Towards Game-Based Simulation as a Method of Teaching Software Engineering

Emily Oh[1] and André van der Hoek [2]

Abstract *— A typical software engineering course consists of a series of lectures along with a small associated class project. Although this may seem like a logical approach, practical, didactic, and timing reasons necessarily lead to a lack of an in-depth treatment of the critical issues involved in the overall process of software engineering. This paper introduces and lays out our plans for constructing SimSE, a graphical, interactive, educational software engineering simulation environment that teaches the software process in a practical manner without the constraints of an actual class project. We anticipate that the use of SimSE will enable students to form a concrete understanding of the software process by allowing its users to explore different approaches to managing the software process and giving them insight into the complex cause and effect relationships underlying the process.*

Index Terms — Simulation, Software engineering education

1. INTRODUCTION

Software engineering is typically introduced to students in the following manner: general theory is presented in a series of lectures and put into (limited) practice in an associated class project. Although at first this seems like a reasonable approach, it is insufficient, on its own, to effectively communicate the complex, fundamental dynamics of real-world software engineering processes. In particular, lectures allow only passive participation by students, and although class projects provide a more hands-on experience, they are inherently constrained by time, scope, and a dominating focus on deliverables, rather than on the process of software engineering. Essentially, then, what is needed is a way to teach the software process in a practical manner without any of the drawbacks involved in an actual class project.

This position paper briefly describes our initial approach to designing and building an educational software engineering simulation environment, SimSE, that directly addresses this need by allowing students to play and experiment with the software engineering process in a life-like setting. Additionally, SimSE is fully graphical—all of the typical surroundings of a real-world software engineering situation are realistically portrayed.

2. APPROACH

SimSE consists of three main components: the simulation model that encapsulates a set of cause and effect rules and is used to drive the simulation with particular scenarios, the graphical user interface (GUI) through which the user interacts with the simulation and views the visual effects of the simulation, and a simulation engine that executes the model.

A simulation model consists of a set of mathematical and logical relationships that, collectively, represent the rules underlying the behavior of the real-world process it embodies. Because their purpose is to represent the concepts being taught, the models used in driving a simulation environment are key to any successful learning experience. We have designed a base set of models, each with a different size and scope, to be used in driving the simulation environment. In parallel, we are developing a modeling language to allow educators using SimSE to build custom models teaching the particular software engineering issues or processes they wish to highlight. The language provides facilities to express such features as the goal to be achieved by the user, the setting in which the simulation takes place, the underlying rules of the simulated process, the visual effects to be displayed, and the available user commands.

A fundamental feature of our simulation environment is that it will be graphical—all of the major artifacts, people, tools, and surroundings typically present in a real-world software engineering situation are realistically portrayed using a tile-based, top-down view of the simulated organization. Learning through visual clues has proven to be far more effective than simply studying textual output, and SimSE takes full advantage of that fact—consequences of the student's decisions are visually illustrated through character actions, speech (in the form of pop-up "bubbles"), and explicit queries of artifacts and individuals for their status and progress. Through these features, the complex cause and effect dynamics of the software engineering process can be effectively taught in an entertaining, lifelike setting.

3. SUMMARY

We have identified and sketched a new approach to teaching the software engineering process. Our belief is that a graphical, interactive software engineering simulation environment can effectively improve students' learning of the software engineering process by providing a complete, realistic, and practical experience of the concepts taught in lectures without any of the drawbacks involved in a typical class project. We are currently in the process of realizing this vision by implementing a prototype environment. Once completed, the effectiveness of this tool will be evaluated by testing it in an undergraduate software engineering class.

[1] Emily Oh, Institute for Software Research, University of California, Irvine, Irvine, CA 92697-3425, USA, emilyo@ics.uci.edu
[2] André van der Hoek, Institute for Software Research, University of California, Irvine, Irvine, CA 92697-3425, USA, andre@ics.uci.edu

0-7803-7444-4/02/$17.00 © 2002 IEEE **November 6 - 9, 2002, Boston, MA**
32nd ASEE/IEEE Frontiers in Education Conference