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 more comprehensive 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 through a family of studies designed to assess SimSE's effectiveness and guide its development. 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) learn ideas about how to more comprehensively evaluate their own approaches, and (2) be provided with evidence about the educational effectiveness of SimSE.

1. Introduction

In recent years, software engineering educators have created a wealth of innovative approaches to teaching software engineering. 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, both summatively (so that other educators can be provided with enough evidence of their effectiveness to warrant the effort involved in adopting them) and formatively (so that the developers of the approaches can make them more effective in an informed and directed manner). However, such extensive evaluations are rarely seen—most consist of 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 seems to work"), 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 multi-angled 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 a more comprehensive evaluation. Our educational innovation is SimSE, a graphical, interactive, customizable, game-based simulation environment for educating students in software processes [7]. We evaluated SimSE, first in an initial pilot study, then in three subsequent studies, each one focusing on a different aspect of evaluation. 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, so that we could learn how to make it more effective. It is our hope that the work described in this paper will: (1) provide an example of how software engineering educational approaches can be more 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 evaluations we conducted to investigate SimSE's value as a teaching tool. Section 4 discusses important issues involved in software engineering educational evaluation and lessons learned from our experience. Finally, 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 an assigned software engineering project or task. 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, and an explanatory tool provides them with a visual analysis of 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 [6, 7].

3. Approach

For our initial evaluation of SimSE, we conducted a pilot study 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, and for guidance in the further development of SimSE. 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 [7] 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 awkward user interface issues and the need for more simulation models of different processes. On the whole, however, this pilot study established that SimSE has the potential to be an educationally effective tool in teaching students software process concepts.

Although the results of this pilot study 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 evaluations 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). All of these studies were partially formative, as their results

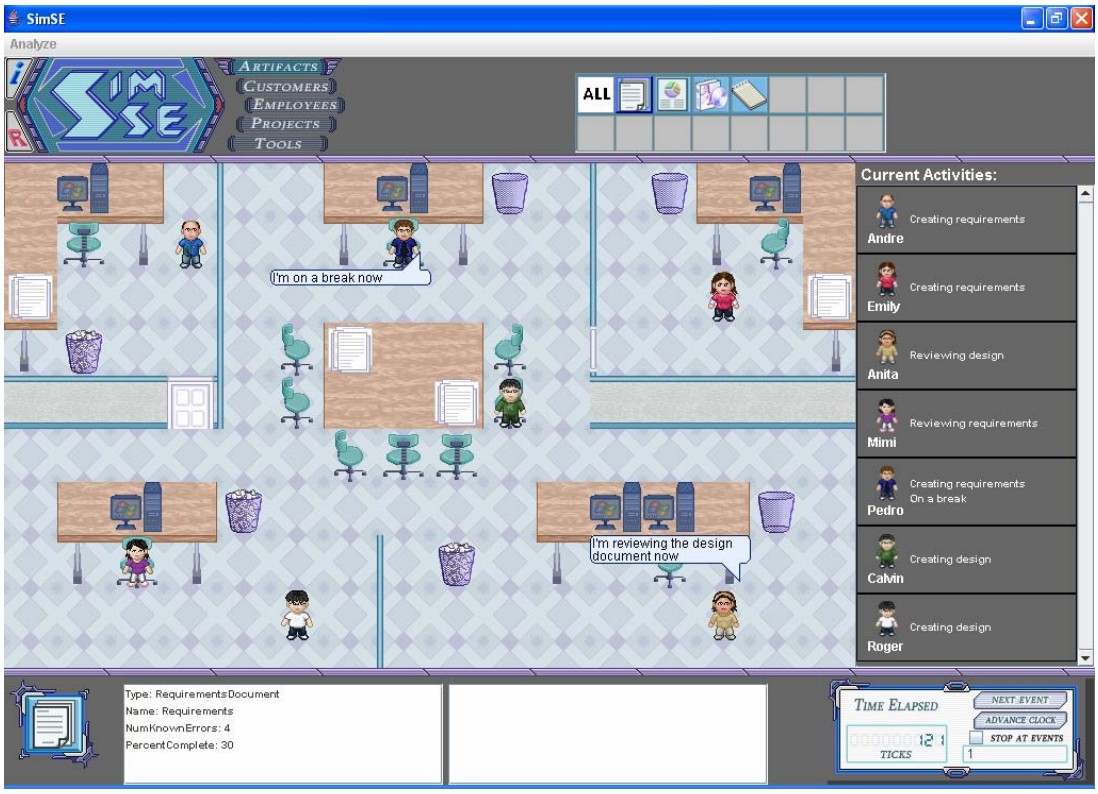

Figure 1: SimSE Graphical User Interface.

informed the continued development of SimSE, which was ongoing during the life of these evaluations. The remainder of this section describes the design of each evaluation in further detail, and then presents a summary of their collective results.

3.1. In-Class Evaluation

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 could be incorporated 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

study, 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 Study

The goal of our comparative study 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.

The subjects 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 this knowledge. 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 study using textbooks and lectures. Hence, using a mix of students from the two different experience levels created a balance addressing both of these concerns, as well as provided some insight into SimSE's value 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.

At the end of the study, 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.

3.3. Observational Study

For our final evaluation 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 [8], Situated Learning [3], Keller's ARCS [4], Discovery Learning [1], and Learning through Failure [9]), and student responses from the first three evaluations hinted at the presence of some of these, but they were not

looked into any further up until this point. In this study, therefore, we specifically set out to detect which of these (and other) learning theories are actually involved in the learning process of a SimSE player, and how.

Eleven undergraduate students who had passed an introductory software engineering course participated. This study 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. 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 that could be used to detect the presence of various learning theories and discover general insights into the learning process. Following the study, the interviewer's observations and interview notes were analyzed to try to discover which learning theories were involved, and how, as well as to discover any other insights about SimSE as a teaching tool that could be gleaned from the data.

3.4 Summary of Results

Although the specific results from each study are valuable in and of themselves, it is even more useful to focus on the collective lessons learned from the evaluations 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 each evaluation setup can be found in [6]).

- *Students who play SimSE seem to successfully learn the concepts it is designed to teach.* Students in the pilot study felt that SimSE was effective at teaching software process concepts. Students who played SimSE in class were quite successful at answering questions about these concepts correctly. In the comparative study, there was a strong correlation between reported time spent playing SimSE and increase in software process knowledge (Pearson $r=0.81$, $p<0.001$). All subjects in the observational study 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 evaluations reported that they enjoyed playing SimSE for the most part. Several students in the observational study 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 studies felt that they did not receive enough guidance to succeed in SimSE. As a result, they experienced some frustration and confusion. In the comparative study, no subject in the SimSE group played as much as they were assigned, partially due to this frustration. In the observational study, it was clear 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 study that the longer a student plays SimSE, the more they learn, both the comparative study 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 evaluations included neither the explanatory tool nor adequate in-

structions, 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.

- *The learning process of a SimSE player involves 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 study, suggesting the applicability of SimSE to different types of learners. Moreover, this information provides us with well-defined guidelines for maximizing SimSE's effectiveness, as we can aim to maximize the characteristics that are known to promote each of these theories.
- *SimSE is most educationally effective when used as a complementary component to other teaching methods.* The results of our evaluations strongly suggested that a certain level of existing software process knowledge must be possessed by a student in order for maximal learning to occur. As mentioned in Section 3.2., some of the students in the SimSE group in the comparative study had no prior exposure to software engineering. These students had the overwhelmingly worst improvement from pre- to post-test, compared to other subjects. Their lack of background knowledge, combined with the inadequate instruction given in learning to play SimSE, 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 study, it was discovered that the opportunity to put previously learned knowledge into practice (i.e., Learning by Doing and Constructivism) 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.

4. Discussion

Evaluation in the domain of education is a difficult endeavor. Due to the multiple interacting factors intrinsic to the educational environment, in most cases it is impossible to adequately isolate the effects of an educational technique [5]. Additional tradeoffs inherent to educational evaluation include ethical issues involved in differential treatment, the challenge of tracking students longitudinally, and the difficulty of getting statistically significant numbers of subjects [2]. The subject of software engineering adds even more challenges: Unlike concrete, well-understood subjects like math and reading, assessment of software engineering skills is less straightforward and best detected when a student enters the working world, which can only be measured through longitudinal studies. On top of this, the immaturity of the domain of software engineering creates difficulties in conducting comparative evaluations. In a maturing domain such as ours, new approaches are usually designed to teach something that is not taught by any existing technique [10]. In other words, a new technique is usually meant to be a *supplement* to a curriculum, not a *replacement* of something else. In our experience, we found it impossible to find an alternative technique that taught the exact same skills that SimSE is designed to teach. Our comparative study did the closest approximation possible, but was, in the end, not close enough to yield many meaningful comparable results.

Despite these difficulties, we must still do all we can to explore the abilities of our educational techniques. Because anecdotal usage alone does not provide much information, and traditional comparative experiments can be problematic, a multi-angled evaluation approach, partially rooted in educational theory, can be a useful solution, as can be seen from our experience. This type of evaluation approach has three major benefits: (1) It provides a more comprehensive picture of a technique's effectiveness, (2) It reveals numerous issues, invaluable to the educational worth and improvement of the technique, that need to be addressed in

order to make a technique more effective, and (3) It provides an overall greater understanding of how learning occurs through a technique, as well as insights into learning in general.

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 evaluations rarely turn out exactly as planned. A prime example of this was our comparative study. The first surprise in this study was that 11 of the 30 initial subjects either dropped out during the course of the study or failed to show up at all, leaving us with far smaller groups than originally planned. The second surprise was that none of the subjects in the SimSE group completed their assignment, while nearly all of the subjects in the other groups did. Although our original intentions for the study were somewhat thrown off by these surprises, analyzing the data from every possible angle allowed us to discover some important, although unintended, insights. Moreover, the fact that none of the SimSE subjects completed the assignment was actually an important piece of information that revealed some significant lessons 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 evaluations. 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 studies suggested that the instruction and guidance students were receiving in learning to play SimSE was inadequate in some way, the observational study revealed the exact issues that needed to be emphasized in the instructions. Framing our observational study 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. Moreover, seeking to exploit the qualities known to promote each of these theories is a useful way to enhance SimSE's effectiveness. A learning-theory-centric observational and interview evaluation design is one that could be applied to a wide variety of software engineering educational approaches.

Another key insight we gained was that using evaluations as formative (using the results to learn about how to effectively improve and further develop the approach) is just as valuable as using them as summative (determining how effective the approach is). Likewise, theory-*building* evaluations (discovering why and how things work) are just as valuable as theory-*proving* evaluations (discovering if things work). Taking this sort of approach resulted in a notably better version of SimSE (after the last study was done and the results were incorporated), than the version we started with.

Clearly, designing and carrying out a multi-angled evaluation like ours is a difficult and labor-intensive endeavor with countless considerations to be made and details to be addressed. Studies must be designed, subjects must be recruited, and significant time must be invested in conducting the evaluations and analyzing the data. However, as we have shown here, this is not an impossible task, and the return on investment is significant.

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 family of evaluations, 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 a greater number of software engineering education researchers aim to more comprehensively evaluate their approaches, the community will be provided with more convincing evidence about each approach'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 evaluation 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 studies. 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 studies 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] V. L. Almstrum, N. Dale, A. Berglund, M. Granger, J. Currie Little, D. M. Miller, M. Petre, P. Schragger, and F. Springsteel, "Evaluation: Turning Technology from Toy to Tool," in *Proceedings of the 1st Conference on Integrating Technology into Computer Science Education (ITiCSE '96)*. Barcelona, Spain, 1996, pp. 201-217.

[3] J. S. Brown, A. Collins, and P. Duguid, "Situated Cognition and the Culture of Learning," *Educational Researcher*, vol. 18, pp. 32-42, 1989.

[4] 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.

[5] M. McNabb, M. Hawkes, and U. Rouk, "Critical Issues in Evaluating the Effectiveness of Technology Conference Summary," in *National Conference on Educational Technology*. Washington, D.C., 1999.

[6] E. O. Navarro, "SimSE: A Software Engineering Simulation Environment for Software Process Education." Irvine, CA: University of California, Irvine, 2006.

[7] 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.

[8] C. R. Rogers, *Freedom to Learn*. Columbus, OH, USA: Merrill, 1969.

[9] R. C. Schank, *Virtual Learning*. New York, NY, USA: McGraw-Hill, 1997.

[10] M. Virvou, G. Katsionis, and K. Manos, "Combining Software Games with Education: Evaluation of its Educational Effectiveness," *Educational Technology & Society*, vol. 8, pp. 54-65, 2005.