

# Comprehensive Evaluation of an Educational Software Engineering Simulation Environment

Emily Oh Navarro and André van der Hoek  
Donald Bren School of Information and Computer Sciences  
University of California, Irvine  
Irvine, CA 92697-3425 USA  
emilyo@ics.uci.edu, andre@ics.uci.edu

## Abstract

*Software engineering educational approaches are often evaluated only anecdotally, or in informal pilot studies. We describe a unique approach to evaluating a software engineering educational technique (SimSE, a graphical, interactive, customizable, game-based software engineering simulation environment). Our method for evaluating SimSE went above and beyond anecdotal experience and approached evaluation from a number of different angles to form a comprehensive assessment of SimSE's effectiveness. In this paper we demonstrate the insights and lessons that can be gained when using such a multi-angled evaluation approach. Our hope is that, from this paper, educators will: (1) be provided with evidence about the educational effectiveness of SimSE, and (2) learn ideas about how to comprehensively evaluate their own approaches.*

## 1. Introduction

In recent years, software engineering educators have created a wealth of innovative approaches to teaching software engineering. Some of the most frequently proposed ones focus on making the students' class project experience more closely resemble one they would encounter in the real world by doing such things as involving an industrial participant [5], using only maintenance or evolution-based projects [8], or even purposely sabotaging the project [3]. Others believe that the addition of one or more specific software engineering sub-topics to the curriculum (e.g., formal methods [14] or Human-Computer Interaction [6]) is the crucial missing piece. Still others take an approach that has students practice software engineering processes in a computer-based simulated environment [4].

Although the majority of these approaches seem promising, many of them are risky and require a great deal of planning and restructuring to incorporate them into a curriculum. It is therefore necessary that these educational approaches are properly and thoroughly evaluated so that other educators can be provided with enough evidence of their effectiveness to warrant the effort involved in adopting them. However, such extensive evaluations are rarely seen—most are only anecdotes of an approach's usage and/or the occasional brief, small-scale pilot study. Although these types of evaluations provide good initial assessments of an approach's potential (i.e., "this approach works"), they fail to go beyond this and answer questions such as *why* and *how* an approach works, what its flaws are, how it can be made more effective, and what kinds of considerations need to be made concerning its use.

In this paper, we present an example of a comprehensive evaluation of a software engineering educational innovation that showcases the difference between the amount of insight gained through a small pilot study versus that from an extensive, multi-angled evaluation. Our educational innovation is SimSE, a graphical, interactive, customizable,

game-based simulation environment for educating students in software processes. We evaluated SimSE, first in an initial pilot study, then in three subsequent non-pilot experiments, each one focusing on a different aspect of assessment. These were designed to provide a collective picture of SimSE's overall effectiveness and a general understanding of its strengths and weaknesses from multiple angles. It is our hope that the work described in this paper will: (1) provide an example of how software engineering educational approaches can be comprehensively evaluated, and (2) provide evidence for SimSE's effectiveness and guidance about how it can be incorporated into a course.

The remainder of this paper is organized as follows: In Section 2, we briefly describe SimSE. In Section 3, we detail the series of experiments we conducted to evaluate SimSE. Section 4 summarizes the lessons that can be generalized for other software engineering educators evaluating their own approaches, and we present our conclusions and plans for future work in Section 5.

## 2. SimSE

SimSE is a computer-based environment that facilitates the creation and simulation of realistic game-based software process simulation models—models that involve real-world components not present in typical class projects, such as large teams of people, critical decision-making, personnel issues, budgets, and unexpected events. In so doing, it aims to provide students with a platform through which they can experience many different aspects of the software process in a practical manner without the overarching emphasis on creating deliverables that is inherent in actual software development.

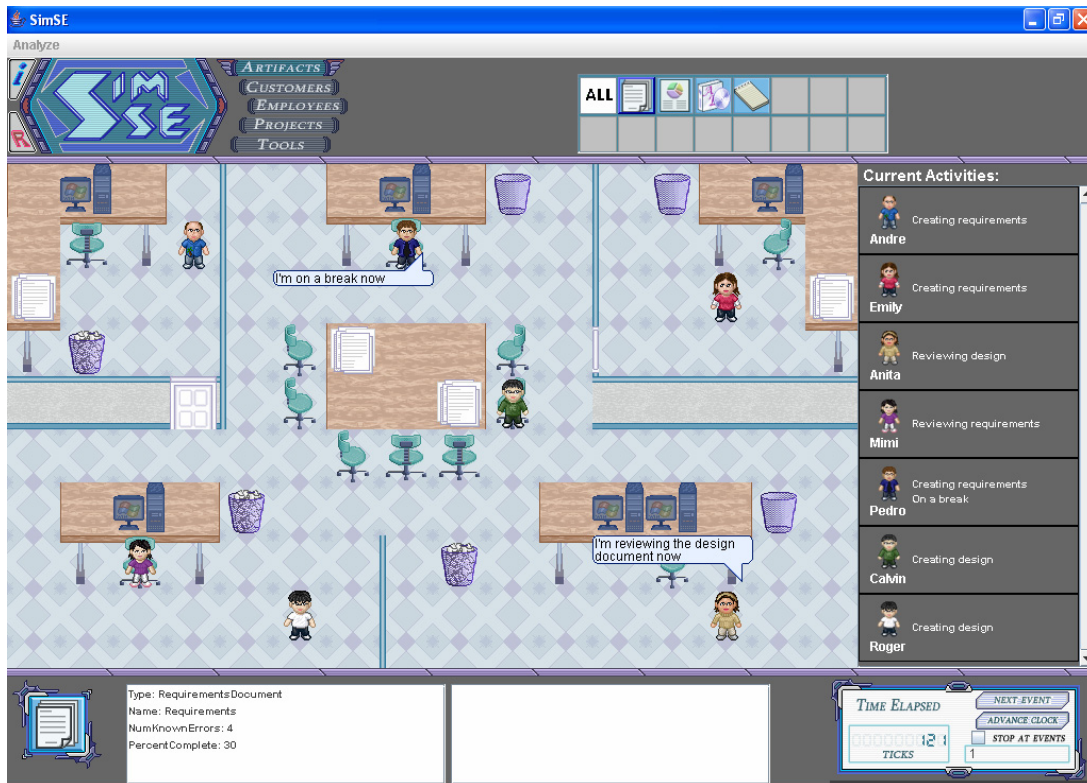
The graphical user interface of SimSE is shown in Figure 1. SimSE is a single-player game in which the player takes on the role of project manager and must manage a team of developers in order to successfully complete (a particular aspect of) an assigned software engineering project. The player drives the process by, among other things, hiring and firing employees, assigning tasks, monitoring progress, and purchasing tools. At the end of the game the player receives a score indicating how well they performed. In addition to the score, an explanatory tool provides students with further information about their game, including which rules were triggered when, a trace of events, and the “health” of various attributes (e.g., correctness of the code) over time.

To date, six SimSE models (and corresponding games) exist: a waterfall model, an inspection model, an incremental model, an Extreme Programming model, a rapid prototyping model, and a Rational Unified Process model. For more information on SimSE, including its design, game play, and simulation models, see [9, 10].

## 3. Approach

For our initial evaluation of SimSE, we conducted a pilot experiment to provide us with an overall understanding of the thoughts, attitudes, and reactions of students who play SimSE, all for the purpose of making an initial judgment about its potential as an educational tool. In addition, we aimed to determine the strengths and weaknesses of SimSE through the feedback of the students who play it. To do this, we had 29 undergraduate students play SimSE for two hours, and fill out a questionnaire about their experience.

In general, students' feelings about the game were favorable (see [10] for full results). On average, students found the game enjoyable and felt it had an appropriate level of difficulty. They also felt that it was quite successful in teaching software engineering process issues, and, for the most part, felt that SimSE would be a helpful part of a software engineering course. Student comments brought to light some needed areas for improvement, such as



**Figure 1: SimSE Graphical User Interface.**

awkward user interface issues and the need for more simulation models of different processes. On the whole, however, this experiment established that SimSE has the potential to be an educationally effective tool in teaching students software process concepts.

Although the results of this pilot experiment were quite positive, they were far from complete—numerous questions remained. Specifically, we still did not know much about *why* and *how* SimSE helps students learn, whether or not it works well in a classroom setting, and what its benefits and drawbacks are compared to other educational methods. Thus, we designed three separate experiments to assess: (1) whether SimSE fits into a traditional software engineering curriculum, (2) how SimSE compares to traditional methods of teaching software process concepts, and (3) which types of learning processes students go through while playing SimSE (i.e., *how* it helps students learn). The remainder of this section describes the design of each experiment in further detail, followed by a summary of their collective results.

### 3.1. In-Class Experiment

Because SimSE was designed to be used in conjunction with a software engineering course as a complement to existing methods, a critical part of our evaluation was to actually use it in such a setting and assess how well it incorporates as a course component. Specifically, we wanted to determine how much students learn from using SimSE in class, how they perceive and feel about the experience, and what kinds of practical considerations instructors should keep in mind when using SimSE in their courses.

Since this was the first time SimSE was being used in the context of a course and we were unsure about how it would work in such a setting, we thought it appropriate to make it an extra-credit rather than compulsory exercise. Hence, we made it a moderate extra-credit assignment, worth 7.5% of the final grade.

We used SimSE over two offerings of the introductory software engineering course at UC Irvine. The students were given the assignment to play three SimSE models and answer a set of questions concerning the concepts the models are designed to teach. These questions were written in such a way that the students had to play the game (often repeatedly) in order to find out the answer. Thus, their answers would provide us with a measure of how much they had learned from playing the game. In addition to questions about the models, students were asked to complete a questionnaire about their experience, similar to the one used in the pilot experiment, so that we could gauge what effect the change of setting (from inconsequential experiment to graded class exercise) had on students' perceptions of SimSE.

### **3.2. Comparative Experiment**

The goal of our comparative experiment was to try and discover how SimSE compares to traditional teaching methods (reading from a textbook and hearing lectures). In particular, we aimed to compare the effectiveness of each method in teaching a specific set of software process concepts, as well as other aspects underlying the learning process—both practical aspects such as time spent and subjective aspects such as student attitudes and motivation. With this comparison, one would be able to make an informed judgment about whether SimSE would truly be a useful addition to a course—an addition more useful than one that included extra traditional assignments such as readings or lectures.

The subjects for this experiment consisted of 19 undergraduate students, 12 who had taken an introductory software engineering course, and 7 who had not. This particular mix of educational experience was chosen for the following reason: SimSE is meant to be used as a complement to existing teaching methods, so it assumes some background knowledge of basic software engineering concepts, and hence, the target population is those students who have taken at least one introductory software engineering course. However, students who have taken a software engineering course will have already been taught (through textbooks and lectures) much of the material that was taught in this experiment using textbooks and lectures. Hence, creating a mix of students from the two different experience levels creates a balance addressing both of these concerns, as well as helps provide some insight into how SimSE does as a teaching tool for those who have no software engineering experience.

The students were randomly divided into three groups (SimSE, reading, and lecture) of approximately equal size, with the condition that in each group roughly half of the people had passed the software engineering course while the other half had not. All subjects were given a pre-test designed to measure their knowledge in the software process concepts that were to be taught using the three methods. At the completion of the test, all subjects were then given instructions about the assignment to complete (SimSE, reading, or lectures). For the next four days, the SimSE group was expected to play three SimSE models, the reading group was expected to complete a set of readings that covered roughly the same software process concepts embodied in the SimSE models, and the lecture group was expected to attend two 50-minute lectures about the same concepts. At the end of the four days, the subjects were given a post-test which contained some of the same questions as those in the pre-test, but also included some different questions to mitigate any bias or foreknowledge.

Also at the end of the experiment, all subjects were asked to fill out a questionnaire regarding their thoughts and feelings about the instructional method in which they participated. This questionnaire contained two main types of questions: The first type of question asked them about the teaching/learning method used, including how much time they spent on the exercise, how much they enjoyed it, how effective they felt it was, and their preference for that method compared to other methods. The second type of question asked them to provide some

background information, which would allow us to detect any correlations between such variables as experience level or gender and the subject's performance on the learning exercise.

### 3.4. Observational Experiment

For our final experiment we conducted an in-depth observational study in which we observed students playing SimSE and interviewed them about their experience. The primary purpose of this study was to investigate the learning processes students go through when playing the game—namely, *how* SimSE helps people learn. We designed SimSE with a number of learning theories in mind (specifically, Learning by Doing [11], Situated Learning [2], Keller's ARCS [7], Discovery Learning [1], and Learning through Failure [12]), and student responses from the first three experiments hinted that some of these were being employed, but no further investigation into the presence of these theories was yet undertaken. In this experiment, therefore, we specifically set out to detect which of these (and other) learning theories actually come into play in the learning process of a SimSE player.

This experiment also had a secondary purpose: to evaluate how well the explanatory tool achieves its goals of aiding students in understanding their score, helping them recognize where they went wrong and/or right in the approach they took, and assisting them in planning a successful approach to the next run of the game. This was done by having some students play SimSE with the explanatory tool and some without, and noting the differences in their behavior, attitudes, and opinions.

For this experiment, 11 undergraduate students who had passed an introductory software engineering course participated. This experiment occurred in a one-on-one setting—one subject and one observer. Each subject was first given instruction on how to play SimSE, and was then observed playing SimSE for 2.5 hours. Eight subjects played with the explanatory tool and three played without. While the subject was playing, their game play and behavior were observed and noted. Following this, the subject was interviewed about their experience. In addition to any spontaneous questions the observer formulated based on a particular subject's behavior, all subjects were asked a set of standard questions. Several of the questions were designed to detect the presence of one or more learning theories in the subject's learning process. Some questions did not target a particular theory, but were instead meant to evoke insightful comments from the subject from which various learning theories could be detected, and from which general insight into the learning process could be discovered. The interview also included questions that were specifically designed for comparison between the subjects who used the explanatory tool and those who did not. Following the experiment, the interviewer's observations and interview notes were analyzed to try to discover which learning theories were employed, and how, as well as to discover any other insights about SimSE as a teaching tool that could be gained from the data.

### 3.5 Summary of Results

Although the specific results from each experiment are valuable in and of themselves, it is even more useful to focus on the collective lessons learned from the experiments taken together as a whole. Thus, we present here the highlights of the most significant lessons and insights we have learned about SimSE's abilities as an educational tool (full results, including numerous data analyses, graphs, and more detail on experiment setups can be found in [9]).

- *Students who play SimSE seem to successfully learn the concepts it is designed to teach.* Students in the pilot experiment felt that SimSE was effective at teaching software process concepts. Students from the in-class experiment were quite successful at answering questions about these concepts correctly. In the comparative experiment, there

was a strong correlation between reported time spent playing SimSE and increase in software process knowledge. All subjects in the observational experiment were able to recount learned concepts and improve their scores from game to game.

- *Students find playing SimSE a relatively enjoyable experience.* Students in all experiments reported that they enjoyed playing SimSE for the most part. Several students in the observational experiment were visibly enthralled with the game.
- *Providing students with adequate and proper instruction in playing SimSE is critical.* Subjects from the in-class and comparative experiments felt that they did not receive enough guidance to succeed in SimSE. As a result, they experienced some frustration and confusion. In the comparative experiment, no subject in the SimSE group played as much as they were assigned, partially due to this frustration. In the observational experiment, it was observed that subjects tended to miss important information if it was not sufficiently emphasized in the instructions. Thus, instruction must be a carefully planned part of SimSE's use, and should include such extensive measures as holding training sessions and/or providing paper-based handouts.
- *Students find SimSE repetitive when played for extended periods of time.* Although it was clear from the comparative experiment that the longer a student plays SimSE, the more they learn, both the comparative experiment and the in-class usage revealed that a longer playing time also contributes to a feeling of repetitiveness. Because the version of SimSE used in these experiments included neither the explanatory tool nor adequate instructions, it is anticipated that the addition of these two factors will lessen the need for so many repetitions of the same model when used in classes in the future.
- *Students learn through playing SimSE by employing the theories of Discovery Learning, Learning through Failure, Constructivism, Learning by Doing, Situated Learning, and Keller's ARCS.* These were the learning theories that were most evident in the observational experiment. Thus, future versions of SimSE, as well as educational simulations in general, should be designed with these theories in mind, aiming to maximize the characteristics that are known to promote each one.
- *SimSE is most educationally effective when used as a complementary component to other teaching methods.* The results of our experiments strongly suggested that a certain level of existing software process knowledge must be possessed by a student in order for maximal learning to be promoted. In the comparative experiment, students in the SimSE group who had no previous exposure to software engineering had the overwhelmingly worst improvement from pre- to post-test, compared to other subjects. Their lack of background knowledge, combined with the inadequate instruction in learning to play SimSE that was given, resulted in the SimSE group improving least from pre- to post-test, compared with the lecture group and reading group (which improved most). In the observational experiment, it was discovered that the opportunity to put previously learned knowledge into practice was a major learning-facilitating characteristic of SimSE. Thus, SimSE should be used with other teaching methods that provide this required knowledge, and should not be used as a standalone tool.
- *The explanatory tool is a needed and useful part of SimSE that helps players understand the reasoning behind their score.* The most frequent complaint of the students who played SimSE without the explanatory tool (in all four experiments) was the lack of feedback given about their performance in the game. Students who played SimSE with the explanatory tool (in the observational experiment) overall found it to be a helpful resource for understanding their score and the simulated process.

## 4. Lessons Learned

Each software engineering educational technique is, of course, different, and therefore requires a different evaluation plan. However, it is our hope that researchers can use the design presented here as an inspiration for the kinds of questions and considerations that should be addressed when formulating evaluation plans for their own techniques. Nonetheless, there are some general lessons about evaluating software engineering educational approaches that can be gleaned from our experience and applied to a wide variety of situations.

One of the most important lessons we learned was that surprises happen, and experiments rarely turn out exactly as planned. A prime example of this was our comparative experiment. The first surprise in this experiment was that 11 of the 30 initial subjects either dropped out during the course of the experiment or failed to show up at all, leaving us with far smaller treatment groups than originally planned. The second surprise was that none of the subjects in the SimSE group completed their assignment, while most of the subjects in the other groups did. Although our original intentions for the experiment were somewhat thrown off by these surprises, by analyzing the data from every possible angle we were still able to learn a tremendous amount. Moreover, the fact that none of the SimSE subjects completed the assignment was actually an important piece of information that revealed some significant insights about SimSE (its need to be used complementary to other methods, its inadequate instructions, and its repetitiveness when used without the explanatory tool). Thus, researchers should keep open minds when unforeseen events occur in the course of their experiments. In such cases, one should dig deep into the data and look for unexpected trends while remembering that, in spite of these surprises (and often because of them), a great deal can still be learned.

We also learned that one-on-one observation and interview is a highly powerful technique yielding numerous insights that are simply impossible to discover in a group setting. For example, while the in-class and comparative experiments suggested that the instruction and guidance students were receiving in learning to play SimSE was inadequate in some way, the observational experiment revealed the exact issues that needed to be emphasized in order for the students to have a more successful experience. Framing our observational experiment in the context of learning theories was an especially valuable choice, as it provided us with a well-established framework for discovering some of the specific ways in which SimSE facilitates learning. A learning-theory-centric observational and interview experiment design is one that could be applied to nearly any software engineering educational approach.

Clearly, designing and carrying out an extensive evaluation like ours is a difficult and labor-intensive endeavor with countless considerations to be made and details to be addressed. Experiments must be designed, subjects must be recruited, and significant time must be invested in conducting the experiments and analyzing the data. However, as we have shown here, this is not an impossible task, and the return on investment is significant in terms of the amount of insight that can be gained.

## 5. Conclusions and Future Work

The typical software engineering educational approach is usually evaluated only anecdotally or through a small pilot study. As a result, much of what there is to be learned about an approach's value goes undiscovered. We have presented a clear example of this: In choosing to extend SimSE's evaluation beyond an initial pilot study into a multidimensional evaluation approach, we were able to discover numerous insights about its effectiveness, its limitations, the critical considerations surrounding its use, and promising directions for future work.

If more software engineering education researchers aim to comprehensively evaluate their approaches, the community will be provided with more convincing evidence about each ap-

proach's efficacy, more frequent sharing and adoption of approaches will be encouraged, approaches will be made more effective through experience, and a greater overall understanding of how software engineering can be taught and learned will be promoted. As a result, the field of software engineering education will progress to become more effective in preparing students for their future careers.

Though our experiment results have provided answers to our initial questions, they have also raised new questions and brought to light issues that need to be addressed, both of which must be dealt with through further experimentation. For example, we are planning to experiment with certain modifications to in-class usage, such as making SimSE a mandatory exercise and increasing the level of instruction students receive in learning to play SimSE. We will also perform further observational experiments with new and revised simulation models and versions of SimSE, to assess the value of these revisions.

## Acknowledgements

Effort partially funded by the National Science Foundation under grant number DUE-0618869.

## References

- [1] S. M. Alessi and S. R. Trollip, *Multimedia for Learning*. Needham Heights, MA, USA: Allyn & Bacon, 2001.
- [2] J. S. Brown, A. Collins, and P. Duguid, "Situated Cognition and the Culture of Learning," *Educational Researcher*, vol. 18, pp. 32-42, 1989.
- [3] R. Dawson, "Twenty Dirty Tricks to Train Software Engineers," in *Proceedings of the 22nd International Conference on Software Engineering*: ACM, 2000, pp. 209-218.
- [4] A. Drappa and J. Ludewig, "Simulation in Software Engineering Training," in *Proceedings of the 22nd International Conference on Software Engineering*: ACM, 2000, pp. 199-208.
- [5] R. J. Fornaro, M. R. Heil, and A. L. Tharp, "What Clients Want - What Students Do: Reflections on Ten Years of Sponsored Senior Design Projects," in *Proceedings of the Nineteenth Conference on Software Engineering Education and Training*. Oahu, HI: IEEE, 2006, pp. 226-236.
- [6] O. Hazzan and J. E. Tomayko, "Reflection Processes in the Teaching and Learning of Human Aspects of Software Engineering," in *Proceedings of the Seventeenth Conference on Software Engineering Education and Training*. Norfolk, VA, USA: IEEE, 2004, pp. 32-38.
- [7] J. M. Keller and K. Suzuki, "Use of the ARCS Motivation Model in Courseware Design," in *Instructional Designs for Microcomputer Courseware*, D. H. Jonassen, Ed. Hillsdale, NJ, USA: Lawrence Erlbaum, 1988.
- [8] J. C. McKim and H. J. C. Ellis, "Using a Multiple Term Project to Teach Object-Oriented Programming and Design," in *Proceedings of the Seventeenth Conference on Software Engineering Education and Training*. Norfolk, VA: IEEE, 2004, pp. 59-64.
- [9] E. O. Navarro, "SimSE: A Software Engineering Simulation Environment for Software Process Education." Irvine, CA: University of California, Irvine, 2006.
- [10] E. O. Navarro and A. van der Hoek, "Design and Evaluation of an Education Software Process Simulation Environment and Associated Model," in *Proceedings of the Eighteenth Conference on Software Engineering Education and Training*. Ottawa, Canada: IEEE, 2005.
- [11] C. R. Rogers, *Freedom to Learn*. Columbus, OH, USA: Merrill, 1969.
- [12] R. C. Schank, *Virtual Learning*. New York, NY, USA: McGraw-Hill, 1997.
- [13] D. A. Umphress and J. A. Hamilton, "Software Process as a Foundation for Teaching, Learning, and Accrediting," in *Proceedings of the Fifteenth Conference on Software Engineering Education and Training*. Covington, Kentucky, USA: IEEE, 2002, pp. 160-169.
- [14] B. R. von Kinsky, M. Robey, and S. Nair, "Integrating Design Formalisms in Software Engineering Education," in *Proceedings of the Seventeenth Conference on Software Engineering Education and Training*. Norfolk, VA, USA: IEEE, 2004, pp. 78-83.