Experiences with Integrating Simulation into a Software Engineering Curriculum

Andreas Bollin[1], Elke Hochmüller[2], Roland Mittermeir[1], Ladislav Samuelis[3]
1) Universität Klagenfurt, 2) Carinthia University of Applied Sciences,
3) Technical University of Košice
Andresa.Bollin@aau.at, Roland.Mittermeir@aau.at,
E.Hochmueller@cuas.at, Ladislav.Samuelis@tuke.sk

Abstract

Software Engineering education must account for a broad spectrum of knowledge and skills software engineers will be required to apply throughout their professional life. Covering all the topics in depth within a university setting is infeasible due to curricular constraints as well as due to the inherent differences between educational institutions and the actual workplaces of individual graduates. This paper shows how a flexible simulation environment can link the various topic areas of software engineering in the same way they are interwoven in the daily work of practitioners. The authors report their experience gained in using such an environment in their courses at their different institutions, each one having a very distinct focus. Customization of the environment and respective didactical changes can address students with different maturity levels, educational aims, and backgrounds.

1. Introduction

Software engineering (SE) covers a broad spectrum of topics which go far beyond programming. SE knowledge and skills include not only product-related abilities and process-related strategies and responsibilities but also social skills and even ethical concerns. Existing software engineering curricula such as Software Engineering 2004 [1] account for this diversity of topics.

Universities usually design their curricula according to their particular needs, restrictions and environmental conditions including federal, legal, industrial needs (e.g. European Credit Transfer System governing student exchange based on effort points [2]). This leads to a huge variety of existing study programs with different focuses. Certainly, it is already a demanding challenge to properly design a curriculum with contents adequate for real-world needs. However, in accordance with the main topic of CSEE&T 2012, "Opportunities and Challenges of Software Engineering Education", in this paper, we want to focus on the opportunity to enrich software engineering classes with experimental learning based on simulation.

We have demonstrated how a simulation environment can improve the teaching of software project management [3, 4]. However, the proper use of such a simulation environment can also be a great catalyst for covering a wide variety of topics throughout different courses. Based on the SEEK curriculum [1] as a common reference, this paper demonstrates some of the opportunities to use simulation to support different SE courses with varying didactical purposes.

The next section describes an overview of related approaches to software engineering education. In section 3, we briefly summarize SEEK [1] which we use as a reference curriculum for our further considerations. Section 4 presents the AMEISE simulation framework we are using. It particularly addresses the assessment of simulation runs as well as the relevance of

978-0-7695-4693-3/12 $26.00 © 2012 IEEE
DOI 10.1109/CSEET.2012.18

AMEISE simulation models with respect to various SEEK knowledge areas. Finally, we discuss our experience with using AMEISE simulations in the context of different SE courses at the three institutions that we are affiliated with.

2. Related work

A considerable number of research papers deals with investigations towards software engineering education from many different points of view. An exhaustive analysis is out of the scope of this paper. This section provides an overview of selected literature and briefly characterizes their relationship to the ideas presented in this paper in terms of three key areas.

2.1 The role of academic software engineering and computer science education

The work of B. Meyer [5, 6] highlights the principles; it deals with the issue of what software professionals have to know in order to efficiently achieve the goals of the curricula. The same author focuses on teaching practices by proper identification and classification on such units as testable, reusable units of cognition, which serve for better understanding of SE topics [7]. The paper of A. Pyster et al. [8] surveys 28 software engineering curricula in order to establish a baseline for graduate education. P. Robillard [9] studies human behavior in software engineering that can identify ways to improve practices where creativity is involved. Positioning of software engineering activities within a holistic view of the software process helps us better define what is specific to software engineering and how to quantitatively and qualitatively measure these activities. Gregory W. Hislop [10] provides an overview of the effort necessary to be invested in SE education. He discusses the historical development, provides some perspective on current status, and identifies some of the challenges faced by software engineering educators.

2.2 Improving software practice through education

Timothy C. Lethbridge et al. [11] argue that the software engineering community could have a significant impact on the future of the discipline by focusing its effort on improving the education of software engineers. The paper identifies the following key challenges: 1) making programs attractive to appeal good students and to meet societal demand, 2) understanding the dimensions of the field in order to focus education appropriately, 3) communicating real-world industrial practices more effectively to students, 4) defining curriculum standards that are forward-looking, 5) educating existing practitioners, 6) making SE education evidence-based, 7) educating educators, and 8) raising the quality and prestige of educational research. The above mentioned ideas are analyzed from the research aspects and concrete research questions are formulated. Software education can be considerably improved by exemplary work on open source code topics. This is broadly treated in the literature, for example in the work of V. Rajlich's research group [12].

2.3 Software process simulation modeling

Software process simulation modeling addresses a variety of issues related to the software development processes. Applications devoted to software process simulations cover processes narrowed to portions of the software life cycle but also cover larger evolutionary models. The summarizing work of M.I. Kellner [13] provides an overview of the most relevant works till 1999. It deals with the questions of "why" (purpose of the simulation), "what" (the scope and the variables that can be usefully simulated), and "how" (modeling approaches and techniques that can be efficiently utilized) simulation modeling is applied. The comprehensive summary of definitions is supplemented by guidelines and recommendations for selecting an appropriate simulation approach for practical situations. Comprehensive reflections on the last 10 years of

CMP	Computing Essentials (172h): Computer Science Foundations, Construction Technologies and Tools Formal Construction Methods	VAV	Software Verification & Validation (42h): Reviews, Testing, Human-Computer Interface Testing and Evaluation, Problem Analysis
FND	Math. Engineering Fundamentals (89h): Discrete Mathematics, Statistics, Measurement and Economics	EVL	Software Evolution (10h): Processes and Activities
MAA	Modeling and Analysis (53h): Modeling Foundations, Specification and Validation of Requirements	PRO	Software Process (13h): Concepts, Implementation
PRF	Professional Practice (35h): Group Dynamics, Psychology, Specific Communication Skills, Professionalism and Ethics	QUA	Software Quality (16h): Quality Concepts, Culture, Standards and Processes
DES	Software Design (45h): Concepts, Strategies, Architecture and HCI Design	MGT	Software Management (19h): Planning, Personnel, Control and Configuration Management

Table 1: The SEEK knowledge areas of software engineering education [1, p.21] (including the suggested amount of time in hours for teaching the topics).

software simulation modeling are provided by the paper of He Zhang et al. [14]. Their work mentions the application of the SESAM tool which also served as basis for our own simulation environment. The work of E.S. Monsalve et al. [15] presents new trends in the application of game technology in SE education.

Finally, Bollin and Samuelis [16] condense experiences gained during the experimentation with a software project management simulation environment at the Technical University of Košice and its integration into the syllabi of its software engineering study program.

3. A reference software engineering curriculum

Various officially available syllabi contain the SE topics we are referring to, including the Software Engineering Body of Knowledge (SWEBOK) [17]. This guide classifies software engineering into a set of knowledge areas and outlines the generally accepted topics in those areas. The available material was developed in a series of iterations over the past 15 years and passed through a rigorous series of review processes. The resulting document became an international standard through ISO/IEC JTC/SC7 [18].

For in-class education and for a course of instructions the collection is too abstract, but parallel to the development of SWEBOK, the ACM and the IEEE-CS sponsored a project to define curricula for different kinds of graduate and undergraduate degree programs in computing [19]. As of 2011, there are five curriculum volumes covering the sub-disciplines “Computer Science”, “Computer Engineering”, “Information Systems”, “Information Technology”, and “Software Engineering”.

The sub-discipline “Software Engineering” (also abbreviated as SE2004) [1] is commonly referred to by its chapter title “Software Engineering Education Knowledge” (SEEK). It is structured into Knowledge Areas (KA) which are broken down into units and topics. Table 1 summarizes these areas and provides a brief description of their content. Each knowledge area (in the detailed curriculum also each knowledge unit) is given a number of hours which correspond to the actual in-class time required. Knowledge units are not necessarily disjunctive. E.g., the entry “MGT” highlights the discipline of software project management, but several topics related to management (for example group dynamics) can also be found in other areas on a different level of abstraction and with varying theoretical background.

In the next section, the SEEK framework will be used as a checklist for analyzing the various fields of application for the simulation environment discussed.

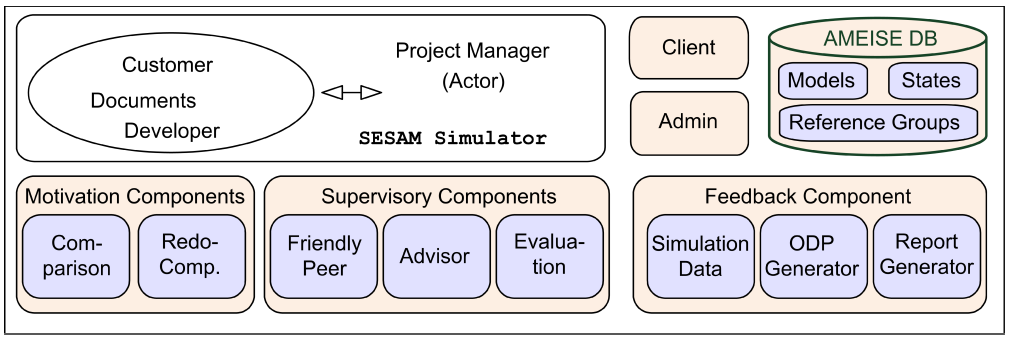


Figure 1: The AMEISE framework is built around the SESAM simulation core. It adds a lot of functionality by supportive motivational, supervisory, and feedback components.

4. The AMEISE framework

This section presents an overview of the AMEISE [20] simulation environment used for teaching software project management at all the three institutions the authors are affiliated with. AMEISE is a Client/Server system using a simulation engine called SESAM (short for Software Engineering Simulated by Animated Models) developed at the University of Stuttgart under the direction of Jochen Ludewig [21].

AMEISE has been designed for blended learning situations. A particular characteristic of AMEISE (and one of the major differences to SESAM) is that it heavily relies on helper components that are built around a simulation core [4, 22]. Additionally, key data of every simulation step is stored in a MYSQL database for later automatic assessment and visualization.

4.1 AMEISE components

Figure 1 presents the component structure built around the simulation engine inherited from Stuttgart. These parts have been added to SESAM during the AMEISE project. Basically, AMEISE enriches the simulation core from Stuttgart by supervisory, motivation and feedback components, which in turn, allow for guidance during simulation runs as well as for immediate feedback after completing a simulation run. Additionally, all relevant steps of a simulation run, the results of former runs, and different simulation models are stored in a MYSQL database, accessible to the components mentioned. AMEISE and its helper components have been described in more details elsewhere [3]. In the following we just give a brief overview of the supervisory and motivation components that are currently available and that can be activated or deactivated by the trainer, depending on his or her didactical strategies:

- The “Friendly Peer” runs as an agent in the background. Whenever problematic decisions during the simulation run are detected, the agent becomes active and notifies the trainee about the potentially upcoming problem.
- The “Advisor” is a passive component. In case of problems during the simulation run, trainees can activate this component by selecting questions out of a predefined list in order to obtain useful responses on how to proceed.
- The “Evaluation” component can be used to generate diagrams and tables visualizing key data from the simulation run.
- The “Comparison Component” enables a trainee to compare his or her own progress with the progress and steps of other simulation runs stored in the database.

- The "Redo-Component" implements a kind of rollback mechanism supporting experimentation with different management decisions. It enables a trainee to jump back to milestones in the project exploring different strategies.

The "Feedback Component" with its data visualization sub-component is crucial for our further considerations. Part of its functionality is shared with the "Evaluation" component, but it can also be used by trainers to generate assessment reports and ODP files providing an overview of simulation runs. AMEISE keeps track of every process step (like reviews or correction activities), and due to the database-driven architecture, interface data (concerning cost, effort/duration, and quality) can be obtained from tables and be played back into them. The type of data is dependent on the simulation model in use. But, already the quality assurance model we are using in our courses covers a wide spectrum of the whole development process.

- For the simulated project AMEISE stores project relevant data like begin and end date, available and spent resources, fixed expenses, degree of completion of the (intermediate) products as well as who is currently on the payroll of the project. It also stores the feedback of the (virtual) customer.
- For developers AMEISE stores their profile (skills, productivity …), their costs when on the payroll of the project, when and if they are working, and finally, the (parts) of document(s) they are or have been working on.
- For the different types of artifacts, AMEISE stores the degree of completion, the number of implemented AFPs, lost AFPs, the number and types of errors, who is and has been working on parts of it, and relationships between parts of the documents and the code.
- For activities the AMEISE environment stores the involved developers and documents, the exact begin and end date, and the actual status.

4.2 Example of using AMEISE simulation runs

The basis for an AMEISE simulation run is the so-called simulation model. It contains the different simulation settings. AMEISE supports models of different complexity and with different core areas (e.g. maintenance or quality assurance activities). In our courses (see also Table 2) we make use of a quality assurance model which focuses on quality aspects, requiring the trainee to manage a 200 AFP project within 9 months, a budget of 225.000 €, and strict requirements concerning the quality of the final product (in terms of a maximal number of errors allowed per 1000 lines of code and a minimal percentage of AFPs required to be implemented). To complete this task, trainees have to hire differently skilled developers from a set of available developers (available in the virtual company), assign tasks to them (for creating/reviewing/correcting documents or implementing/inspecting/ testing code) so that, finally, a tested software system including documentation can be delivered.

The standard educational setting for AMEISE simulations consists of the following sequential phases [3]: (1) an introductory lesson where the necessary background and theory are introduced or repeated, (2) planning for the first simulation run, where trainees (usually groups of two students) are requested to prepare for their first simulations, (3) execution of the first simulation run, (4) feedback-session, where trainees do get their personal assessment reports and an overview of other groups' best and worst practices, (5) planning for the second simulation run with the objective to improve the results of the previous one, (6) execution of the second simulation run, and finally, (7) feedback round for reflecting and interpreting the trainees performances. The feedback sessions (4) and (7) are highly relevant as most learning takes place there. To support these phases, AMEISE comes with a rich set of pre-defined tables and diagrams accessing this data. But, new tables and diagrams can be designed on the fly as

Group	Author	Reviewer(s)	Corrector	# Errors (1)	(2)	(3)	(4)
sep-03	Richard (+)	Bernd (+), ~~Stefanie (+)~~, Customer (+)	Richard (+)	115	67	61	55
		---------- Bernd (+), ~~Stefanie (+)~~, Customer (+)	Stefanie (-)		26	17	38
		---------- Bernd (+), Stefanie (+)	Christine (-)		0	0	**38**

Legend: (1) ... Errors contained in the document (2) ... Errors found during review, (3) ... Errors corrected, and (4) Errors remaining in the document

Reviewing skills: Bernd and Stefanie → high, Richard → low, Others → medium
Specifying skills: Richard → excellent, Diana → bad, Others → medium

(a)

Group	Spec (AFP)	Code (AFP)	Spec (PM)	Spec (€)	Tests (PM)	Tests (€)	Total (€)
sep-06	199.33	193.62	1.98	18,697.71	5.12	48,310.90	67,008.60
sep-07	198.99	195.96	1.72	16,616.24	8.04	77,795.42	94,411.66

(b)

sep-04	Spec	SD	MD	Code	Manual
Length (d)	13.00	18.00	28.00	29.00	18.00
Effort (h)	74.58	56.11	99.84	143.90	28.67
Correction (h)	16.42	39.22	17.64	14.46	16.34
Detected Errors	112.82	57.60	51.80	25.76	55.37
Errors left	17.00	41.00	69.00	57.00	92.00
# Reviewers	3.00	1.00	1.00	1.00	1.00

(c)

sep-04	Spec	SD	MD	Code	Manual	Total
Spec	42	-			-	42
SD	41	17	-	-	-	58
MD	51	16	49	-	-	116
Code	37	9	10	15	-	71
Manual	20	-	-	-	18	38

(d)

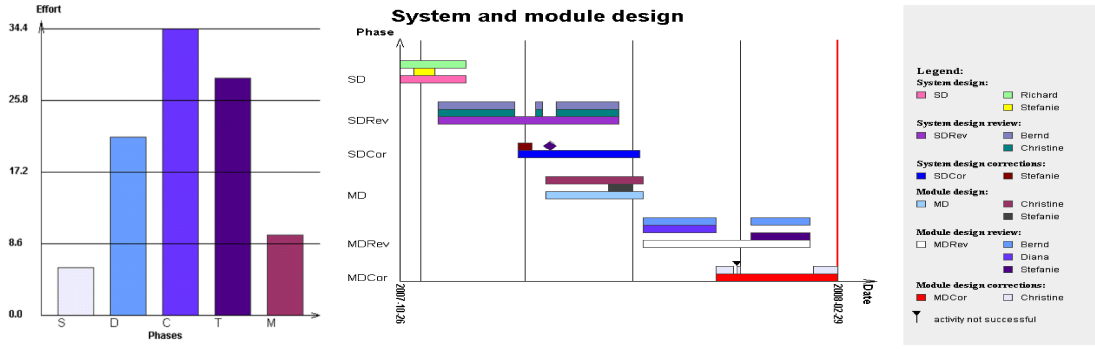
Figure 2: Sample screen shots of a table and diagram generated by the evaluation component for different simulation runs. (a) Diagram showing the strategy behind the review and correction process. (b) Table presenting the effort and needed resources (time in person months, costs in Euro) for different phases in the project. (c) Table summarizing the effort spent for creation, review and correction of different types of documents including the code. (d) Table summarizing the type and number of errors remaining in the various phase specific documents.

needed, and thus the type of feedback (and with that the focus of the course) can be controlled by the lecturer

To get a feeling for the variety of available data, Figures 2 and 3 present sample screen shots we are generating at the end of our simulation runs. At first we are focusing on *project quality*. Figure 2(a) shows the course of events during the review and correction phase of the "System Specification" document. There, the author of the document was a developer called "Richard" and the document contained 115 errors at the beginning. Then, three successive reviews of the document took place (with two reviewers and the customer), and AMEISE is able to report the exact number of errors found, corrected, and lost or introduced during these phases. Trainees experience the influence of different team compositions (size, skills). Another example of evaluative data that we are using is shown in Figure 2(c). It shows a table containing the time that was spent on reviewing and correcting the different types of documents including the detected and remaining number of errors, and Figure 2(d) expands the number of errors even to their types and origin. E.g. the 58 errors in the system design document include 17 pure system design errors and 41 system design errors resulting from 42 specification errors previously not corrected.

Besides data related to quality, we also deal with the *costs of activities*. Figure 2(b) presents a table that focuses on the quality (in reached AFPs) *and* time (in Person months) needed for the specification and test phase. It also provides the total cost (in Euro) for these two phases. And, an even more detailed analysis is possible. With AMEISE, bar-charts and Gantt-charts can be used. Figure 3(a) visualizes the effort that was spent on the different phases of the project, but, as shown in Figure 3(b), AMEISE is able to break down the data to the exact time-span the developers needed to fulfill their tasks as well as to intermittent idle time.

These tables and diagrams are highly appreciated by the trainees, and it turned out that their reflection and interpretation substantially contribute to the success of our lectures.



(a) (b)

Figure 3: Sample screen shots of diagrams generated by the feedback component for different simulation runs. (a) Diagram showing the distribution of effort during the phases of the project. (b) Gantt-chart showing developers involved in the system and module design phase.

4.3 Software engineering education support

With a simulation environment that makes internal data (product and process-related) explicit, a lot of opportunities arise. By varying the standard educational setting and by varying the type and amount of feedback given, different knowledge areas of a SE education curriculum can be supported. As AMEISE stores a lot of data in the database, a substantial part of the SEEK areas can be covered:

- Modeling and Analysis (MAA): here, the importance of requirements can be stressed. The AMEISE simulation run is used to introduce and motivate for the Function Point count and solid upstream work.
- Professional Practice (PRF): group and communication skills are essential. AMEISE can be used to show the influence of the different skills of the group members and to demonstrate effects of communication overhead. In our courses we achieve this by e.g. comparing the results of different development teams or the results of different review strategies the trainees are using. There, the effect of the skills of the developers and also the communication overhead in teams can be made explicit. On the other hand the narrow communication channel to a purely managerial supervisor is stressed.
- Verification and Validation (VAV): the importance of reviews and testing can be stressed. The quality assurance model mentioned in Section 4.2 explicitly focuses on these two activities. The measures AMEISE is collecting make the quality of the process explicit, and different project decisions can be quantitatively compared.
- Software Process (PRO): AMEISE supports phase-driven process models, thus the environment can be used to show how these models work and how the process is implemented.
- Software Quality (QUA): this area is closely related to the VAV area. When talking about standards and processes, AMEISE can be used to show the importance of QA-concepts. It can also be used to demonstrate what quality is, and how it can be achieved.
- Software Management (MGT): planning, person management, and control are the core of the AMEISE environment. It can thus be used to cover the whole development life-cycle and to show how single management decisions can influence the overall project success.

Institution	AAU		TUKE	CUAS	
Degree Type	Bachelor	Master	Master	Bachelor	Bachelor
Degree Program	Information Management.	Informatics	Informatics	Telematics/Network Engineering	Med. Information Technology
(Proposed) Term	4 or 6	4	3	4	4
Course Title	IT Project Mgmt. and Change	System Develop-ment Process	SW Quality & Management	Software Engineering	Software Engineering
Total Course Hrs.	45	22.5	48	45	45
SEEK KA per Course (Class-room Hours)	PRF(8), MGT(20), EVL(4), PRO(4), QUA(4), VAV(2)	PRF(2-4), VAV(4), EVL(0-4), PRO(6-8), QUA(4), MGT(2-4)	FND(10), VAV(10), PRO(8),QUA(8), MGT(8), EVL(4)	MAA(18), DES(8),PRF(4), VAV(6), PRO(2), QUA(3), MGT(4)	MAA(18), DES(8),PRF(4), VAV(6), PRO(2), QUA(3), MGT(4)
AMEISE Period	2006 – 2011	2002 – 2011	2008 - 2011	2003 - 2011	2006 - 2009
Students in Total	205	277	300	188	65
Simulation Runs	2 – 3	2	2 – 3	2 – 3	2
Students/Sim.	(2 to) 3	2 (to 3)	2 (to 3)	2 (to 3)	2 (to 3)
AMEISE Effort/Student (h)	14 – 16	10 – 12	12 – 15	12 – 15	12 – 15
Didactical AMEISE Focus	staffing/planning project control cause/effect analysis	process models, QA, and project control	SW metrics, phase model exp. QA, history of informatics	phase model exp. RE & QA cause/effect analysis	phase model exp. quality assurance cause/effect analysis

Table 2: AMEISE Simulations at Alpen-Adria Universität (AAU) Klagenfurt, Technical University of Košice (TUKE), and Carinthia University of Applied Sciences (CUAS) Klagenfurt.

As can be seen in Table 2, Mathematical Fundamentals (FND) and Software Evolution (EVL) are part of our courses, but they are not directly supported by the AMEISE framework. AMEISE can be used to motivate for the use of measures, and there is also one model focusing on different maintenance tasks. However, this maintenance model is not in use at our institutions, and thus no experience concerning this model could be collected so far.

5. AMEISE experience

Over the last nine years, AMEISE simulation runs were carried out in order to amplify as well as support several courses on SE or project management. This section summarizes the experience gained and reports on effects observed during the various AMEISE courses.

In all our courses we applied the standard educational setting as explained in section 4.2 with the first simulation run usually taking place in a classroom situation. We encourage students to work in teams of two persons in order to stipulate discussions and properly reflect on decisions. For all our simulations we used the same phase-driven process model with quality assurance (cf. 4.2). Table 2 gives an overview of the different settings of our courses using AMEISE. Due to different curricular objectives and restrictions the courses are designed for serving different didactical purposes.

- *AAU:* AMEISE is part of two courses of different curricula, one on the bachelor's and one on the master's level. The topics of the lectures are fixed (IT Project Management and Software Development Process), but the backgrounds of the students attending the courses are usually different. Depending on their area of specialization they know about SE basics or not. AMEISE simulation runs are mainly used as a motivating factor. They are obligatory, but not part of the final grading. However, the feedback we get at the end of the semester always emphasizes that the simulation runs belong to the most interesting and instructive parts of the lectures.

- *TUKE:* The syllabus of the course is divided into 2 parts. The first part is technically oriented and devoted to the internals of the object-oriented software and then focuses towards software metrics in general. In other words, the first part deals more with software as product. The second part is devoted to managerial aspects, focusing on the processes in SE. The course is merged with lectures given by external practitioners who highlight the experimental side of software engineering in industry. Laboratory practices are based on AMEISE simulation runs. Within the Erasmus and CEEPUS program several students helped creating auxiliary files and installation scripts on the local AMEISE server.
- *CUAS:* In contrast to the two other universities' curricula, the CUAS study programs include informatics merely as a basic subject area with the software engineering course as the only SEEK-related one. Hence, the course is designed to cover a broad spectrum of SEEK knowledge areas on an introductory level. While students learn the application of analysis and design methods by modelling small lab examples, AMEISE is used to practice basic concepts in processes and management by simulation. The related educational objectives include the proper understanding of process models and their relevance for the success of development projects, acknowledgement of requirements engineering and quality assurance as critical success factors, and the comprehension of effects on management decisions in case of restricted resources.

In spite of the different course settings and backgrounds (as discussed above), we experienced some common effects like:

- *Competition motivates.* Students at AAU and CUAS took part in game-like competitions; TUKE students received even their credits on basis of their performance. This was motivation enough to strive for best simulation results.
- *Preparation pays off.* Students putting adequate effort on planning achieved better simulation results than others who started with their simulation run too early.
- *Feedback is essential.* Feedback sessions (especially after the very first simulation run) discussing evaluation results are essential for understanding cause/effect relationships and for improving the evaluation results in subsequent simulation runs.
- *Simulation experience "validates" theoretical knowledge.* With small effort, students can apply their theoretical knowledge about process models and project management. They can experiment with different management decisions and strategies and compare the achieved effects in a simulation environment using a process model based on valid empirical data. Hence, the theoretical knowledge initially transferred by lectures can be experienced by students through simulation.

6. Conclusion

The vast variety of skills and knowledge to be covered by SE curricula, the diversity of curricular restrictions to be obeyed during the development of related study programs, and the rapid evolution in information technology call for highest flexibility in designing adequate SE courses. Though the study programs of our three institutions aim diverse objectives emphasizing different SEEK knowledge areas, we demonstrated in this paper that a simulation environment like AMEISE can be commonly utilized and customized for supporting varying didactical concepts in teaching Software Engineering.

Generally, experimental learning by simulation can be applied to speed-up and deepen various core aspects in SE education. Moreover, the AMEISE simulation framework with its full range of assessment possibilities and its potentially broad coverage of SE topics is flexible

enough to be continuously beneficial even in cases of SE course redesign necessary in response to evolving environmental constraints and real-world needs.

Acknowledgement

We are grateful to the reviewers of CSEE&T 2012 for their fruitful comments and suggestions to improve this paper. The work at TUKE described in this paper was partially supported by the project: KEGA 050-023TUKE-4/2010 – Modern Software Engineering in Education - Proposal of the Structure and Realization of Actual Software Engineering Subjects for Informatics Study Programme at Technical Universities.

References

[1] ACM IEEE, "Software Engineering 2004", 23 August 2004. [Online]. Available: http://sites.computer.org/ccse/. [Accessed: Jan. 23rd, 2012].
[2] European Communities, "ECTS Users' Guide", [Online]. Available: http://ec.europa.eu/education/lifelong-learning-policy/doc/ects/guide_en.pdf", Feb. 6th, 2009. [Accessed: Jan. 23rd, 2012].
[3] A. Bollin, E. Hochmüller and R. Mittermeir, "Teaching Software Project Management using Simulations", in 24th IEEE-CS Conference on Software Engineering Education and Training (CSEE&T 2011), Honolulu, USA, May 2011.
[4] R. T. Mittermeir, E. Hochmüller, A. Bollin, S. Jäger and M. Nusser, "AMEISE – A Media Education Initiative for Software Engineering: Concepts, the Environment and Initial Experiences", in Proceedings International Workshop ICL – Interactive Computer Aided Learning, Villach, Sept. 2003.
[5] B. Meyer, "Software Engineering in the Academy", IEEE Computer, pp. 28-35, May 2001.
[6] B. Meyer, "Reality: A cousin twice removed", IEEE Computer, pp. 96-97, July 1996.
[7] B. Meyer, "Testable, Reusable Units of Cognition", IEEE Computer, vol. 39, no. 4, pp. 20-24, April 2006.
[8] A. Pyster, R. Turner, D. Henry, K. Lasfer and L. Bernstein, "Master's Degrees in Software Engineering: An Analysis of 28 University Programs", IEEE Software, pp. 94-101, September/October 2009.
[9] P. N. Robillard, "Opportunistic ProblemSolving in Software Engineering", IEEE Software, pp. 60-67, November/December 2005.
[10] G. W. Hislop, Software Engineering Education: Past, Present, and Future, IGI Global, 2009.
[11] T. C. Lethbridge, L. J. Richard, J. Diaz-Herrera and B. J. Thompson, "Improving software practice through education: challenges and future trends", in Future of Software Engineering, FOSE '07, 23-25 May 2007.
[12] J. Buchta, M. Petrenko, D. Poshyvanyk and V. Rajlich, "Teaching Evolution of Open-Source Projects in Software Engineering Courses", in Proceedings of the 22nd IEEE International Conference on Software Maintenance, pp. 136-144, 2006.
[13] M. I. Kellner, R. J. Madachy and D. M. Raffo, "Software Process Simulation Modeling: Why? What? How?", Journal of Systems and Software, vol. 46, no. 2/3, pp. 1-18, 15 April 1999.
[14] H. Zhang, B. Kitchenham and D. Pfahl, "Reflections on 10 Years of Software Process Simulation Modeling: A Systematic Review", in Making Globally Distributed Software Development a Success Story, Springer Berlin / Heidelberg, Lecture Notes in Computer Science, Q. Wang, D. Pfahl and D. Raffo, Eds., 2008, pp. 345-356.
[15] E. S. Monsalve, V. M. B. Werneck and J. C. Sampaio do Prado Leite, "Teaching Software Engineering with SimulES-W", in Software Engineering Education and Training (CSEE&T), Honolulu, HI, 22-24 May 2011.
[16] L. Samuelis and A. Bollin, "Experiences gained in teaching software project management by simulation," in Informatics 2009: Proceedings of the Tenth International Conference on Informatics, Herľany, Slovakia, November 23-25, 2009.
[17] P. Borque and R. Dupuis, "Guide to the Software Engineering Body of Knowledge", IEEE Computer Society, 2004 version. [Online]. Available: www.swebok.org. [Accessed: Jan. 23rd, 2012].
[18] IEEE, "Guide to the Software Engineering Body of Knowledge - SWEBOK," Technical Report, ISO/IEC 19759:2005, Software Engineering, 2005.
[19] IEE/ACM Joint Task Force on Computing, "Computing Curricula 2005, the Overview", 2005. [Online]. Available: http://www.acm.org/education/curricula-recommendations. [Accessed 16 November 2011].
[20] AMEISE - A Media Education Initiative for Software Engineering. Funded by the bm:bwk under Grant NML-1/77, 2001. [Online]. Available: http://ameise.uni-klu.ac.at. [Accessed: Nov. 16th, 2011].
[21] J. Ludewig and A. Drappa, "Simulation in Software Engineering Training", in Proceedings, 23rd International Conference on Software Engineering, May 2001.
[22] R. T. Mittermeir, "Facets of Software Evolution", in Madhavji, N.H., Lehman, M.M., Ramil, J.F. and Perry, D.W. (eds.): Software Evolution, Wiley, 2006.