A Practical Approach to align Research with Master's Level Courses

Marco Kuhrmann
Technische Universität München
Faculty of Informatics, Software & Systems Engineering
Garching, Germany
kuhrmann@in.tum.com

***Abstract*—Software Engineering is a discipline in computer sciences that covers different topics ranging from formal methods to practical topics. An essential part of Software Engineering is the organization and the management of software projects. From several studies we know that we master the "craftsmanship", which means coding, but suffer in the organizational topics, i.e. project or process management. Those topics are important to Software Engineering, however, they are rather boring for students, which makes it hard to enthuse them about such topics. During the last years we developed a teaching format that on the one hand covers those high-level and abstract topics and, on the other hand, provides students with the opportunity to have experiences, and lecturers to conduct research. In this paper we present a concept for courses that combines the classic teaching formats lecture, seminar and practical training into a new format that allows for interactive teaching as well as for conducting research. We contribute a blueprint, which can be implemented in further courses. We present experiences made in a first implementation in the area of software process management, and conclude the paper with a discussion and a teaching agenda.**

***Keywords*-education; teaching methods; lecture design; software engineering**

I. Introduction

Software Engineering (SE) aims at developing software systems in a systematic, methodically sound, and economic manner to master the key challenges of time constraints, budget adherence, quality and functionality. Beyond the software technology that covers mainly technical aspects such as programming, SE also covers organizing and managing software projects.

In Software Engineering education, however, we focus on theoretical basics, architecture, requirements engineering, programming, or other advanced (technical) topics. When it comes to more abstract or methodical topics we face the problem that students are "bored beyond belief". To give an example: During the last three years, our chair gave a lecture on project management that about 450 students (bachelor's level) attended. At the same time, we offered an advanced lecture on process management (master's level), a practical training on Global Software Development (GSD [1]) and several examination papers. Summarized, about only 30 students (of those 450 candidates) were interested in those advanced topics, which is less than 10% of the potential number of participants. We had to realize that other topics such as iPhone programming, robotics or game engineering attracted significantly more students. A possible explanation is the "fun factor". Another reason is that most of the students, even if they work in a company, have no experience with long-term projects in larger teams and complex software systems that go beyond prototypes. Most of the students have no relation to such topics. Partners from industry note that the students are usually not ready for work, even if they master a couple of programming languages. An extreme feedback: "I need to qualify a graduate for 2 or 3 extra months to make him fit…" Besides company-specific knowledge (which a university cannot teach) most partners from industry complain about missing soft-skills *and* missing understanding of how organizations and projects work, which are exactly the objectives of our research and of the advanced courses we offer.

A. Problem Statement

As it is difficult to get students interested in such abstract methodical topics, students have rather few opportunities to gain experiences to understand why organizations and projects work the way they do. Classic teaching formats provide students with theoretical basics and a toolbox. Even practical trainings are limited to prototypes that often do not incorporate with real world problems, e.g., social aspects, communication, negotiation, conflict management, and so on. From the lecturers perspective teaching is also often not aligned with current research. Consequently, lecturers may not be motivated to give interesting lectures, but to finish a class as fast as possible to get back to research. From the students' perspective this is a further demotivating factor.

Yet missing is a practicable approach to (1) teach theoretical basics, which are (2) combined with practical trainings, and accompanied (3) by research activities.

B. Contribution

We contribute a blueprint for a teaching format at master's level, which combines several classic teaching formats to create a "win-win" situation where teaching and research go side by side. To this end we provide a detailed description of the course blueprint and report on a first implementation. We furthermore discuss our experiences and draw a teaching agenda for further implementations.

C. Outline

The remainder of the paper is organized as follows: In Sect. II we discuss the context and the related work. Section III introduces our approach to combine teaching and research. In Sect. IV we want to share our experiences that we made during the first implementation of the proposed blueprint. We conclude the paper in Sect. V by describing our teaching agenda and future work.

II. Context & Related Work

In this section we describe the context by giving information about typical teaching formats that are established at our faculty. After that we discuss the related work.

A. Established Teaching Formats

At our faculty we have curricula built on a small set of teaching formats, which we will name the established or the *classic* formats. Table I gives a brief summary on the most common teaching formats.

Table I
Brief summary of the established (classic) teaching formats implemented at our faculty.

Format	Description
Lecture	A lecture is a teaching approach in which the lecturer more or less directly instructs the students. Except some questions lectures are, however, not interactive. Lectures are, usually, supplemented by exercises in which theoretical contents are "practically" applied. Exams are either oral or written (mid and/or end term).
Seminar	A seminar is a format in which students work independently on individual topics. Usually, advisors deliver the topics and guide the students. As outcome the students give presentations and write an essay on their particular topic. The final grade is based on the evaluation of both, the presentation as well as the essay.
Practical Training	Practical labs/sessions/trainings (short: training) focus on transferring theoretical knowledge into a practical environment. This format is usually applied when learning programming languages. The final grade is based on the (set of) outcomes, e.g., software.
Guided Research	This teaching format addresses students, who are interested in participating in current research. Students work as a part of the research group on current topics. The goal of this teaching format is to get a paper published. Advisors, e.g., senior researchers, evaluate the student's performance on which the grade is determined. This teaching format is especially designed to support a long-term investigation, which is complemented by a Master's Thesis.
Theses	Theses are comprehensive pieces of work that students write individually to get their academic degrees, e.g., Bachelor's or Masters' Theses.

Since 22 chairs at our faculty compete for the students, it is hard to interest more than a handful of students for specific topics. Furthermore, since the teaching formats shown in Table I are usually closed in themselves the creation of consecutive courses is challenging. Only the *guided research project*, which is a rather new format, allows for bridging the gap between the Bachelor's Thesis and the Master's Thesis by integrating students into the research group.

B. Related Work

Since Software Engineering is also affected by many (non-technical) disciplines, Ghezzi and Mandrioli [2] state that teaching SE in isolation is not meaningful. The main challenge when teaching SE is to provide students with a mixture of theoretical basics and practical skills. It seems to be common sense that teaching SE has to include practical parts. Especially Gnatz et. al [3] note that SE cannot be thought using text books. An example for an insufficient way to teach project management is the PMBOK Guide [4] – having read this book does not make students project managers, nor understanding project situations, nor enabling them to apply project management methods in practice. Instead, Gnatz et. al present an approach to combine theory and practical hand-on-lab sessions; a pattern that is also applied in [5]–[8]. Those courses mainly address the topics of project management and systematic software development in (small) teams, external or internal clients in one- or multi-site [9], [10] settings. Mandl-Strieglitz [11] proposes a simulation-based approach to teach students project management that gives them the opportunity to make "real" experiences, similar to what we have reported in the context of GSD [1]. Combining practical parts and theory becomes hard (or impossible) if the curriculum is not based on an appropriate strategy. Even if there are enough students, there is a huge uncertainty w.r.t. the pre-knowledge. Another aspect is the researchers' point of view. Pádua [12], for instance, investigates the performance of SE courses, while Chen and Chong [13] investigate a particular topic of interest. It is a basic question to determine, what the added value beyond teaching is and how can courses contribute to current research. In [12] a systematization is provided. Nevertheless, such an approach works only if the course is self-contained, if the particular lecture is given at least once a year, and the course can attract a comparable number of students per term.

The idea to combine teaching formats such as (theoretical) lectures, practical sessions, and also research activities is not new. Especially the integration of lectures and practical sessions is state of the art, but missing a link for systematically conducting research. The challenge is to justify all teaching formats so that they fit into one course. We did not have an appropriate blueprint available at our faculty. Considering, e.g., the aforementioned key contributions we designed a new format and inferred a blueprint from our experiences.

III. Combining Teaching and Research

The paper at hands describes a new teaching format that combines the established formats into an integrated course that not only combines theory and practice (from a student's

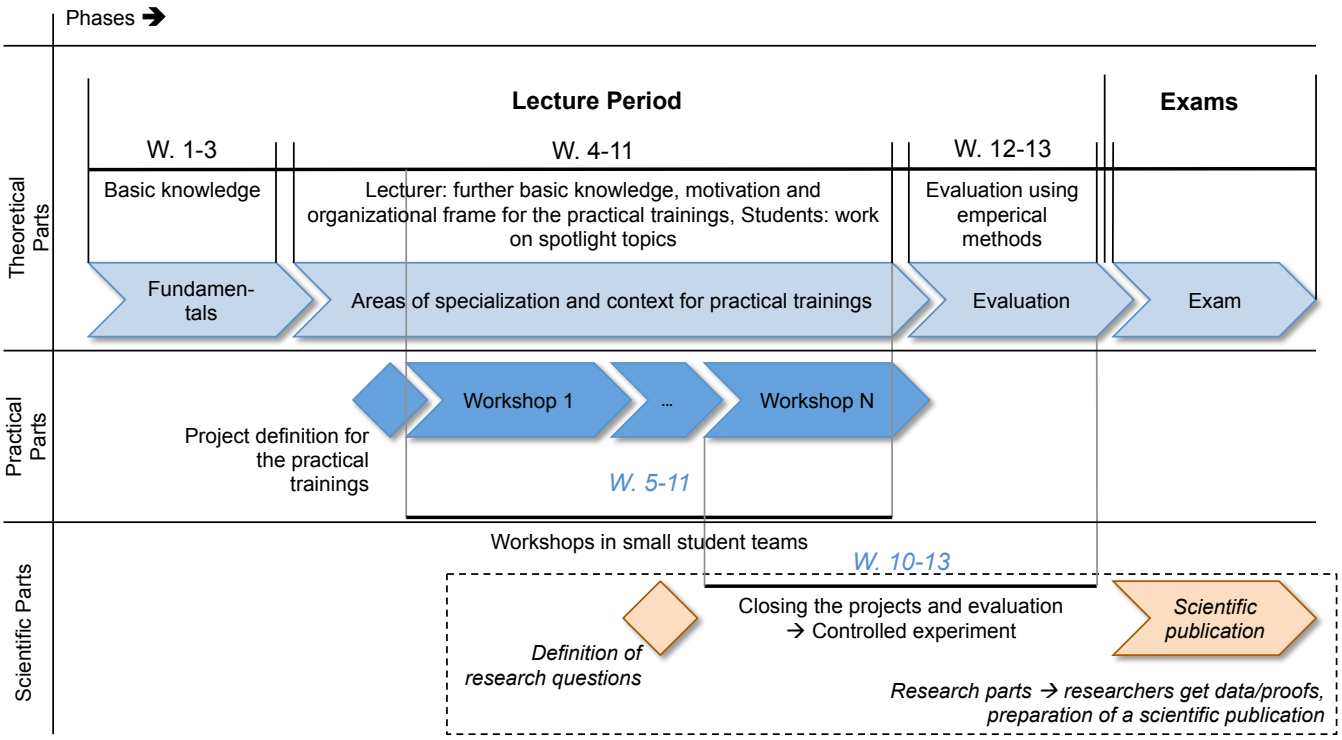

Figure 1. Course blueprint showing the theoretical (light blue), the practical (dark blue), the scientific parts (ochre), and a planning pattern.

point of view), but also gives space to systematically conduct research[1]. This makes teaching an integrated part of researchers' work rather than a burdensome obligation. In the following we describe the key concept we created based on our experiences.

A. Goals

Our teaching format aims at systematically combining teaching and research, and to create a "win-win" situation where students (1) get high-quality courses, can (2) participate in current research and learn about current and hot topics, and (3) researchers can include student groups into their research activities.

B. Audience

Our teaching format addresses lecturers that need to create specialized and/or advanced courses at the master's level. It is targeted to small or medium groups of up to 25 students.

C. Organization Blueprint

In this section we present the teaching format in a blueprint-style that allows for simple adaptation and reuse. The blueprint (Fig. 1) is based on certain assumptions. The calculation is based on a semester with 13 weeks of lectures. A pilot implementation of this blueprint and first experiences are presented in Sect. IV.

1) Theoretical Parts (light blue): The course consists of three phases. In the first phase (about three weeks) the fundamentals and the basic knowledge is imparted. During those first three weeks the lecturer acts as a classical "teacher" (classic lecture style).

In the second phase of about eight weeks the lecturer prepares the frame for the spotlight topics that cover areas of specialization. The students work independently on the spotlight topics; they prepare presentations and small essays to summarize their outcomes. During this second phase of the course, the lecturer is in the role of a mentor/consultant and moderates the interactive sessions. Also, during this phase the practical parts are prepared.

The third phase (about two weeks) is the evaluation phase. Outcomes from the practical parts are evaluated according to scientific methods. Since students usually do not know much about empirical methods the lecturer is, again, in the role of the "teacher". On the other hand, the lecturer is also in the role of an auditor and guides the evaluation of the outcomes from the practical phases.

2) Practical Parts (dark blue): At the time when the second phase of the theoretical parts starts, the classical exercises (which can be used at the very beginning of the course, i.e. two sessions) are replaced by practical trainings.

[1]This teaching format was awarded with the "Ernst Otto Fischer Teaching Award" (2012) by the Faculty of Informatics, Technische Universität München (http://portal.mytum.de/studium-und-lehre/lehrpreise/ernst_otto_fischer_lehrpreis.html).

Therefore, the lecturer needs to prepare a *project assignment* in which the project objectives are contained. The project objectives need to be aligned with the course's contents w.r.t. the agenda of the overall course and topics that can be worked on in a self-contained manner, e.g., work packages that can be handled in one session. The work packages have to support continuous work to avoid repeated incorporation. Another important aspect is that outcomes of the practical parts need to be prepared for evaluation and thus, need to be defined in order to be measurable.

3) Scientific Parts (ochre): The "experimentation phase" starts around the final workshop of the practical parts. If the considered research objective is appropriately tailored experimentation can, of course, start earlier. Research questions that are defined by the lecturer are mandatory prerequisites for the experimentation. When the research questions have been defined the lecturer checks the tentative results of the passed workshops, and if necessary, consolidates the results to ensure that all teams can start under the same conditions. The experiments can start after all the preparations have been completed. Experiments should be designed to be completed without interruptions within one session of, e.g., 90 minutes or 180 minutes, respectively.

Data, which have gained during experimentation, can be prepared for being published as a scientific publication. However, the publication is written outside the class.

D. Organization Recommendations

The aforementioned blueprint allows for flexible and advanced courses, but requires some organization effort.

Prerequisites: Since the presented teaching format allows for the systematically embedding of research activities into courses, lecturers need more preparation. In addition to "classic" supporting material, e.g., scripts, slides, and so on, lecturers especially have to prepare the artifacts for the project/experimentation setting such as:

- Project assignment for the sample project
- All input artifacts required for the sample project
- Consolidated artifacts for the experiment setting(s)
- Surveys and so on for the evaluation
- Analysis report(s)

Examination: Since the course is not done "as usual" the examination procedure needs to be tailored w.r.t. the concrete setting (and embodiment) of the course. Special attention should be payed to the possibilities of an examination procedure that allows for a continuous evaluation of the students' performance. There is not only one appointment where students have to be in great shape, as students' work continuously over the whole semester for their final grades. For small and medium groups (up to 25 students), the following procedure worked well:

Students prepeare presentations and write essays on their spotlight topics. Those are a part of the final grade. The rest of the grade is given based on an oral exam. The final grade can be found by weighting the parts, e.g., 1/3 for the presentations and 2/3 for the oral exam.

IV. Pilot Implementation

The blueprint was implemented for the first time in the winter term 2011/2012. We describe the concrete class and our experiences.

A. Course Description

The blueprint was implemented for the first time in the second run of the lecture "Software Engineering Processes" (the first lecture was given in the classic way). This lecture gives an introduction to the domain of process engineering and process management [14]–[16]. Students learn about:

- Software processes in general, why they are important and how they relate to organizations that drive software projects.
- Concrete software processes, such as Rational Unified Process [17], Extreme Programming (XP [18]), Scrum [19], V-Modell XT, Kanban, and so on.
- The process ecosystem including, e.g., maturity models (CMMI [20]), processes for IT operation and services (ITIL [21]), and about the relations among them.
- Software process metamodels, e.g., ISO 24744 [22], SPEM [23], V-Modell XT [24], and supporting tools to author software processes and methods.
- The software process life cycle and the (common) tasks needed to be performed to analyze, conceptualize, design, implement, publish, and assess a software process.

This course was implemented at the master's level. In order to provide a high-quality class, to foster interaction, and due to the experimental character, the group size was restricted to 15 students. The whole lecture was organized to be held in a block of 180 minutes per week to create the necessary space for practical trainings and experiments. A schedule for the pilot implementation is shown in Table II. The table boils

Figure 2. Workshop 2: Final artifact and process models (conceptualization) are the results of the workshops. (week 8 and 9, see Table II).

Table II
SCHEDULE OF THE PILOT IMPLEMENTATION OF THE LECTURE "SOFTWARE ENGINEERING PROCESSES" (WINTER TERM 2011/2012).

Week	Phase (Who)	Topics and Description
1–3	Basic Knowledge (lecturer)	Motivation, establishing the relations to and incorporation into Software Engineering, repetition of project management basics, e.g., organization models, project settings/organization, and so on. Fundamentals, terminology, and concepts of software processes
4–7	Context for the project (lecturer) Context-specific knowledge (lecturer) Spotlight topics (students)	Preparation and incorporation of the spotlight topics. Where needed, context-specific background knowledge and detailed information w.r.t. certain topics was provided by the lecturer. Preparation of the sample project (including: motivation of the project, presentation of the requirements, incorporation of the initial supporting material, and so on). Students work on their spotlight topics with focus on specific methods/approaches and tools (e.g., Scrum, Kanban, CMMI, Eclipse Process Framework, SPEM, and so on), and give a presentation in class.
8–9	Two workshops addressing software process analysis and conception (students)	Students work on their practical tasks/work packages in workshops. The first workshop aims at identifying stakeholdes, roles, and responsibilities. The second workshop is focused on artifact structures and processes (see Fig. 2) and prepares the consolidation. Lecturer advises the students w.r.t. their particular tasks.
10	Evaluation of the first workshops and presentation of the outcomes (students) Consolidation of the outcomes and preparation of the experiment (lecturer)	Students present their first outcomes. Lecturer evaluates the outcomes and prepares a consolidated set of outcomes, which are the input for the planned experiment (each experimentation group is provided with the same initial conditions for the experiment). The subject of investigation is the performance of a process model realization depending on the (design) philosophy of a given process framework.
11–12	Realization/implementation workshop (2 student groups)	Students work on the tasks demanded by the (consolidated) project assignment. One group used the Eclipse Process Framework for the process's realization, the other group used the V-Modell XT platform, respectively. The lecturer monitors the experiment.
13	Evaluation/audit of the outcomes (students and lecturer)	In the first part the basics regarding assessment and evaluation were thought, i.e. creation of surveys and assessment fundamentals (lecturer). In the second part of the last session, the students firstly evaluate themselves (self-audit of their own work), and secondly audit the outcomes of the other group.

down the blueprint (Fig. 1) and shows, who is in charge to do certain tasks, what are the particular contents of a phase, and what is the duration a phase.

B. Experiences

We implemented this teaching format in the *second* run of the lecture "Software Engineering Processes". Therefore, we are able to compare both classes and to report some experiences (Table III) that we made during the second run.

C. The Students' Voice

Besides the formal course evaluation by the faculty we performed an internal evaluation. Students were asked to write a *one-minute-paper* that contained the following three questions to be answered in short words:

1) (up to 5) points that are positive
2) (up to 5) points that are negative
3) (up to 5) points that I still wanted to say (informal)

Table IV summarizes the evaluation of the one-minute-papers. Those opinions were considered during summarizing our experiences (Table III).

D. Drawbacks & Critical Discussion

Summarized, we consider this new teaching format to be a success. Nevertheless, we realized some drawbacks that need to be critically discussed:

Table IV
EVALUATION OF THE ONE-MINUTE-PAPERS.

Pro
Structure of the topics and the class, Combination of theory and practice, Projects in teams (atmosphere), Self-motivation due to presentations, Continuous evaluation and finding of the final grades
Con
Tough schedule, Tailoring of the tasks for the practical sessions was not always optimal Students signed off, just because of the examination procedure
Informal
"Thank you, this was the lecture I learned the most." "Super class, and I loved those many samples from practice."

Effort: From the lecturer's point of view, we have to note that the preparation and the coordination of the pilot class caused much effort (about twice the effort of running this lecture in the classic shape as done the year before). Although some preliminary work is available, there were no experiences in running a class this way. In consequence, we cannot yet generalize and thus have to implement the new teaching format again to increase our experience base.

Complexity of the samples: Teaching students with real world examples requires appropriate samples. However, a lecture aims at educating students and not at solving

Table III
EXPERIENCES AND LESSONS LEARNED (👍 : POSITIVE, 👎 : NEGATIVE, 👎/👍 : REQUIRES MORE EXPERIENCE FOR A FINAL RATING).

Conext & Experience	Lesson Learned	
The requirement that students become active themselves, work on spotlight topics, and present their outcomes in class, made students feel inconvenient for the first moment. Those feelings faded away during the first two presentations.	Students improve their presentation skills	👍
Since the spotlight topic presentations fostered the discussion, we had to tighten the schedule. We decided to limit the presentation slots to 15 minutes to have more space for the discussion.	Discussion is important but can get out of control	👎/👍
Each presentation was supplemented by a short essay of 4-5 pages (LNCS style). The essay should summarize the basic facts and findings. This approach worked well, since all students had summaries of all topics available.	Students document their findings, but need assistance (review) to write the essays	👎/👍
Not only working on a synthetical sample, but on a real world process has proved successful. For this class we got a process by the working group *Software Processes* of the German Computer Society.	A real-world-example is of advantage, but it needs to be tailored in advance	👎/👍
Working in self-organizing teams has proved successful. Each student had his tasks and could also give information w.r.t. the overall project at anytime.	Students work independently and identify themselves with the project	👍
Combining active and interactive parts during lectures, exercises, and practical trainings (the project) proved successful. Students actively communicated with each other without the usual anxieties.	Students improve their communication skills	👍
Mutual evaluation of the process realizations proved successful. Each student did a self-assessment according to given criteria, and in a second step, reviewed the outcomes of the other group (according to the same criteria). Everybody respects everybody else's work.	Although students are usually not experienced in evaluation, they learn to rate other's work	👍
In the first session we presented the examination mode. We told the students that we would find the final grades by a continuous evaluation, that they would have to give two presentations (including the essay) each, and that they have to pass an oral exam. Some students had fear of "too much" work and signed off.	The examination mode is fair, but causes effort for the students, which they try to avoid	👎/👍

somebody else's problems. Therefore, sample projects need to be tailored in advance to fit in the lecture, which means that complexity needs to be reduced including a reduction of "reality".

The "right" research question: Our main goal is to provide lecturers with the opportunity to combine teaching and research. Since the schedule of a lecture is fixed, research questions need to be tailored to fit in the classes. Due to the 180 minute time slots, we learned that (1) research has to be intensively prepared and planned, and that (2) not every research objective is reasonable to be treated in such a setting.

V. TEACHING AGENDA & FUTURE WORK

We presented a teaching approach that combines the classic lecture with seminars and practical trainings, and allows for systematically conducting research. Lecturers give a lecture on a particular topic, and provide students with the opportunity to work independently on spotlight topics, and thus foster interaction in classes. Furthermore, lecturers can define projects/experiments that are performed during practical sessions. Students work on real problems that are motivated and backed-up by the theoretical contents of the lecture. They can transfer the theoretical knowledge to gain experiences. The lecturer can align the project with his research in order to get data (e.g., experiments), or to evaluate current research (e.g., experiments, surveys, prototypes). Summarized, we extended the classic lecture by *reorganizing* the schedule to integrate interactive parts and to give researches space to conduct research without reducing the contents of teaching.

Experiences: Our experiences from the pilot implementation show that this teaching format met our expectations. Classes were interactive, students got a theoretical foundation as well as in-depth knowledge w.r.t. spotlight topics. The students worked practically, experienced real world problems, and learned how to deal with problems. The exams showed that students learned more effectively and *understood* the contents better than in the first (classic) run of the lecture. Furthermore, we could perform a controlled experiment and thus could integrate current research.

The pilot implementation causes, however, about twice the effort in organizing the lecture than the classic format. Since we were not experienced with this particular teaching format, the additional effort may decrease, which is also a subject for further investigation.

Future Work – A Teaching Agenda: Since the pilot implementation met our expectations we will implement the new format in several new lectures. For the winter term 2012/2013 we are already preparing a lecture on agile project management methods[2]. Since agility allows for manifold experiments (e.g., team building, social aspects, development performance, software quality, and so on) we can apply our blueprint to teach the students project management while conducting several experiments. We also discuss with

[2]Lecture "Agile Project Mangement & Software Development", winter semester 2012/2013, master's level, http://www4.in.tum.de/lehre/vorlesungen/vgmse/ws1213/index.shtml

lecturers at our chair, whether they want to (partially) adapt our concept in their lectures. Furthermore, we are looking for more lecturers, who are interested in this teaching format. As a first step we start to disseminate our concept at our faculty. Finally, the lecture "Software Engineering Processes" will be given again in the winter term 2013/2014 to monitor the implementation (quality) of our concept.

ACKNOWLEDGEMENTS

First of all, we want to thank Manfred Broy, who gave us the freedom to create this new teaching format at his chair. Furthermore we want to thank Daniel Méndez Fernández and Georg Kalus for their essential and valuable support during the implementation of the lecture "Software Engineering Processes", and Manuel Then for his support on the exercise materials. Finally, we want to thank Katharina Spies for her organizational support to set up the lecture, and Birgit Penzenstadler, Sebastian Eder, Jonas Eckhardt, and Hennig Femmer for reviewing this paper.

REFERENCES

[1] Deiters, C., Herrmann, C., Hildebrandt, R., Knauss, E., Kuhrmann, M., Rausch, A., Rumpe, B., and Schneider, K., "GloSE-Lab: Teaching Global Software Engineering," in *Proceedings of 6th IEEE International Conference on Global Software Engineering*. IEEE, 2011.

[2] Ghezzi, C. and Mandrioli, D., "The Challenges of Software Engineering Education," in *International Conference on Software Engineering (ICSE)*. ACM, 2005.

[3] Gnatz, M., Kof, L., Prilmeier, F., and Seifert, T., "A practical approach of teaching software engineering," in *Conference on Software Engineering Education and Training*, 2003.

[4] Project Management Institute, *A Guide to the Project Management Body of Knowledge*, 4th ed. Project Management Institute, 2009.

[5] Richardson, I., Milewski, A. E., and Mullick, N., "Distributed Development — an Education Perspective on the Global Studio Project," in *International Conference on Software Engineering (ICSE)*. ACM, 2006.

[6] Huang, L., Dai, L., Guo, B., and Lei, G., "Project-Driven Teaching Model for Software Project Management Course," in *International Conference on Computer Science and Software Engineering*. IEEE, 2008.

[7] Dahiya, D., "Teaching Software Engineering: A Practical Approach," *ACM SIGSOFT Software Engineering Notes*, vol. 35, no. 2, 2010.

[8] Bavota, G., De Lucia, A., Fasano, F., Oliveto, R., and Zottoli, C., "Teaching Software Enginerring and Software Project Management: An Integrated and Practical Approach," in *International Conference on Software Engineering (ICSE)*. IEEE, 2012.

[9] Brereton, P., Gumbley, M., and Lees, S., "Distributed student projects in software engineering," in *Conference on Software Engineering Education and Training*. IEEE, 1998.

[10] Keenan, E., Steele, A., and Jia, X., "Simulating Global Software Development in a Course Environment,," in *International Conference on Global Software Engineering (ICGSE)*. IEEE, 2010.

[11] Mandl-Striegl itz, P., "How to successfully use software project simulation for educating software project managers," in *Frontiers in Education Conference*. IEEE, 2001.

[12] Pádua, W., "Measuring complexity, effectiveness and efficiency in software course projects," in *International Conference on Software Engineering (ICSE)*. ACM, 2010.

[13] Chen, C. and Chong, P., "Software Engineering Education: A study on conducting collaborative senior project development," *The Journal of Systems and Software*, 2010.

[14] W. S. Humphrey, "The software process: Global goals," in *International Software Process Workshop (SPW)*, ser. Lecture Notes in Computer Science. Springer, 2005.

[15] D. Rombach, "Integrated Software Process and Product Lines," in *International Software Process Workshop (SPW)*, ser. Lecture Notes in Computer Science. Springer, 2005.

[16] A. Goodman, *Defining and Deploying Software Processes*. Auerbach Publishers Inc., 2005.

[17] P. Kroll and P. Kruchten, *The Rational Unified Process Made Easy – A Practinioner's Guide to RUP*. Addison-Wesley, 2003.

[18] K. Beck, *Extreme Programming*. Addison-Wesley, 2003.

[19] K. Schwaber, *Agile Project Management with Scrum*. Microsoft Press, 2004.

[20] R. Kneuper, *CMMI: Improving Software and Systems Development Processes Using Capability Maturity Model Integration (CMMI-Dev)*, 1st ed. Rocky Nook, 2008, no. ISBN: 978-3898643733.

[21] Office of Government Commerce, *ITIL Lifecycle Suite 2011*. The Stationery Office Ltd., 2011.

[22] Joint Technical Committee ISO/IEC JTC 1, Subcommittee SC 7, "Software engineering – metamodel for development methodologies," International Organization for Standardization, Tech. Rep. ISO/IEC 24744:2007, 2007.

[23] OMG, "Software & Systems Process Engineering Metamodel Specification (SPEM) Version 2.0," Object Management Group, Tech. Rep., 2008.

[24] Federal Ministery of the Interior, "V-Modell XT Online Portal," Online, 2010. [Online]. Available: http://www.v-modell-xt.de/