Teaching Software Project Management using Simulations

Andreas Bollin[1], Elke Hochmüller[2], Roland T. Mittermeir[1]
[1)] *Universität Klagenfurt* and [2)] *Carinthia University of Applied Sciences*
{andi, roland}@isys.uni-klu.ac.at and *E.Hochmueller@cuas.at*

Abstract

An experience-dominated subject like software project management cannot be learned by merely attending lectures. Additional labs, however, even with only modest real-life projects, call for substantial effort to be spent by the instructors as well as by the partaking students. Our experience shows that using a software development simulation tool enhances the mix of methods used in conventional teaching substantially.

1. Introduction

Software Engineering (SE) education has to account for various facets of activities and roles software engineers will meet throughout their professional life. They extend far beyond programming skills and computer science knowledge [9]. They also include social skills, economic planning and responsibility [5], and even ethical concerns and evaluation [10]. This has to have consequences for teaching. It calls for partly replacing lecture style with project work, both, intra-, as well as extramurally. But while project focussed teaching has been widely recommended not only for SE-classes, it suffers from the time it takes to complete a project and from the rather eclectic selection of topics addressed within the context of a given project. Hence, sceptical colleagues often raise the issue of effectiveness versus efficiency.

This paper focuses on simulation as a compromise between both contestants. The authors' experience of supporting software process education by using a simulation system dates back to 2002. This paper reports on the lessons learned and discusses the pros and cons of using simulations to provide experience in software project management (PM) that would otherwise be obtained only by real projects.

The next section addresses aims and obstacles to be met in software project management education. Then, the simulation system we are using is explained. Section 4 reports experience gained within the organisations we are affiliated with as well as results from applying the system extramurally and in other educational institutions. Section 5, comparing the simulation approach with conventional project based learning, wraps up the paper.

2. Aims and Obstacles in Software Project Management Education

Software engineering education aims at training skills in the development and assessment of software products at different stages. Teaching various kinds of development methods (analysis, design, and programming techniques and languages) and quality assurance approaches (reviewing and testing strategies) is usually carried out by means of lectures and corresponding labs. Topics of such labs are initially small, disconnected, oversimplified examples to be solved by individuals. They progress to software development projects requiring a group of students to complete them. Such projects require already some attention to managerial issues and to practice social competences within the team. But the common objective of all these exercises is to train technical skills essential for future software developers working as team members.

978-1-4577-0347-8/11/$26.00 ©

Training students to become leaders of software development teams requires different knowledge and specifically managerial skills. Theoretical knowledge of software development processes can be transferred by lectures. But how can software management skills (e.g., planning, budgeting, staffing, steering) be imparted on students? Lectures on the subject will be considered trivial or exaggerated. Key issues have to be experienced. Ordinary programming labs with simple examples lack the necessary complexity. Student projects lasting at least a critical period of time are needed. Certainly, experiencing such projects will increase the students' practical knowledge in their software development and assessment skills taught beforehand. But the huge effort needed for construction and evaluation tasks will likely overshadow the students' focus on applying a particular software development process and noticing the failures due to not conducting managerial tasks properly.

In case of small programming examples, students may try and compare different solution strategies. Because of time and effort, this approach is not feasible for comparing different PM strategies. Furthermore, to what degree is any experience gained from student projects carried out within university settings comparable to that of a real-life project in industry?

In order to overcome these obstacles, this paper proposes using a simulation environment as efficient and effective means to enhance education in project management skills. With simulation we strive for achieving the following aims in PM education:

a) Experimenting with different approaches. Trainees shall be able to experiment with different project management strategies in order to experience the respective effects of their decisions. This shall be possible by simply carrying out several simulation runs as a whole. An additional option would be the repetition and respective experimentation of just some parts of a simulation run.

b) Supervisory elements. The simulation model should provide active and/or passive tutoring agents in order to supervise and guide the project managers in fulfilling their tasks and in decision-making. An active tutoring component could observe the project managers' actions and contact the student-manager in case of crucial situations or wrong decisions. A passive tutoring component could be asked by the trainee for situation-specific expert advice.

c) Success evaluation. The project managers should get useful feedback about their performance on the basis of project specific evaluation criteria. This feedback should contain enough information to allow cause effect analysis of decisions made during a simulation run.

d) Reality conformance. The trainees should be confronted with situations close to reality. This also means that their decisions should lead to effects which are comparable to those in real-life projects. A prerequisite for close-to-reality simulations is the availability of respective empirical data from real projects.

With these main objectives in mind, we developed the concept for AMEISE, a simulation framework for practicing management of software engineering projects.

3. The AMEISE Framework

AMEISE is a Client/Server system using a simulation engine called SESAM, Software Engineering Simulated by Animated Models. SESAM has been developed at the University of Stuttgart under the direction of Jochen Ludewig [2]. The AMEISE team adopted the educational model proposed by the SESAM-team and used their system as initial prototype, evolving it in a series of iterations to the currently available AMEISE system [7, 8]. Helper components built around the simulation core and the fact that key data of every simulation step is stored in a MYSQL database for later assessment and visualization constitute the main difference between SESAM and AMEISE.

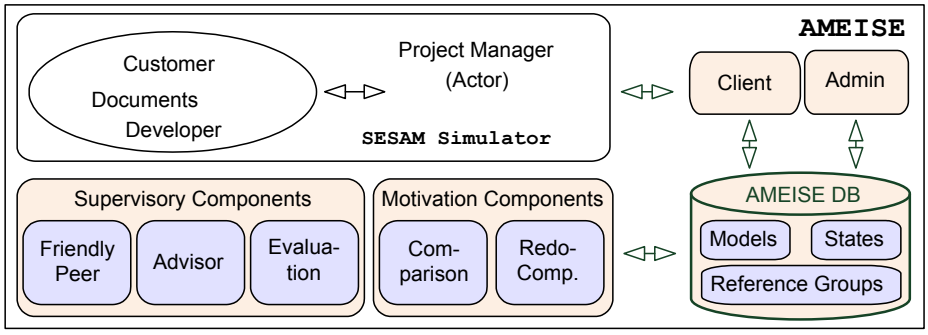

Figure 1: AMEISE is built around the simulation environment SESAM and extends it by a set of features and supervisory/motivation components for both, the trainers (e.g. via Administration, Monitoring or Evaluation tools) and the trainees (e.g. via the Friendly Peer or Advisor agents).

3.1. AMEISE Components

AMEISE has been designed for blended learning situations. Hence, one major difference to SESAM is immediate feedback after completing a simulation run, as well as feedback and help during simulation runs. Fig. 1 presents an overview of AMEISE extensions. Besides the *Client* (a SWING based Interface to the simulation core), it provides an *administration* environment for administering courses and users, three supervisory and two motivation components. The core of AMEISE is the *MYSQL database*. It contains the simulation settings (called *models*), the *states* of active simulations and the full trace of historic simulation runs (*reference groups*). There exist models of different complexity. In our courses we used the model QA 200. It focuses on quality aspects, requiring the trainee to manage a 200 AFP project within 9 months and a budget of 225.000 €.

The following supervisory components are available (and can be activated or deactivated by the trainer, depending on his or her didactical strategies):

Friendly Peer. It runs as an agent in the background. Whenever it detects problematic decisions during the simulation run it becomes active notifying the trainee about the upcoming problem (comparable to a friendly experienced colleague sitting next to you in the office).

Advisor. It is a component comparable to an experienced colleague next door. Whenever a trainee needs advice he may step by asking questions (to be selected from a predefined list).

Evaluation. This component can be used to generate diagrams and tables visualizing key data from the simulation run. It can also be used by the trainers to generate assessment reports and ODP files providing an overview of simulation runs.

For motivational reasons, the system also provides two Client extensions. The *Redo-Component* is a rollback mechanism enabling the trainee to jump back to a milestone in order to try a different strategy. The *Comparison Component* allows the trainee to compare his own progress with the progress and steps of simulation runs stored in the database. However, after some experimentation we decided to hide these extensions in standard situations. They might give AMEISE the touch of a computer game, something not intended in our classes. They are helpful though in special explorative situations.

3.2. Didactical Concept

The AMEISE educational model is based on experimental learning. This implies that trainees should be able to learn from their own failures. Therefore, we strongly recommend trainees to perform at least two simulation runs. While trainees should strive to achieve their best

possible performance in both runs, they usually will be faced with some challenges during the first simulation which need to be discussed in order to identify improvement potential for the next simulation. Benefitting from experience gained during the very first simulation, usually trainees are able to improve their performance in subsequent simulation runs.

Basically, the AMEISE simulation environment is used in the context of general software engineering or management courses in order to experiment with project management skills. Before carrying out AMEISE simulations, theoretical knowledge on PM has to be delivered by respective lectures. The role of a project manager in AMEISE simulations will usually be resumed by a single trainee or a group of two persons. The latter is preferable because it stipulates discussion and reflection of decisions.

Instructors can decide which supportive components (friendly peer, advisor) will be accessible during simulation runs. We also recommend a standard educational setting for AMEISE simulations consisting of the following sequential phases including at least two simulation runs (the actual amount of simulation runs is up to the instructor):

- ***Preparation.*** The particular simulation model to be used for the simulation runs should be introduced some days ahead of the first run. The contents and duration of this introduction may vary depending on the extent the relevant project management aspects were already dealt with in previous lessons. However, trainees should at least be provided with an AMEISE hand-out including useful hints and a list of the virtual software developers indicating their hourly rates of wages and specific qualifications in various software engineering activities. In addition to relevant model-related knowledge, the trainees should get sufficient information about the AMEISE environment in order to use it properly.
- ***Planning the first simulation.*** Having received the introductory information, trainees (single persons or groups of two) are requested to prepare a schedule and the allocation of virtual software engineers to phases and activities within their simulated project. In case of lacking or insufficient preparation, planning will have to be carried out at the beginning of the first simulation session.
- ***First simulation.*** We recommend the first simulation to take place in a classroom with an instructor present. Thus, any ad-hoc questions arising during the first simulation run can be answered immediately. During the simulations students can hire (and fire) employees (with given salary and a spectrum of qualifications). Employees can be assigned tasks such as talking to the customer, designing, coding, participating in review teams, testing or integrating. Assignments may be cancelled instantly or after completion of the actual subtask.
- ***Feedback after the first simulation.*** Feedback concerns expended time and budget, degree of completion and number of errors in code and documentation. It is broken down for particular phases. Diagrams show task assignments over time. Quality growth indicators are also given. The following assessment options can be used in combination.
  - Online assessment in presence of the instructor (recommended for immediate feedback on the first simulation run),
  - Individual online assessment in the absence of the instructor for further simulation runs.
  - Delivery of a detailed evaluation report generated for each AMEISE simulation.
  - Instructor's feedback in a plenary session with discussion and comparison of interesting results.

  Experience showed that a plenary session after the first simulation helps not only to compare the different results in the class but also to show the diverse effects achieved due to different decisions taken by trainees on certain aspects of the simulated project.
- ***Planning the second simulation.*** For didactical reasons, a second simulation run is highly advisable. With a second simulation, trainees will get the chance to demonstrate their ability to improve their simulation results. Because of the challenges faced with during the first

Table 1: List of AMEISE simulation runs and number of respondents of the online survey. The table also provides the name of the institution, country, and year of its first use.

Institution	Country	In Use since	No. Courses	No. Students	No. Respondents
Alps-Adriatic University Klagenfurt	Austria	2002	15	437	58
Carinthia University of Applied Sciences	Austria	2002	16	233	22
Johannes Kepler University Linz	Austria	2003	2	12	0
University of Applied Sciences Kufstein	Austria	2005	3	76	59
Ecole Nationale d'Ingénieurs de Tunis	Tunisia	2004	3	20	0
RWTH Aachen	Germany	2005	1	11	0
University of Heidelberg	Germany	2006	1	15	0
University of Applied Sciences Emden	Germany	2007	1	11	0
University of Kosice	Slovakia	2008	3	177	47
Total			**45**	**992**	**186**

simulation and the motivation to achieve better results during the subsequent attempt, trainees will usually spend more effort in planning the second simulation run.

- ***Second simulation.*** Familiarity with the AMEISE environment is gained already during the first simulation. Hence, the presence of an instructor is not necessary for further simulation runs. This allows trainees to decide individually on when to start their second run. They may suspend a simulation run and resume it later.
- ***Feedback on the second simulation.*** While all feedback options described above exist, trainees should be able to interpret the results provided by the online assessment feature already themselves. In addition to discuss the trainees' performance of the second simulation, a plenary session should also include analysis and discussion of deviations between the first and the second simulation runs.

4. AMEISE Experience

Over the last eight years, AMEISE simulation runs were held in support of several courses on SE or PM. AMEISE simulations were also done at other academic institutions. Table 1 presents an overview of the institutions using AMEISE and also the number of courses and trainees who used the simulation environment.

During these years a lot of experience was gained, both, in classroom situations and in blended-learning settings. This experience influenced the development of new features and led to a refinement of the didactical concept. This section summarizes this experience and reports on effects observed during the various AMEISE courses.

4.1. Assessment

AMEISE started in the year 2002, supported by the Austrian Ministry of Science and Education within their eLearning initiative „New Media in Tertiary Education“. The objective was to improve education in PM. We started to systematically collect feedback from trainers and trainees from the beginning of the project by means of standardized questionnaires, interviews, and group discussions. Since summer 2008 the questionnaire for trainees is in electronic form[1]. Feedback data is collected via Internet and stored in a MYSQL database.

Taking part in the survey is voluntary. During the last three years 186 questionnaires have been collected and analysed. The majority of the data (76%) stems from simulation runs conducted in classroom situations, 24% of them ran unattended from at home. In general, we may report that students were satisfied with the insights they gained from

[1] The online survey is conducted via LimeSurvey (see http://www.limesurvey.org).

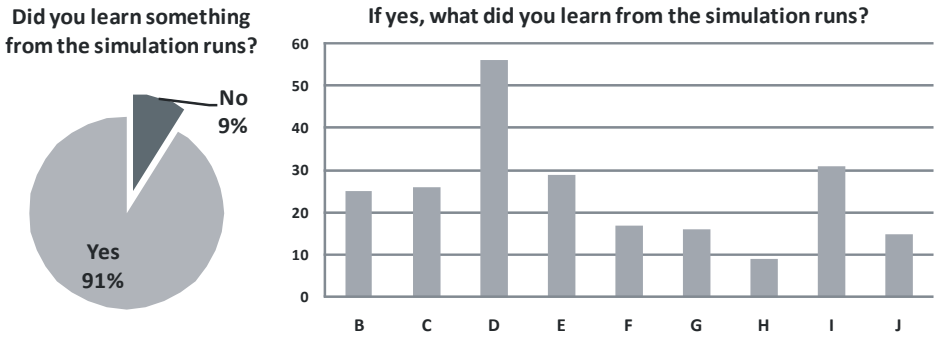

Figure 2: *Left*: 91% (136 out of 150 respondents for this question) reported they had learned something during the AMEISE simulation run. *Right:* Number of mentions (136 respondents, multiple answers permitted) concerning the questions what has been learned from the simulation runs: B … Use of resources, C … Rules for staff assignment, D … Importance of project management, E … Importance of quality assurance, F … Typical tasks of a project manager, G … Complexity of project management, H … Importance of communication, I … Role of SW Development lifecycle, J … Other.

AMEISE, as 91% of them claim that they have learned something from the simulation runs (see Figure 2, left side).

Trainees have also been asked by open ended questions what they have learned from simulation runs. The free-text answers have been categorized. The results are presented in Figure 2 (right side). The majority of the answers mention the understanding of how important project management is (41% out of 136 responses), followed by the awareness that "[…] now I understand what a software development lifecycle is all about" (22%). Approximately 19% of the trainees state that they learned, for the first time how to plan and use resources and how to do personnel management. Something recurrently noticed by trainers listening to trainees during their work is substantiated by the answers: the awareness that project management is really complex and needs a lot of effort (12%). Especially this observation led students to take managerial issues serious and to improve preparation for their second (or even third) simulation run.

In order to achieve conformance to reality, the AMEISE environment should enable trainees to gain experience from real situations and help them to identify with the role of a project manager. Last but not least the simulations provide additional motivation to stay concentrated on the topic.

Figure 3 shows some results referring to these goals. Over two thirds of the 153 respondents for this question had fun during the simulation runs and thus were eager to repeat the simulations (and to learn something new). 60% of 150 respondents think that AMEISE is directly related to reality and of practical use. Also important for our educational aim is the question whether trainees identified themselves with the role of a project manager. Here, nearly 74% of 130 respondents answered this question positively.

Trainees have also been asked to tell us more about impediments and possible improvements. Their free-text answers have been categorized and summarized in Figure 3 (bottom right). 37% of 136 respondents think that an improved GUI could raise the level of satisfaction, especially when it is more colourful or provides a 3D model of the company. It is noteworthy to state that this opinion diminishes with the second simulation run. Most of the trainees then tell their trainers that, after a while, the situation gets so demanding that even a better user interface would not matter.

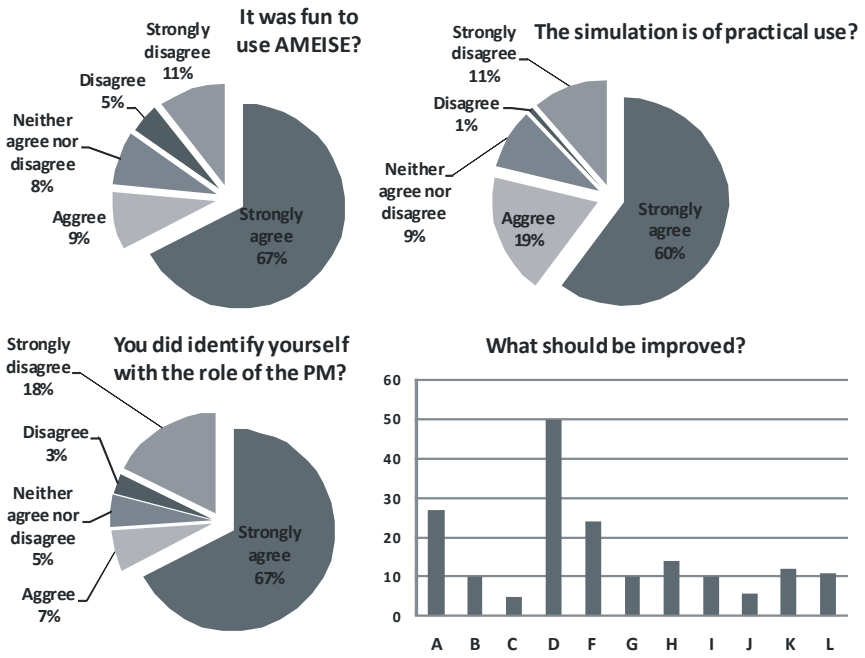

Figure 3: *Upper left*: Pie-chart summarizing the results of the question (n=153) whether playing AMEISE was fun or not. *Upper right*: Pie-chart summarizing the results of the question (n=150) whether the trainees believe that the simulation has some connection to reality. *Bottom left*: Pie-chart (n=130) visualizing how intensive a trainee had the feeling of being the project manager (PM). *Bottom right*: Trainees (n=136) think that the following issues should be improved in order to maximize the learning effect (multiple answers are permitted): A … Improve the return messages from the simulation core, B … Improve the introduction session/provide more materials for the process model, C … Provide more help during the classroom sessions, D … Improve the GUI/make it a more colorful 3D environment, F … Provide a status box with project key-data, G … Add more commands, especially for communicating with developers, H … Nothing, I … Improve the response time of the Client, J … Provide other, non-SW related, simulation models, K … Add half-time job possibilities, L … Create a game-like environment with auto-proceeds.

Second in the list of impediments is the opinion that improved return-messages from the simulation engine would be helpful (20 %). AMEISE has been developed by a team of dozens of people, basically non-Anglophone, so there is definitely the need for permanently checking the language component for consistency and for the terminology used in other SE courses. Introducing game-like utilities to the AMEISE-interface ranks third among the answers (18%). Especially a box displaying the status of developers and the progress of their work is highly favoured. “How should I know who is doing what and how long a task will take?” is a typical question. The development team of AMEISE had several discussions on this topic but decided not to implement such helper gadgets. They would weaken the strength of AMEISE, specifically its proximity to reality. Team members will not and cannot give detailed answers on this issue. A real manager has to form her/his opinion and make related notes on his or her own.

The remaining answers demonstrate that users of AMEISE appreciate the environment, but would prefer having it even closer to reality. The simulation model should be refined and also the set of commands should be extended (e.g. some trainees suggest commands like “tell [a developer] to speed up” in order to keep some milestones). Fine-granular models are available. But they are too complex to be executed within a couple of hours or

by inexperienced student-managers. Hence, they are not useful for classroom situations. Further, in reality the effect of a request like the one cited is highly situation-dependent.

Reporting detailed statistical analysis of responses would go beyond the scope of this paper. Nevertheless, feedback from the trainees confirms that they experience AMEISE as a valuable supplement to traditional PM lectures.

4.2. Off-site Experience

Our experience with AMEISE does not only stem from the questionnaires and feedback from trainers as mentioned above, but also from the operation in non-typical classroom situations and in environments that are not directly controlled by the developer teams.

Firstly, AMEISE can be applied in non-IT courses and with students not having any previous knowledge in software development. In these cases the introductory course is extremely important. The final results of those simulations are slightly below average. Especially when the terminology and the process are not explained carefully, satisfaction and subjective learning effects decrease by approximately 15%.

Secondly, the importance of a commonly understood terminology cannot be overstressed. As mentioned in [7], the success of a simulation run depends on the understanding of the tasks to be conducted. A slightly different use of terminology or a misinterpretation of commands (e.g., trainees confusing the semantics of reviewing and testing it) is reflected in both, lower simulation results, and lower satisfaction.

Finally, the trainees' preparation has severe influence on duration and results of the first simulation run. Without a project plan at hand, trainees need at least 1 ½ hours longer than trainees with a project plan prepared. The latter typically need 4 hours. Often, we refer students to an observation with professionals. Instead of starting directly to interact with the AMEISE client (as students tend to do), they spent over an hour developing a rigorous project plan, finishing the simulation run only 2 hours later. Still, their results are among the best ever achieved in the history of AMEISE simulation runs. The message is clear: preparation pays off.

5. Pros and Cons for Simulating Development Processes

One may be sceptical about using simulations for teaching such complex issues as software project management, involving technical, economic, and social issues. Definitely, they should not stand alone and students have to be made aware of the differences between *reality* and an *image of reality* as represented in a specific simulation model. But simulations allow students to delve into a world that would otherwise be inaccessible to them.

To assess the dangers and merits of simulations as part of training or education requires first to focus on the educational goals. Just mentioning SE is insufficient. Comparing the goals of Ludewig's [2] and our approach with that of Hood and Hood [4] shows this. While the former aims for highlighting issues project-managers have to be aware of, the latter are highlighting problems resulting from imprecise requirements. But while the topics are different, both share the problem that mere classroom-teaching leads to insufficient results. Software Engineering is engineering. Therefore, as alluded to by Hilburn [3], students have to be confronted with out-of-class reality. However, following his suggestion of three comprehensive SE-Labs, two of them in industry, seems too time consuming, especially if the educational institution cannot control the student's activities during summer. The approach followed at Klagenfurt University with a team project in the BA-programme and an industry internship as well as a simulation project for MA-students (usually in this order) seems to be a close compromise towards his suggestion.

Assessing this approach, the question is twofold. What is gained by requiring the project management simulation attached to an SE-course? What is lost by the simulation against a real industry internship?

The first question can be addressed by the answer of one student who, challenged by one of the co-authors after mentioning various issues he learned during the simulation, "Well, that's fine. But didn't you hear all this in my SE class you attended two years ago?" The answer was "Yes, you told us. But now I experienced it."

That is the critical issue. Telling about SE-issues, be they related to managerial issues in the proper sense or be they related to topics combining technical with managerial issues such as „Early focus on quality pays off later!" are taken as "theoretical" issues or as "book-wisdom". The students experience obtained from small lab-examples or even from simple work they did during vacations does not sufficiently conform to these messages. Even intra-university team projects lasting for a full semester are only weak proxies for real software projects among teams of professionals. The divergence from reality starts with the composition of the team (usually a set of friends or peers of similar qualification) and it ends with the pseudo-customer, a teacher being considerate of other obligations students have to fulfil. Finally, it culminates in the delivered products. Most term projects have no maintenance contract attached to them. Some projects deliver software to be integrated into a larger system (e.g., all those developing extensions to the AMEISE system). But even in this case, it is too often the teacher or some contracted project staff member who does part of final testing and even some additional repair if the effort already spent is beyond the requirements of a term project.

Student internships run under slightly different side-conditions. Therefore, they will come closer to the items mentioned by Shaw under the headline "Integrate an engineering point of view in undergraduate computer science and other information technology curricula" [9]. However, even in these internships, mentors are aware that a student-member of the group is "just a student" and therefore this member will be treated differently from fully paid fulltime employees. Whether ethical issues [10] will emerge in such a project or whether they are not better addressed in seminar-like courses might remain as point of debate.

Hence, apart of the efficiency issue, there remain issues causing a gap between reality and the image of reality drawn by student projects, even if the respective student is integrated in an industry project for a time span of 4 to 6 month (the duration of internships required from SE-master students in Klagenfurt).

But simulations are not free from drawbacks either. Simulations always rest upon a particular model. This model focuses on particular aspects of reality while ignoring others. Further, idiosyncrasies of the simulation tool influence the construct validity of simulation-based experiments. In case of SESAM/AMEISE due effort has been taken to be as close as possible to reality by using the effort data reported by Capers Jones [6]. This not only is an important aspect concerning validity. It also adds to the credibility of the results by students. The drawback is though that this data is dated and pertains to waterfall-like project models. Hence, AMEISE cannot be used for modelling agile processes or maintenance processes (in spite of efforts to extend it in these directions). This leads to justified criticism on part of the students.

Another critical issue is the modelling of software developers who can be hired to work on the simulated project. They have a name, certain qualifications of varying relevance for the project and they have a particular salary. These aspects suffice for the purposes of the simulation. But employees are just names, not persons. Thus, students quite often forget about having them hired, having given a task to them, or ignore that somebody is idle, just consuming salary. Of course, such effects show lack of planning. But they may also be

considered as artefacts created by the specifics of the user interface or of the virtual nature of these "persons" to whom their manager cannot enter any emotional relation.

Further, real projects don't only progress as planned. Situations like changing requirements, people becoming sick or leaving, etc. might emerge. While variants of the simulation model taking care of such effects exist, we do not use them, since such effects might easily overshadow our main goal of showing the importance of keeping track of the project's progress and of properly scheduling QA-activities. Random variations of performance parameters only adequately account for effects a project manager has to account for without being able to influence them.

On the other hand, the real-time needed by students for managing a 200 AFP project is for most of them between 4 to 5 hours. This allows redoing the project. It even allows further experimentation. Students can try different ideas and observe the related effects, another important aspect of SE education [1].

Nevertheless, instructors are always obliged to explicate very clearly during the first feedback round the difference between reality and simulation, encouraging the group to discuss how an effect occurring in the simulation would look like in a real project.

6. Conclusion

The AMEISE system enables SE students to experimentally exercise their software project management skills by means of simulation in a hassle-free manner without any grave consequences. Like in real-life projects, trainees have to base their decisions on incomplete and uncertain knowledge. But in real life, the opportunity to fairly analyze and duly criticize the project managers' decisions by a neutral and knowledgeable instance rarely arises. AMEISE provides such immediate and profound feedback based on its framework of rules supported by empirical data. Trainees can review the project's progress and reflect on how their decisions influenced the course and results of the project. Timely and plausible feedback enables the trainees to reflect on their actions, to learn about their consequences, and to improve their skills in a manner which can be even fun, entertaining and motivating.

References

[1] Victor R. Basili, Richard W. Selby, David H. Hutchens: *Experimentation in Software Engineering*; IEEE Trans. on Software Engineering, July 1986, Vol SE-12 (7), pp. 733 – 743

[2] Anke Drappa and Jochen Ludewig: *Simulation in Software Engineering Training*; in Proceedings, 23rd International Conference on Software Engineering, IEEE-CS and ACM, May 2001: 199 - 208.

[3] Thomas B. Hilburn: Software Engineering Education: A Modest Proposal; IEEE Software, Vol. 14 (6), Nov./Dec. 1997, pp. 44- 48.

[4] Dennis J. Hood and Cynthia S. Hood: Teaching Software Project Management Using Simulations; Proc. ITiCSE '06, ACM-SigCSE, 2006, pp. 289 – 293.

[5] Watts S. Humphrey: *Introduction to the Team Software Process™*; Addison Wesley, 2000.

[6] Capers Jones: *Programming Productivity*, Capers Jones, McGraw-Hill, 1986.

[7] R.T. Mittermeir, E. Hochmüller, A. Bollin, S. Jäger, M. Nusser: AMEISE – A Media Education Initiative for Software Engineering: Concepts, the Environment and Initial Experiences; in Auer (ed): Proceedings International Workshop ICL – Interactive Computer Aided Learning, Villach, Sept. 2003, ISBN 3-89958-029-X.

[8] R. T. Mittermeir: Facets of Software Evolution; in: Madhavji, N.H., Lehman, M.M., Ramil, J.F. and Perry, D.W.: Software Evolution, Wiley 2006.

[9] Mary Shaw: *Software Engineering Education*: A Roadmap; in A. Finkelstein (ed.): *Future of Software Engineering 2000*; ACM, 2000, pp 373 – 380.

[10] J. Barrie Thompson: Software Engineering Practice and Education: An International View; Proc. SEESE'08, ACM, 2008, pp. 95 – 102.