Teaching by Modeling instead of by Models

Thomas Birkhoelzer, Emily Oh Navarro, André van der Hoek

***Abstract*— Teaching and training is one of the important applications of software engineering process simulation. Up until this point, however, it has only been used in the context of students running simulations of process models that were built by someone else.**

In this paper, we suggest a different approach: to use the modeling activity for teaching as well, rather than the simulation activity only. In particular, we pro-pose to assign students the task of building a new soft-ware process simulation model using an existing educational software process simulation environment, SimSE.

First experiences from a feasibility project are reported.

***Index Terms*—Process modeling, software project simulation, teaching of software engineering.**

I. Introduction

TRAINING and teaching is an important application of software process simulation [1]. Various models and environments have been developed targeting this context, e.g. [3], [4], [5], [6], [7] [8]. These all share the purpose of giving students virtual experiences of realistic software processes that would otherwise be infeasible to practice in an academic environment.

So far, the reported usage of simulation and modeling in this context is always structurally similar: An existing model is used by the trainee for virtual experiences, i.e. simulation is leveraged for teaching, and modeling is done outside the learning situation by an instructor or some other expert beforehand.

From other technical fields, however, it is well known that the modeling by itself provides valuable learning insights. For example, many classes in engineering disciplines include the development of a simulation model in the respective field of application as a final assignment.

In this paper, it is suggested to use software process modeling and simulation in a similar way: the active development of the model as a task for students, not just their passive usage of a pre-existing simulation.

Experiences from a first project are reported and needs for further work and developments are discussed.

Manuscript received February 6, 2005. The SimSE project is partially funded by the National Science Foundation under grant numbers DUE-0341280, CCR-0093489 and IIS-0205724.

T. Birkhoelzer is with the Department of Electrical Engineering and Information Technology, University of Applied Science, Konstanz, Germany (phone: +49 7531 206239; e-mail: birkhoelzer@fh-konstanz.de).

E. Navarro is with the Department of Informatics, Donald Bren School of Information and Computer Science, University of California, Irvine (e-mail: emilyo@ics.uci.edu).

A. v. d. Hoek is with the Department of Informatics, Donald Bren School of Information and Computer Science, University of California, Irvine (e-mail: andre@ics.uci.edu).

II. Didactic goals of modeling

The development of the model by students has three unique didactic advantages:

- *Modeling requires articulateness and explicitness.*
  For a simulation model, all assumed relations and mechanisms of projects or processes must be made explicit and precise in order to be executable. For example, it is one thing to just state that a tool would "improve the process". A simulation model, on the other hand, requires one to articulate this assumption explicitly: which attributes (e.g. error rate or productivity) are influenced and how?
- *Enactment of a simulation provides immediate feedback.*
  Enacting a simulation usually provides immediate and obvious feedback about the consequences of relations and mechanisms stated, much better than any instructor critiques, especially with regard to errors or neglected side-effects. This is not to say that the simulation can replace an instructor. The simulation just provides the mechanical feedback, such that instructors can concentrate on translating this into lessons learned.
- *Creative task as motivation.*
  Technical students – engineers as well as computer scientists – usually love to create things, not just to use them. In this sense, model creation can provide a much higher motivation than just model usage.

III. Modeling a system engineering project

A. Background

As a feasibility study of developing a simulation model by students as part of normal project management course work, three undergraduate students of "project engineering" at the University of Applied Science, Konstanz developed a simulation model as a project assignment. They were in the third year of their studies with a background in electrical engineering and project management, not in software engineering or computer science.

SimSE was chosen as the modeling tool and simulation environment, mainly because of its integrated model builder tool and graphical simulation environment.

Together with the SimSE tools, an existing SimSE model of a software project following a waterfall process model was available, which was used as example and template.

B. SimSE Environment

SimSE is a game-based, graphical, interactive software process modeling and simulation environment designed specifically for generating educational simulations. Its purpose is to allow students to practice quasi-realistic, large-scale software processes in a fun (and hence, more

educationally effective [2]) setting; and to allow instructors to build the simulation models for their students to "play."

One of the most significant features of SimSE is its model builder tool, which was designed to make model building simpler by obliterating the need to learn and program in a process modeling language. The model builder tool completely hides the underlying textual modeling language from the modeler by providing a graphical user interface. This interface allows one to build a model using only buttons, drop-down lists, menus, and text boxes – no programming is required. Once a modeler specifies all of the object types, start state objects, actions, rules, and graphics for a model, the environment then generates a simulation game based on that model.

The graphical, high-level nature of the model input requires less initial learning overhead compared to a special modeling or programming language, which was an important issue considering the students' non-computer science background.

Moreover, the model builder, as well as the generated simulation game, are Java applications. Therefore, only a Java development kit is required as a prerequisite, which eases the usage by students on their private hardware.

More information about SimSE and its waterfall model that was extended for this project can be found at [9].

C. Modeling task

As modeling task, a system engineering project was chosen, i.e. a project combining hardware and software parts into a system. Because SimSE is a game-based simulation environment, it requires that a scenario or "story" accompany each model. In this case, the simulation's goal was the development of a control component for a test robot. The idea of this story stemmed from work done by one of the students during an internship. Thus, this immediately connected the theoretical model to practical experiences.

The students built the model as an extended waterfall model that incorporates hardware aspects as well as software, using the following basic flow: Starting from system requirements, hardware, software, and supply components are pursued in separate paths (with the associated artifacts) and finally integrated into the product.

Therefore, the modeling task was similar to the existing waterfall simulation model. Theoretically, the artifacts, activities, and relations from this existing model just needed to be cloned and extended. At the same time however, to avoid an overly complex model, the existing mechanisms needed to be simplified. Both, appropriate extension and simplification, required a thorough understanding of the model and, more importantly, of the intentions behind the model.

D. Modeling Workflow

The SimSE model builder tool supports a modeling workflow, which closely resembles a didactic decomposition of project management issues.

1. *Definition of the project constituents (artifacts, deliverables, participants, tools).*
   This is the first and basic step of any project management. In the SimSE model builder, this corresponds to defining the object types (templates for the simulation objects) and start state (instantiated objects that the simulation begins with). For the system engineering project, business, software and hardware artifacts were defined reflecting the basic steps of a development process in each category (e.g. specification, design implementation, test). Participants are employees with different experience in these three fields (software, hardware, business).
2. *Definition of the actions a manager (player) can take.*
   These are the inputs into the simulation. There are typically two classes of such actions: task assignments (create, review, correct) and management actions (give bonus, purchase tool, motivate by free coffee, fire). Whereas the first ones are direct consequences of the list of artifacts, the second class enables and enforces the students to define their management style. Each manager/player action corresponds directly to a SimSE action – the specific ways that a player can manage, control, and drive the simulation.
3. *Definition of the actions that occur autonomously.*
   In addition to the actions triggered by the manager, there are events beyond the control of a manager. In SimSE, this is modeled by actions that occur automatically (e.g., employees take breaks) or randomly (e.g., the customer introduces new requirements, employees get sick). The definition of such events teaches basic risk management.
4. *Definition of the effects the actions should have.*
   In SimSE rules are attached to each action. These rules specify how that action affects the rest of the simulation. For example, a creation rule affects the completion percentage of an artifact depending on the productivity of the participant.
   The rules and actions can be prioritized according to which ones should be evaluated first, based on their dependencies. For instance, an action that is triggered based on an employee's energy level (e.g., take a break) should be evaluated after another rule that modifies that employee's energy level is fired.
   Whereas the definition of such rules seems to be straightforward on a first glance, actually conducting it reveals two basic challenges: The large amount of such effects and the difficulty to quantitatively describe it by mathematical formulas. For example, every project manger would immediately agree that the productivity of an employee depends on the experience, the mood, and the number of parallel tasks, but how to model this by a mathematical formula, e.g. as a sum or a product?
5. *Definition of the dependencies.*
   Dependencies between artifacts and activities are modeled in SimSE by effects of rules as well. For example, a creation rule can have the effect of reducing the completion percentage of a dependent artifact (or the effect of increasing the number errors in this artifact).
   Whereas this provides a realistic management situation (a dependent artifact can be worked on before finishing its precursor), it adds another level of complexity. The student project stopped at this point.

6. *Definition of the graphics (in parallel to the previous steps).*
   To each constituent of the project, an image needs to be assigned for the simulation. In addition, a pictorial layout of the simulated office must be defined. Whereas this provides no direct insight into project management, it adds a lot to the impression of ownership.

IV. Lessons and Recommendations

During the project, the following lessons were learned:

- *Modeling is difficult but possible*
  The modeling task provides a challenge to students. The translation of project management knowledge into mathematical formulas and mechanisms is unusual and unfamiliar. Nevertheless, the student group finally succeeded: despite some odds, they were able to form a first simulation model. Of course this model is not yet complete (specifically, further effects could be modeled and many dependencies are missing).
- *Creative aspects provide high motivation*
  The opportunity to create their own processes and mechanisms served as an important motivation for the students. In this context, also less scientific aspects like graphics and playful aspects should not be neglected. It seems to be more fun to design a game-like simulation than just chart outputs.
- *Examples are helpful and necessary*
  For the system engineering project, the most successful modus operandi was the study of the artifacts and mechanisms of the existing model and their appropriate modification. Designing everything from scratch would have been much more difficult. Based on that experience, it is recommended to document and use such examples as kind of templates or patterns.
- *Tools need improvements*
  Within the traditional usage of simulation models in a teaching context, the students use only the simulation environment, not the modeling tool. Therefore, the simulation environments are designed for non-expert use, whereas the usability of the modeling tool got less attention. Instead, the modeling tools were designed to allow maximal expressiveness and flexibility.
  Modeling by students requires modeling tools for non-expert users as well. Usability with minimal training might be more important than complicated functionality.
- *Implementation-independent notational support missing*
  One of the most difficult tasks was the specification of the intended relations and mechanism on paper. For example, the modeling tool provides interactive menus to specify the modeling rules. However, to design, discuss, and document these rules outside the program on paper, an appropriate notation (not a programming language) is necessary. For process modeling, such a notation is not readily available or broadly established. It is difficult for students to develop such a notation by themselves. Therefore, early support by the instructors on formal model design (not just implementation) is necessary. It might even be helpful, if the modeling tools would be accompanied by notational tutorials and examples as well.

V. Conclusion

Modeling by themselves forces the students to be precise and explicit about their assumptions regarding projects and processes, as the enactment by the simulation provides immediate feedback. Moreover, the creative nature of the task seems to meet the motivation of many students in this area.

This is not to say that such modeling should replace the use of preexisting models in a learning context. Both have their distinct advantages: Own modeling is restricted to small problems with a limited level of detail and sophistication and might even remain incomplete. It is not expected that such models can be used to gain really new insights from the model itself. To gain such insights requires using and studying more elaborate and detailed models developed by experts. Using the later one as templates for the own development as described in Chapter 4 can benefit both approaches by improving the own results as well as the appreciation of the “expert-models”.

Therefore, it is expected that the development of simulation models by students themselves can be a valuable component of process and project teaching and training complementing the use of expert-models. The overall positive outcome of the system engineering project modeling supports this expectation.

To alleviate this, however, further work is necessary especially towards developing easy-to-use modeling tools, appropriate templates and modeling tasks, and unified and simplified notations.

Acknowledgment

The system engineering project has been developed by Patrick Schnell, Markus Schulz, and Stefan Weidele. We would like to thank them for their intellectual curiosity to try such a task and for their dedication to carrying it through.

References

[1] M. L. Kellner, R. J. Madachy, and D. M. Raffo, “Software Process Modeling and Simulation: Why? What? How”? *Journal of Systems and Software 46*, Elsevier, New York, 1999, pp. 91-105.

[2] M. Ferrari, R. Taylor, and K. VanLehn, “Adapting Work Simulations for Schools,” *The Journal of Educational Computing Research, 21(1)*, 1999, pp. 25-53.

[3] A. Drappa and J. Ludewig, “Simulation in Software Engineering Training,” *Proceedings of the 22nd International Conference on Software Engineering*, ICSE, Limerick, Ireland, June 2000, pp. 199-208.

[4] J.S. Collofello, “University/Industry Collaboration in Developing a Simulation Based Project Management Training Course,” *Proceedings of the Thirteenth Conference on Software Engineering Education and Training*, S. Mengel and P.J. Knoke, Eds.: IEEE Computer Society, 2000, pp. 161-168.

[5] H. Sharp and P. Hall, “An Interactive Multimedia Software House Simulation for Postgraduate Software Engineers,” *Proceedings of the 22nd International Conference on Software Engineering*, ACM, 2000, pp. 688-691.

[6] D. Pfahl, M. Klemm, and G. Ruhe, “A CBT Module with Integrated Simulation Component for Software Project Management Education and Training,” *Journal of Systems and Software 59*, Elsevier, New York, 2001, pp. 283 298.

[7] E. Oh Navarro and A. van der Hoek, “Software Process Modeling for an Interactive, Graphical, Educational Software Engineering Simulation Game,” *Proceedings of the 5th International Workshop on Software Process Simulation and Modelling (ProSim 2004)*, Edinburgh, May 2004, S. 171-176.

[8] Th. Birkhölzer, L. Dantas, C. Dickmann, J. Vaupel, „Interactive Simulation of Software Producing Organization's Operations based on Concepts of CMMI and Balanced Scorecards,“ *Proceedings of the 5th International Workshop on Software Process Simulation and Modelling (ProSim 2004)*, Edinburgh, May 2004, S. 123-132.

[9] E.O. Navarro and A. van der Hoek, “Design and Evaluation of an Educational Software Process Simulation Environment and Associated Model,” *Proceedings of the Eighteenth Conference on Software Engineering Education and Training*, Ottawa, Canada, IEEE, 2005 (to appear).