to move beyond the lecture format: It is therefore important to encourage consideration of a variety of teaching and learning approaches.

Smith T., Tull A., Cooper K. and Longstreet S..
USING SIMULATION TRAINING GAMES TO CREATE MORE ACTIVE AND STUDENT CENTERED LEARNING ENVIRONMENTS FOR SOFTWARE AND SYSTEMS ENGINEERING EDUCATION.
DOI: 10.5220/0003621603860392
In *Proceedings of 1st International Conference on Simulation and Modeling Methodologies, Technologies and Applications* (SIMULTECH-2011), pages 386-392
ISBN: 978-989-8425-78-2

Games have been recognized as a tool to promote deep, reflective learning (Gee, 2003). An extensible, web-enabled, freely available, engaging, problem-based game platform that provides students with an interactive simulated experience closely resembling the activities performed in a (real) industry development project would transform the SE/Sys education infrastructure. Pedagogical improvements include more active, student centred learning that stresses higher levels of critical thought and real-world applications.

Literature is available that reports the issues with more traditional (lecture based) pedagogical methodology and the benefits of simulation game learning. Therefore, the need for SE/Sys simulation games has been established; however there is a limited body of work available on games to fill this need (refer to Section 5). Our research addresses a key concern that has received little attention to date: how to systematically integrate "good" game design concepts to make the game fun and engaging *and* embody a collection of well defined SE/Sys learning objectives in the game play. We believe games that accomplish both of these requirements are necessary to successfully shift the educational paradigm towards a more effective pedagogy.

This short paper is organized as follows. We review the issues of traditional (lecture based) courses and the benefits of simulation games based learning reported in the literature in Section 2, which reiterates the need for SE/Sys simulation games. In Section 3, two collections of requirements for the SE/Sys games are described: "good" game design and the learning objectives. An overview of our proposed framework, SimSYS, is described in Section 4. Related work on SE/Sys game literature is presented in Section 5; conclusions and future work are in Section 6.

2 TRADITIONAL VS. SIMULATION GAME BASED SE/SYS EDUCATION

Here we review the reported issues with the traditional (lecture based) pedagogy and the reported benefits of more interactive approaches, including the use of simulation games.

2.1 Traditional Pedagogy

The majority of introductory courses on software engineering are textbook and lecture based. Textbooks are static, representational pieces of SE knowledge – they do not actively engage student learning, let alone instil life-long learning skills. Traditional approaches are almost entirely dependent on physical classrooms and synchronous meeting time with faculty present; this limits the opportunities that students have for sustained practice and immediate, useful feedback. Moreover, in traditional teaching methods, faculty and students are often outside of a feedback loop – that is, faculty do not know if students understand key concepts until after summative assessments (such as midterms or final projects), and students are not aware of their mastery of course content until after an assessment occurs and long after said content was first presented to them. Finally, in the text and lecture style of teaching, one of the biggest issues that faculty face, even with successful students, is the limited ability that these higher-performing students have with taking their knowledge from text and/or lecture materials and later applying it to real-world software management scenarios. Text and lecture curricula depend on plugging in numbers or regurgitating formulaic examples; they provide few opportunities to learn transference skills, where variables in problems strengthen student abilities to apply learning to similar but different situations.

2.2 Simulation Game based Pedagogy

The simulation game approach allows students to create a personalized learning experience, progressively incorporating new knowledge and scaffolding it into what they already know (Alvermann, 1985, Weinstein, 2000). The variability within this interactive environment permits students to work on lower-level tasks repeatedly as they begin to develop broader analytical skills and make progress towards completing the game objectives. At the same time, each student can engage course-based material at his or her own pace; he or she is able to explore a variety of actions to progress towards completing the game. Feedback is frequent and immediate, thereby reinforcing mastery of fundamental skills required for advancing further into the game.

Because a simulation game is task-oriented, it encourages practice and mastery rather than rote memorization. It encourages students to use higher orders of thinking because not only do student players need to track fundamental concepts and resources, they must also weigh appropriate applications (Walker, 2008, Westera, 2008, Nadolski, 2010). A game encourages strategic learning where students practice transference skills to more complex scenarios that, while related in practice, are dissimilar in appearance to examples presented in class (Alexander, 1998, Blessing, 1996).

Creating a playable simulation for use within the SE curriculum establishes a more motivating learning environment since the game appeals to a student's sense of fantasy and amusement; it is also self-directed, appealing to a student's curiosity; and it is a continuous challenge where existing tasks or knowledge appear incomplete, inconsistent or incorrect, thereby pushing a student to continue and foster deeper levels of learning (Malone, 1980).

3 SE/SYS SIMULATION GAME PLAY REQUIREMENTS

3.1 What makes a "Good" Game?

Game researchers have established many characteristics that make a "good" game. We categorize established characteristics of "Good" Game Design around two key QoS attributes: *Usability and Playability.*

- **Interactive Embedded Tutorials**. To help the player learn how to play the game, this approach provides a fast and intuitive means. It removes the burden of the player working through a separate tutorial or user manual. (*primarily Usability*)
- **Multiple Real-time Strategy Scenarios**. To keep the game fresh and interesting to players over time, the game needs to present different "twists" each time. The graphic interface and underlying engine need to support the generation of scenarios and the player's interactions in real-time. (*primarily Playability*)
- **Self-assessing Framework**. To provide immediate feedback to the player regarding their performance, or success, the game needs to keep track of this and present it to the player. (*Usability and Playability*)
- **Graphical Sophistication.** To appeal to a broader audience, the game needs to have a high-end graphical interface. (*primarily Playability*)
- **Multiple Levels of Difficulty**. The game should be multi-level in order to continue to challenge the player and retain their interest over time. A player will become bored if the game only has one level of difficulty. (*primarily Playability*)
- **Multiple Players**. The game should be multi-player to allow players to compete against one another, rather than against a program. Recently, for example, new popular games are being launched using the **Google-game**-platform. (*primarily Playability*)

For educational games, we propose adding the following three characteristics:

- **External-assessing Framework.** To anonymously measure the learning outcomes achieved by the player and record progress in the game.
- **Clearly Identifying In-game Objectives with Course Learning Outcomes.** To make the learning explicit and cultivate better self-awareness within the student players.
- **Curriculum Guide for Faculty.** Because the game will be designed to be used at other campuses, a 'game-master' guide will be developed. This will be a template that other faculty can use to consider how to best utilize the game for their curricula.

3.2 What are we Teaching?

We envision a Game Development Platform (GDP) that supports developing a collection of games, each with their own learning objectives. The Software Engineering Education Knowledge (SEEK) standard of SE2004 will be used as one part of the foundation in our research: we use it to identify, prioritize, and select game scenarios. Another source of learning objectives is our industrial advisory board members. Representatives from companies with local offices are involved with research, senior design projects, and provide feedback on curriculum content. Our position is that learning objectives can be considered as high level requirements that can be systematically elicited, specified, analysed, and managed using established best practices in requirements engineering.

For prioritization, SEEK assigns to each topic one of three Bloom taxonomy levels: **knowledge** (Remembering previously learned material) **comprehension** (Understanding information and the meaning of material presented) and **application** (Ability to use learned material in new and concrete situations). Furthermore, the topics are categorized as Essential, Desirable, or Optional. SE2004 categorizes application level topics as Essential. For the learning objectives traced back to specific SEEK topics, the prioritization of adopting the topics is that Essential, Application level topics have the highest priority.

4 SimSYS GDP

In order to achieve the highest impact with game based pedagogy, we contend that the simulation has a strong core with flexibility for tailoring to specific institutional needs in mind. The architecture for the SimSYS GDP we propose is illustrated at a high level in Figure 1. It is intended to support the development of collections of simulation training games and their execution, where each game embodies a specific set of learning objectives. As a game is played, the game play events are logged; they are analyzed to automatically assess a player's accomplishments and automatically adapt the game play script.

4.1 Architecture

Each component is briefly described below in terms of its purpose and capabilities it provides.

Game Play Integrated Development Environment (IDE). The IDE provides a What You See and Hear is What You Get (WYSHIWYG) UI that abstracts the XML specification of a game and a text editor to modify the XML directly. One concern with XML game scripts is their potential size and complexity; game designers need to be able to modularize the game specification. Designers can structure their game play specifications into scenes. They can work on one scene at a time, considering the characters, dialog, graphics, sound, and possible game play alternatives. The IDE allows the game designer to execute the game script from within the IDE for convenience. The Game Play scripts are stored in the Game Play Data Repository; they can be saved, loaded, copied, or deleted.

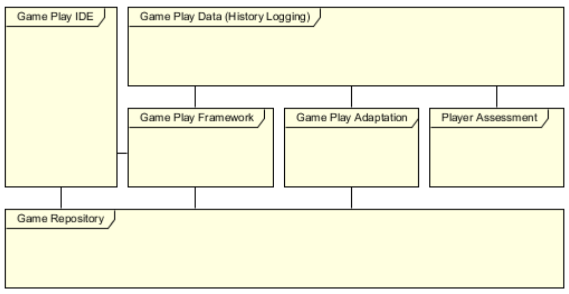

Figure 1: Overview of the Game Development Platform.

The Agent-oriented Paradigm (AOP) is an alternative approach for constructing software systems that is well-suited for modelling human interaction such as collaboration, negotiation, and conflicts. AOP is based on the concept of an agent, which are software entities that are situated, autonomous, flexible, and social (Wooldridge, 2009). Agents sense the environment and perform actions that change the environment. They have control over their own actions and internal states; they can act without direct intervention from humans. Agents are responsive to changes in the environment, goal-oriented, opportunistic, and take initiatives. They interact with other agents (software, human) to complete their tasks. The agent-oriented approach is beneficial in systems that (per O'Malley, 2001): require complex/diverse types of communication; have behaviour that is not practical/possible to specify on a case-by case basis; involve negotiation, cooperation and competition among different entities; must act autonomously; and is expected to expand or change. Recently, the agent-oriented paradigm has been applied to games. Agent-oriented design solutions have been proposed for intelligent gameplay, behaviour adaptation, and computer human interaction (Dignum, 2009, Goschnick, 2008, Shukri, 2008).

Sidebar 1: The Agent-Oriented Paradigm.

Game Play Framework. The framework executes the game play script. Many commercial and open source game frameworks are currently available; our framework is novel in that we propose an agent-oriented solution. We believe AOP is an excellent match for the SE and Sys education GDP; our position on this is presented in (Tull, 2011). See Sidebar 1 for more information about AOP.

Game Play Data Repository. This repository logs all game play events that occur while the game is being played.

Player Assessment. This component uses the Game Play Data Repository to automatically analyse a player's accomplishments with respect to the learning objectives. For example, consider a game that has project management learning objective(s) related to scheduling. If the player makes choices in the game that consistently lead to delivering a product late, then this pattern should be identified and reported. In order to do this during the execution of the game (i.e., dynamic) to provide effective feedback, then efficient algorithms to identify and rank the areas for improvement are needed.

Game Play Adaptation. This component uses the Game Play Data Repository to automatically adapt the game play script to the behaviour of the player.

There are a number of cases that could be considered. For example, if a collection of players (e.g., a class) consistently score very highly in one part of the game, then this part of the game could be replaced with a more difficult challenge related to the same learning objectives (static replacement, for future players). Parts of the game related to its learning objective that have not yet been played could be replaced before the player gets to them (dynamic replacement, for the current player).

4.2 Use of the GDP

The use of the GDP is envisioned as follows. The requirements for a specific game (e.g., a game covering learning objectives related to the agile method Scrum, requirements engineering, or software architecture) are captured using the Game Play IDE. This is a manual step, in which the game designer instantiates a template for the specific simulation training game; the game designer can tailor the template.

The template has a collection of questions to assist the designer (helping to improve the consistency and completeness of the game play requirements). For our project, the template is represented in XML; however alternative representations are possible. Each question helps to specify the behaviour of GDP components.

Template questions begin at high-level, non-functional learning objectives. Questions like, "Will the player learn about human resources?"

Answers open new, more functional and more specific questions, "What personality qualities do non-player characters (NPC's) have?"

At their most specific, the questions transition to strictly functional, "Which of the following 3 random interactions are possible between NPC's during a project meeting?"

Other requirements that make up the "what, when and where" of the scenario are configured by a similar vein of questions. For example:

- "What is the setting?"
  - "What is the building floor plan?"
    - "Which office is the player's?"
    - "How big is the office?"
  - "When is it?"
    - "What time of day?"
    - "What day of week?"
  - "What is the narrative background?"
    - "What is the player's history?"
    - "What is the player's job?"
- "Who are the NPC's involved?"
  - "How many are there?"
  - "What do they look like?"
  - "What are their names?"

Answers will shape the scenario. Specifying the size of the player's office tells the Game Play Framework to graphically display an office of that size, and constrains how the player can move in this space. Choosing "Human Resources" as a learning objective configures new assessment criteria in the Game Play Assessment component, which will now monitor how the player interacts with NPC's, and how NPC's interact with each other. Selecting which random events are possible during the scenario configures new possibilities and strategies in the Game Play Adaptation component, which now has new actions at its disposal.

4.3 Discussion

The GDP we are proposing has a number of unique, novel contributions:

- Educational Game Specification Template. Templates capture expertise, improve consistency and completeness, and support the modularization of game specifications. They can be tailored to provide flexibility. The use of templates is an established best practice in SE/Sys engineering processes. Game designers can instantiate the templates using a WYSHIWYG UI.
- Agent-oriented game engine. Agents are well suited to embody intelligent gameplay, behaviour adaptation, and computer human interaction properties; these support the development of fun and engaging games for SE/Sys education.
- Automated assessment of the player's accomplishments.
- Automated, self-adaptation of the game play script.

Furthermore, the SimSys GDP provides many of the facilities necessary to support "good" game characteristics:

- **Interactive Embedded Tutorials.** Game scripts provide the capacity to create in-game tutorials.
- **Multiple Real-time Strategy Scenarios**. The Game Adaptation component supplies the necessary functionality for scenarios that change and modify themselves dynamically, creating novel experiences for the player.
- **Self-assessing Framework.** The Game Assessment component is designed to provide all the facilities necessary to identify player progress and generate immediate feedback.

- **Graphical Sophistication.** The Game Framework provides capabilities for 2D graphics and animation, and can support a wide range of interfaces (simpler to more complex).
- **Multiple Levels of Difficulty**. Game Adaptation provides one mechanism for changing difficulty. Other mechanisms are planned.
- **Multiple Players**. SimSys does not currently support multiplayer games. This is planned as an avenue of future research.
- **External-assessing Framework.** Game Assessment also has the facilities to identify player progress, and report assessment metrics to an instructor or teacher.
- **Clearly Identifying In-game Objectives with Course Learning Outcomes.** Game IDE Templates straightforwardly ask which learning outcomes the scenario designer is seeking. Armed with this knowledge, scenarios are in a better position to inform the player of pedagogical context.
- **Curriculum Guide for Faculty.** The Game IDE provides a simple, structured method which guides scenario designers in the construction of particular lessons.

5 RELATED WORK

The related work presented here is narrowly restricted to SE Education Game literature, due to space restrictions. SimSE is a game designed to simulate the software development process from a project management perspective. The game major components are the Model Builder, the Model Generator, and the Simulation Environment. The Model Builder allows the instructor to create a model according to specific characteristics that he/she want the students to learn. The Model Builder allows the instructor to specify the life cycle and the specifics of the project. The Generator uses the model specifications to create the scenarios for the game and the player (student) acts as a project manager, making decisions for tasking available employees, acquire new tools or use available tools, etc. The player can then advance the clock and observe the consequences of the decisions, which are made by the Simulation Environment. The overall goal of the game is to finish a project with good quality and within the available budget and schedule. A final score from 0 to 100 (100 means all the goals have been achieved) determines the level of success of the student. From a game design perspective there are also some major issues with SimSE. First, the game is not very interactive and after making some decisions the player advances the clock and analyzes the results; the interface design is quite static (the characters do not move). While these may be viewed as small issues in terms of research, they have a significant impact on game usability and adoption.

SESAM is also a simulation tool for the management of the software development process. It has the same goals as SimSE but lacks a graphical interface. SESAM uses a new defined language that allows the users to give commands as they play and also to create new models/projects. The drawbacks related to SimSE are also present in the SESAM project.

6 CONCLUSIONS AND FUTURE WORK

We contend that creating a game that simulates specific, key elements in software engineering education within an interactive student-centred environment rather than a passive content-centred environment will lead to higher success for SE students. The paradigm shift to which this game contributes will augment faculty resources so that more course time can be spent reflecting on issues that players had while working towards the course goals instead of faculty providing content delivery. This will help students become life-long learners, practice deeper problem-solving skills, and enhance their ability to communicate about SE in a professional context (Longstreet, 2011). Because the game environment will better motivate students to repeatedly practice outside of class time and challenges them to improve where they specifically have a need, higher-risk students will have better chances of success in SE courses.

Our simulation based game offers a concrete, effective and efficient way to implement the curriculum guidelines mandated in SE2004. It is an opportunity to augment a classroom experience by adding a dynamic, student-centred and thought-provoking approach to SE education. As the game can be made freely available for universal adoption, it could be used to supplement the curricula at smaller institutions or schools with more limited access to broad SE expertise such as community colleges, rural schools and institutions that serve under-represented communities.

We see future work in this project with the continued development of the GDP and an example scripted game. The GDP needs to be extended to support more sophisticated games that are multiplayer, have multiple roles (e.g. tester), and multiple levels of difficulty. Finally, assessing student progress in courses that use the game will be key to identify the specific strengths and limitations of a simulation based game approach.

REFERENCES

Alexander, P. A. and Judy, J. E., 1998. The interaction of domain-specific and strategic knowledge in academic performance. *Review of Educational Research*.

Alvermann, D., Smith, I. C. and Readance, J. E., 1985. Prior knowledge activation and the comprehension of compatible and incompatible text. *Reading Research Quarterly*.

Anderson Jr., E. G., and D. J. Morrice. 2000. Simulation game for teaching service-oriented supply chain management: Does Information Sharing Help Managers with Service Capacity Decisions? Production and Operations Management 9: 40-55.

Blessing, S. B. and Anderson, J. R., 1996. How people learn to skip steps. *Journal of Experimental Psychology: Learning Memory and Cognition*.

Chen, F. and R. Samroengraja. 2000. The stationary beer game. *Production and Operations Management* 9:19-30.

Díaz-Herrera J. and Hilburn, T. (editors), Software Engineering 2004 Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering, A Volume of the Computing Curricula Series, August 23, 2004, The Joint Task Force on Computing Curricula, *IEEE Computer Society and the Association for Computing Machinery*.

Dignum, F., Westra, J., van Doesburg,W. A., and Harbers, M., 2009. Games and Agents: Designing Intelligent Gameplay. *International Journal of Computer Games Technology*.

Gee, J. P., 2003. *What video games have to teach us about learning and literacy*, Macmillan, USA.

Goschnick, S., Balbo, S. and Sonenberg, L, 2008. ShaMAN: An Agent Meta-model for Computer Games, in *Proceedings 2nd Conference on Human-Centered Software Engineering*.

Horn, R. E., and A. Cleaves. 1980. *The Guide to Simulation/Games for Education and Training.* NewburyPark, CA: Sage Publications

Longstreet, C. S. and Cooper, K., 2011. Using Games in Software Engineering Education to Increase Student Success and Retention. In *Proceedings 2011 Conference on Software Engineering Education and Training*.

Ludewig, J., Bassler, T., Deininger, M., Schneider, K., Schwille, J., 1992. SESAM-simulating software projects. In *Proceedings Fourth International Conference on Software Engineering and Knowledge Engineering*.

Malone, T., 1980. What makes things fun to learn? Heuristics for Designing Instructional Computer Games. In *Proceedings of the 3rd ACM SIG SMALL Symposium and the First SIGPC Symposium Small Systems*.

Mustafee, N. and Katsaliaki, K. The Blood Supply Game. In B. Johansson, S. Jain, J. Montoya-Torres, J. Hugan, and E. Yücesan, (Eds), *Proceedings of the 2010 Winter Simulation Conference*. 327-38.

Nadolski, R. J., Hummel, H. G. K., van den Brink, H. J., Hoefakker, R. E., Slootmaker, A., Wu, B., and Bakken, S. K., 2010. Experiences from Implementing a Face-to-Face Educational Game for iPhone/iPod Touch. In *Proceedings 2nd International IEEE Consumer Electronics Society's Games Innovation Conference*.

O'Malley, S. and DeLoach, S, 2001. Determining When to Use an Agent-Oriented Software Engineering Paradigm. In *Proceedings of the Second International Workshop On Agent-Oriented Software Engineering*.

Riis, J. O., 1995. Simulation Games and Learning in Production Management. *International Federation for Information Processing*. Springer.

Shukri S. and Shaukhi, M., 2008. A Study on Multi-Agent Behavior in a Soccer Game Domain. *World Academy of Science, Engineering and Technology*.

Tull, A., Smith, T., and Cooper, K., 2011. Towards an Agent-oriented Framework for Serious Game Engines: Architecting with Behavioural Software Agents, SimulTech 2011 (to appear).

Walker, A. and Shelton, B.E., 2008. Problem-based educational games: connections, prescriptions, and assessment. *J. of Inter. Learning Research*.

Wang, A., and André van der Hoek, 2004. SimSE:an educational simulation game for teaching the Software engineering process. In *Proceedings of the 9th annual SIGCSE conference on Innovation and technology in computer science education.*

Weinstein, C. E., Husman, J. and Dierking, D. R., 2000. Self-regulation interventions with a focus on learning strategies. In M. Boekaerts, P. Pintrich and M. Zeidner, (Eds), *Handbook of self-regulation*. Academic Press.

Westera, W., Nadolski, R., Hummel, H. and Wopereis, I., 2008. Serious games for higher education: a framework for reducing design complexity. *J. of Computer Assisted Learning*.

Wooldridge, M. 2009. *Introduction to MultiAgent Systems*, John Wiley & Sons, 2nd edition.