Teaching Empirical Software Engineering Using Expert Teams

Marco Kuhrmann, University of Southern Denmark
kuhrmann@acm.org

Abstract

Empirical software engineering aims at making software engineering claims measurable, i.e., to analyze and understand phenomena in software engineering and to evaluate software engineering approaches and solutions. Due to the involvement of humans and the multitude of fields for which software is crucial, software engineering is considered hard to teach. Yet, empirical software engineering increases this difficulty by adding the scientific method as extra dimension. In this paper, we present a Master-level course on empirical software engineering in which different empirical instruments are utilized to carry out mini-projects, i.e., students learn about scientific work by doing scientific work. To manage the high number of about 70 students enrolled in this course, a seminar-like learning model is used in which students form expert teams. Beyond the base knowledge, expert teams obtain an extra specific expertise that they offer as service to other teams, thus, fostering cross-team collaboration. The paper outlines the general course setup, topics addressed, and it provides initial lessons learned.

1 Introduction

Software engineering aims at the systematic application of principles, methods, and tools for the development of complex systems. This comprises the *software technology* as well as the *software management* part of this discipline [5], and in each of these parts, humans are involved. Due to this human involvement and the multitude of fields for which software has become crucial, software engineering is considered hard to teach. Literature is rich and discusses different experimental settings [15], in-class projects [23], or project courses [6] in general—each addressing technology (e.g., analysis, coding, and testing) or management issues (e.g., project management, the software process, and teams and soft skills [7, 30]).

However, most of the software engineering courses address the system/product development. Yet, when can a project be considered efficient? How to select methods having a higher probability of success in a specific context? How can the dis-/advantages of certain technologies, methods, or tools be evaluated? In order to make software engineering claims measurable, *empirical software engineering* is applied to (i) analyze/understand phenomena in software development, (ii) identify/evaluate strengths and weaknesses of software engineering approaches, and (iii) investigate the state of the art/practice to identify promising solutions/approaches. This makes empirical software engineering hard to apply for researchers and practitioners, but it makes it even harder to teach. Wohlin et al. [45] consider the engineering method and the empirical method variants of the *scientific method* [3]. That is, teaching empirical software engineering means teaching the scientific method and their adaptation to software engineering. However, scientific work differs from "pure" system development. For example, while a development project can be carried out in a semester project, a sound empirical investigation is harder to implement, since resources required for this purpose would then be missing in the development. Also, students would need to know, e.g., how to set up experiments or surveys, how to conduct them, and how to analyze and make use of the findings—again, not directly contributing to a small project with a deadline and a working piece of software as the desired outcome.

So, what to do? Wohlin [43] considers three general options to teach empirical software engineering: integration in software engineering courses, as a separate course, and as part of a research method course. Yet, these approaches have some difficulties. For instance, in a theoretical course, students would hear about different empirical instruments, could train selected methods, or review and discuss research papers. According to Dale's *Cone of Learning* [10], those activities would largely remain at the passive level (see further Section 2). So, what would remain? Understanding the scientific method in general and empirical software engineering in particular and to see its value requires hands on. That is, staying in Dale's model, a course on empirical software engineering also needs to cover the active levels of the cone.

Objectives This paper aims at providing a course that helps students learning scientific work by *doing* scientific work. However, scientific work requires collaboration, causes effort, and consumes time. Furthermore, quite often, students lack skills crucial to

scientific work, such as to carry out a comprehensive literature research, exact problem definition, statistics, or professional writing.

Therefore, the main challenge to be addressed is to define a teaching format that (i) provides students with the basic knowledge concerning scientific work, (ii) enables students to understand the role of empirical research in software engineering, and (iii) to train scientific work by carrying out (small) research projects and to run through a research cycle, including presentation, writing, and reviewing.

Contribution The paper at hand contributes a course design for an empirical software engineering course and experiences from a first implementation. The overall design follows an approach that brings teaching closer to research [27], and that was successfully applied to different methodical topics, in particular software process modeling [24] and advanced project management [26]. Different that the other implementations of the base concept, the course presented in the paper at hand addresses large classes (50+ students). To keep this course manageable, *expert teams* were introduced. Each of these teams focuses on a specific competency beyond the general knowledge and offers this competency to other teams. That is, in addition to the intra-team collaborations, a cross-team collaboration pattern is implemented. The course evaluation shows the selected approach reasonable; in particular, students consider the course challenging yet good. More important, students changed their view on scientific work and started to consider it valuable.

Outline The remainder of this paper is organized as follows: Section 2 sets the scene by providing background information. Section 3 presents the course design including learning goals, organization, course layout, team structures, and deliverables. Section 4 provides insights into the initial implementation and and evaluation. The paper is concluded in Section 5 with discussion on the lessons learned so far.

2 Fundamentals and Related Work

Empirical software engineering and its integration with software engineering curricula was for instance elaborated by Wohlin [43], who mentioned three general levels of integration: integration in software engineering courses, as a separate course, and as part of a research method course. Wohlin argues that introducing empirical software engineering will provide more opportunities to conduct empirical studies in student settings. However, he also mentions a need to balance educational and research objectives. Similar arguments are provided by Dillon [12], who states that successful observation of a phenomenon as part of an empirical study should not be an end in itself, and that students should have enough time to get familiar with the related ideas and concepts associated with the phenomenon. However, this shows the two different streams in using empirical instruments in teaching: On the one hand, students are educated such that they can serve as subjects in empirical studies (including all the risks as mentioned by Runeson [37]), and, on the other hand, empirical studies are used as teaching tools (as for instance done in economy for years, cf. [1, 33]). And it must not be questioned that empirical instruments provide a good basis to organize whole courses or individual sessions, e.g., [24, 26].

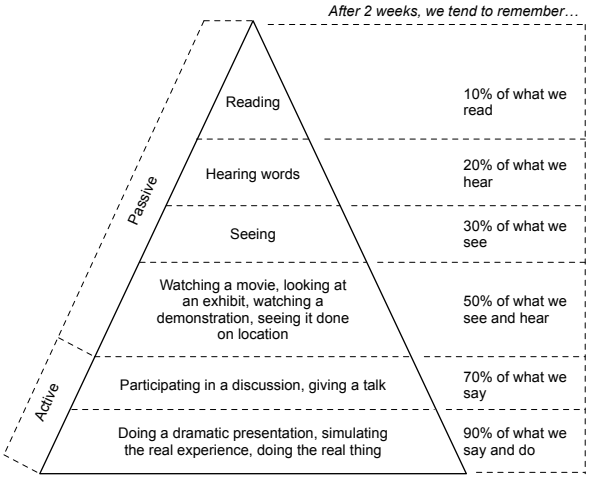

Figure 1: Dale's cone of learning (according to [10]).

However, these approaches aim at utilizing empirical instruments to *support* courses. Usually, students only get in touch with empirical instruments as subjects in an empirical inquiry, and they have to carry out tasks, e.g., in a controlled experiment, e.g., [15–17, 25, 26]. Teaching empirical software engineering as a subject, however, would require a self-contained course—or as Wohlin [43] mentioned: a self-contained course or as part of a course on research methods. In respect of Dale's *Cone of Learning* [10], such a course would need to cover the different levels of the learning cone (Figure 1). Yet, while the passive parts of the cone are easy to implement, addressing the active levels is way more challenging, since this requires the students to carry out actual research.

In [27], we proposed a teaching model to better align research with Master-level courses—mainly utilizing empirical instruments to re-organize exercise parts to bring students closer to real cases, but in a protected environment, which, inter alia, allows for simulation of critical or even failure situations [25, 26]. Applying this approach to several more method-focused courses, experience gathered so far was used to apply this approach to empirical software engineering. The paper at hand thus provides a new building block in software engineering education, which proposes an initially evaluated template for setting up courses on empirical software engineering.

Learning Goals	
G_1	*Learn the scientific way of work:* Students are introduced to the scientific method, learn the relevant terminology and concepts, and learn about the process of planning, conducting, and reporting scientific work.
G_2	*Learn to work with scientific literature:* Students learn how to find, read, and how to critically evaluate scientific literature. Students carry out reviews of real conference papers (training with given criteria), and students carry out group-based peer reviews of the essays written in the course.
G_3	*Obtain detailed knowledge about scientific methods:* Complementing the overview, students get detailed knowledge about selected scientific methods. The students are enabled to explain, discuss, and apply the chosen methods.
G_4	*Carry out a scientific study:* In small teams, students learn science by doing science. Studies are carried out by team setups comprising theory and practice teams; cross-cutting teams provide support, e.g., for reporting, data analysis, and data visualization.
G_5	*Train and improve communication and collaboration skills:* Students go through large parts of a scientific investigation and, thus, need to collaborate with other teams. Furthermore, they need to give presentations about their topics and they have to write "conference" papers (course essays) to report their findings.

Table 1: Summary of the course's learning goals.

3 Overall Course Design

This section presents the learning goals, the general organization model, the overall course design, the group setup and the topics handed out to the students. Furthermore, in the section, we explain how the different student groups form the team of experts throughout the course.

Learning Goals With the course contents, structure and the team setup presented, the course addresses the learning goals summarized in Table 1.

Empirical Methods A variety of empirical methods/techniques is subject to teaching. Before going into the details, we briefly summarize the methods selected for the course in Table 2. The instruments listed in Table 2 are of interest when setting up the actual "research work" for the students, since every method has certain constraints. For instance, while smaller experiments or (partial) literature studies are suitable for an educational setting, a real case study is difficult to implement (time and effort). The actual selection is further discussed in Section 3.3.

3.1 General Organization Model

To address the different learning goals, and, at the same time, to cover the variety of different empirical instruments, the course implements *expert teams* according to the general organization model as illustrated in Figure 2. For each empirical method, two

Instrument	Summary
Experiment	Experiments investigate effects of treatments under controlled conditions. They are rigorously designed and results constitute tests of a theory. Experiments can be, for instance, (semi-)formal, address multiple factors, and they can be conducted under lab conditions or in the field [45].
Case Study	Case studies aim to investigate a phenomenon in its natural context. They help answering explanatory questions, and they should be based on an articulated theory regarding the phenomenon of interest. Case studies can be implemented in a variety of setups, e.g., single case, multi-case, and longitudinal case studies [39].
Survey Research	A survey aims at collecting information from or about people to describe, compare or explain their knowledge, attitudes, and behavior [14]. Surveys can, for instance, be implemented as interview studies or as online questionnaire [29].
Simulation	Simulation refers to the use of a simulation model as an abstraction of a real system or process. Typical purposes for using such models are experimentation, increased understanding, prediction, or decision support. Simulations can, for instance, be carried out as people-based or computer simulations [31, 45].
Literature Study	A literature study aims at collecting reported evidence to (i) capture and structure a domain of interest, (ii) to aggregate available knowledge, and (iii) to synthesize generalized knowledge about the topic of interest. Literature studies in software engineering come as systematic review [19] or as systematic mapping study [36].

Table 2: Summary of the main empirical instruments (study types) to be considered.

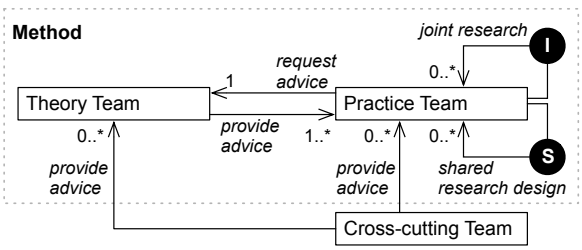

Figure 2: Overall organization model of the Scientific Methods course.

types of teams are built. A *theory team* is supposed to build competency about the actual method, e.g., what is the method about in detail, how to apply it, and how to report findings. *Practice teams* take over an actual research task to be implemented following a specific method, e.g., an experiment or a literature review. Theory teams then provide advice to the practice teams and monitor the implementation of the task, and practice teams request services from the theory team, e.g., feedback on the procedures. Furthermore, practice teams can be connected with each other. Figure 2 defines the two relationships "I" (Interface) for joint research, i.e., research on one topic but from different perspectives, and "S" (Shared) for an independently conducted research task, i.e., a shared research design is independently implemented by multiple teams. Finally, for specific topics that address cross-cutting concerns, like statistical analyses or data visualization, *cross-cutting teams* are established. These teams serve all theory and practice teams.

3.2 General Course Layout

Figure 3 illustrates the overall structure of the course *Scientific Methods* and shows how the different topics (Section 3.3) are aligned in the course.

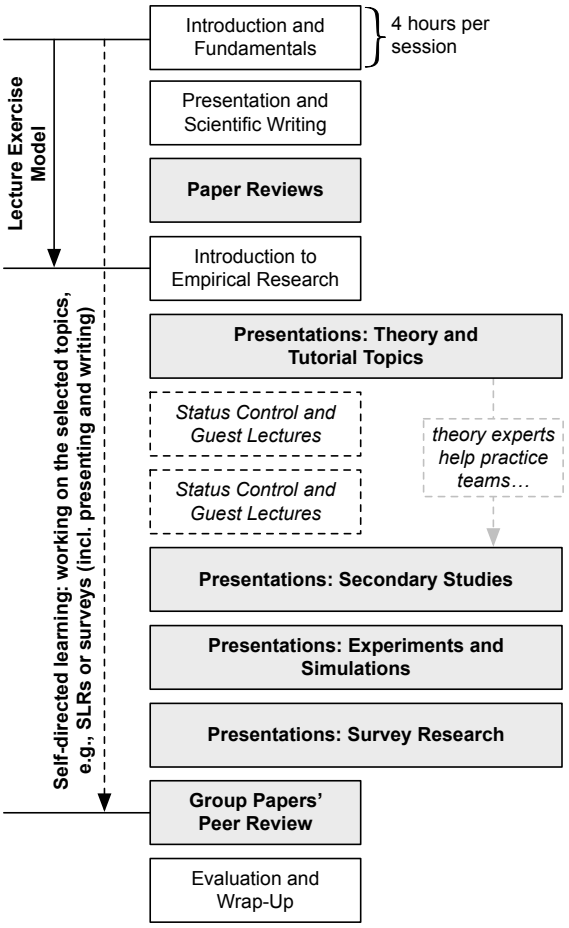

Figure 3: Overview of the overall structure of the Scientific Methods course.

The course starts with a general introduction to the topic, which covers the foundations of scientific work (session 1) and basic knowledge regarding reporting and presenting scientific work (session 2). In the first session, the different "Mini-Projects" are introduced, and students select their topics for the semester (the final selection from the topic pool is shown in Table 3). In the second session (as part of an introduction to publication processes) the review process is introduced. Based on this introduction, students are handed out an assignment in which they have to review 2 randomly selected papers following a review template (session 3; homework). In session 4, students get an introduction (or re-cap) on the basic maths of empirical research, e.g., hypothesis construction, statistical tests, and errors. In the first part of the course (4 weeks), students are introduced to the subject, basic elements of the work procedures are introduced, and students carry out first activities.

While the actual scientific methods (Table 2) were only presented as "teasers" in the first block, the second part of the course starts with detailed elaborations on these methods. The teams, which opted for theory topics (Table 3) present their respective methods. The presentations include an overview of the method, a description of how the method is applied (in general and illustrated by examples), and the presentations conclude with recommendations regarding the implementation for the practice teams. After the presentations, the theory teams switch their role and become "consultants" for the practice teams (cf. Figure 2).

The following five weeks are fully devoted to project work, i.e., the practice teams work on their topics. In this 5-weeks slot, two in-class sessions are scheduled in which the teams report the current project state. These sessions comprise guest lectures by researchers, who present their research and explain how it was conducted, and tutorials are implemented, such as implementing a survey as online questionnaire.

In the next slot, the outcomes of the respective projects are presented. In parallel, students started writing their essays, which have to be handed in the week after the last student group presentation. These essays are written as conference papers following the rules of a scientific conference, i.e., structure, page limits, and so forth. These papers are collected and distributed for peer-review among the groups.

The course layout from Figure 3 directly addresses the learning goals (Table 1): the first part addresses the learning goals G_1 and G_2, the second part addresses the learning goals G_3, G_4, and G_5, and the last part addresses G_2 again.

3.3 Topic Overview

The choice of topics for the course presented is influenced by (i) available topics from ongoing research, (ii) available options to replicate completed research, and (iii) a share of theoretical topics for students that

No.	Topic	Kind	Team		References	
			M	A	R	S
1	What is a Survey?	Theory	3	2	[20,29]	[4,13]
2	What is an Experiment?	Theory	3	3	[2,45]	[7,8]
3	What is a Systematic Review?	Theory	3	3	[19]	[11,42]
4	What is a Systematic Mapping Study?	Theory	3	3	[35,36]	[21,34,44]
5	What is a Simulation?	Theory	3	3	[31,45]	[28,32,41]
6	What is a Case Study?	(C) Theory	3	3	[38,39]	[9,40]
7	What are Threats to Validity?	(C) Theory	3	3	[45]	—
8	Introduction to R	(C) Tutorial	4	4	*self-search*	
9	Introduction to Data Visualization	(C) Tutorial	4	4	*self-search*	
10	Test approaches in Agile SW-development	Experiment	4	3	[45]	[15–17]
11	Perception of SE Semester Projects (Students)	Survey	4	4	[20,29]	—
12	Perception of SE Semester Projects (Teachers)	Survey	4	4	[20,29]	—
13	Quality Management in SPI	SLR	4	3	[19]	[11,18,42]
14	Industry exceptions on Testing Research (Group A)	Survey	4	4	[20,29]	—
15	Industry exceptions on Testing Research (Group B)	Survey	4	4	[20,29]	—
16	Success Factors in SPI	SMS	4	3	[19]	[11,21,42]
17	Agility as SPI Paradigm (Group A)	SLR	4	4	[19]	[11,21,42]
18	Agility as SPI Paradigm (Group B)	SLR	4	3	[19]	[11,21,42]
19	Comparison of Place Cell Models	Simulation	4	4	[31,45]	[28,32]
20	Comparison of Navigation Strategies	Simulation	4	4	[31,45]	[41]

Table 3: Overview of the topics in the Scientific Methods course including a classification (kind), team setup (M: max. team size, A: actual team size), and provided references (R: reference publications explaining the method, S: reference/input studies on this particular research project).

are reluctant towards working "in the wild". Table 3 gives a short overview by naming the topics, categorizing them, and providing information regarding maximum team size and references to respective research projects/publications if applicable. These topics represent the finally selected topics from a pool of about 30 proposals. As mentioned before, the list comprises a number of theory and tutorial topics. Groups having selected those topics did not carry out "real" research, but built the methodical competence and consulted the practice teams. That is, it was ensured that each practice team has a consultant team (in addition to the teacher) available.

The practice topics from Table 3 are selected from ongoing research (or from completed research that was identified worth replication). For these topics, existing research collaborations were triggered to identify *topic sponsors*. For instance, potential topic sponsors were asked: *Do you have ongoing research that we could contribute to?*, *Do you have research designs that we could use?*, *Do you have data that you would like to have a preliminary analysis for?* However, the conditions were made clear: (i) the topics must be manageable within 4 weeks, (ii) for secondary studies, a pre-digested dataset has to be delivered, (iii) sponsors must not expect a full and mature, i.e., publication-ready, result set, and (iv) sponsors should be willing to carry out quality assurance tasks and, if applicable, some consultancy, or even give a guest lecture on the respective research.

3.4 Team Structure and Collaboration

This section introduces the actual team setup of the initial implementation. Figure 4 illustrates the bird's-eyes perspective on the team setup and shows the relation of the theory teams and the practice teams.. The teams' numbers in Figure 4 correspond with the topic numbers from Table 3.

Due to the sponsors and the research they brought to the table, practice teams had three different types of projects: individual projects, interfaced projects, and shared projects. In interfaced projects (teams 11 and 12), students set up a study on the same subject, but had to take different perspectives and slightly different methods to be applied. Nevertheless, both teams needed coordination, notably concerning the questionnaire designs and the scheduling of interview slots. In shared projects, students either shared a study design (and applied it to different target groups, e.g., teams 14, 15) or implemented *independently* conducted research based on an identical task (e.g., teams 17, 18).

Survey Research Teams The survey research teams (Figure 5) worked on two tasks: one interfaced task and one shared task. The interfaced task means that

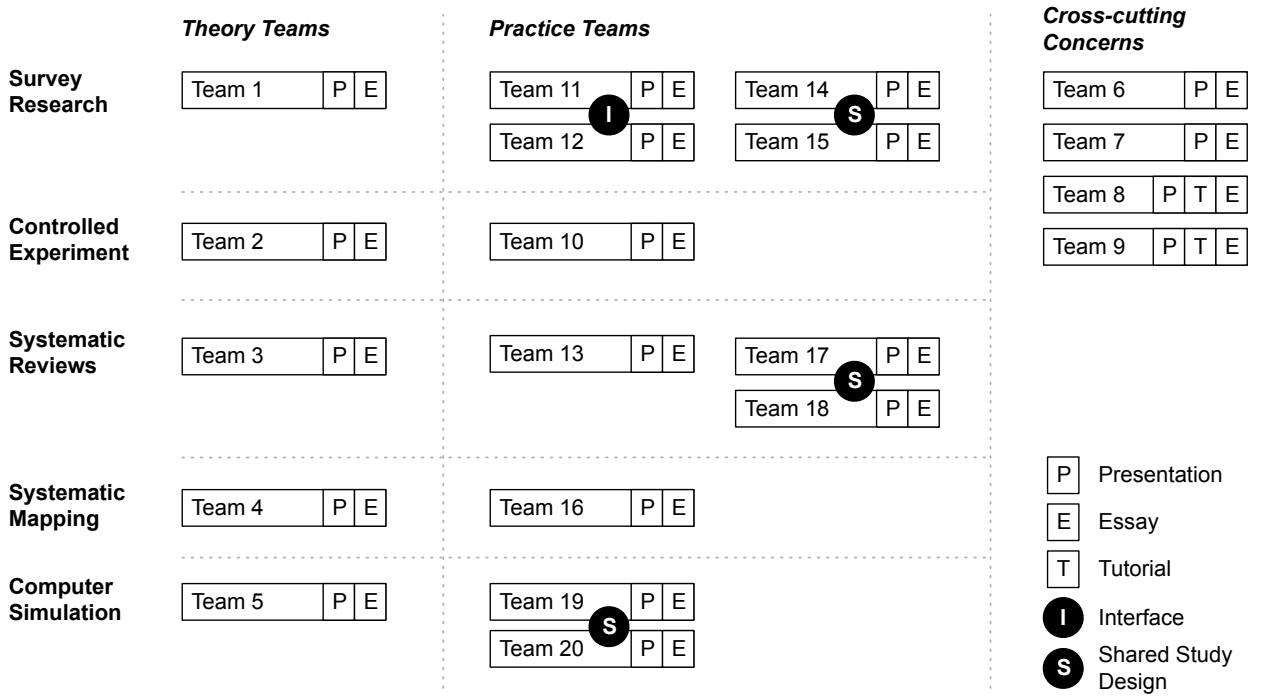

Figure 4: Overview of the teams structure: theory and practice teams, method-based team clusters, and teams addressing cross-cutting concerns. Deliverable types are explained in detail in Table 4.

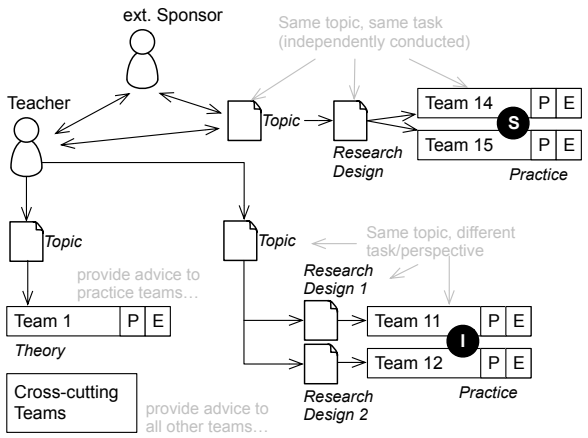

Figure 5: Group setup of the survey teams (theory, practice, cross-cutting).

both teams worked on related research designs derived from the shared topic: *Analyze the perception of the semester projects* from the perspective of the students and from the perspective of the teachers. For this, several individual and joint sessions were organized, inter alia, to elaborate shared questions of the respective questionnaires to allow for discussing the overall topic from different perspectives, e.g., students' vs. teachers' perspectives of project topics or group setups.

The shared task means that an external sponsor shared a research design, which was handed out to two groups. Both groups implemented the research designs, yet surveying different groups of which the contact information was provided by two local industry clusters. In this case, the two groups received a predefined research kit and had to implement this kit, i.e., organize and conduct interviews.

Systematic Review Teams Two systematic review (SLR) topics were selected from the topic pool. Since SLRs are time-consuming, especially the actual search and selection stages, both teams were provided with pre-digested datasets emerging from a systematic mapping study (scoping study: [21]) and two selected substudies [18, 22] thereof. For the SLRs, two external sponsors were acquired, who contributed to the topic and research design definition, provided pre-digested datasets, and supported the quality assurance of preliminary results.

The general organization follows the setup shown in Figure 5, yet, the shared study design followed a slightly different approach. Both teams 17 and 18 received the same research kit and were asked to carry out the same tasks in an independent manner. The purpose was to carry out the systematic review from two different groups to, eventually, demonstrate the expected difference in the results caused by personal decisions of the respective reviewers.

Cross-cutting Concerns The teams covering the cross-cutting concerns have a special role in this setup. In particular, every team has to report their results.

Deliverable Types	
R_1	*Review:* In the first individual assignment, students have to deliver reviews for two randomly selected conference papers (following a given template, about 1 page per review).
P	*Presentation:* Each team has to give a 15-minute presentation on its topic. The presentations are scheduled in topic slots as shown in Figure 3.
T	*Tutorial:* For the teams 8 and 9, the students have to prepare a 15-minute tutorial, which can be done in class as well as "offline".
E	*Essay:* Each team has to submit an up to 10-page essay in which the project is described. For the theory teams, the essay must comprise definitions, summaries about the application of a method based on further studies, check lists for the practice teams, and observations of the practice teams. The practice teams report their findings from their respective projects. The essay is developed in LaTeX following the latest ACM conference templates.
R_2	*Review:* Each team has to review two papers from other project teams. Other than R_1, this review is carried out as a group task.

Table 4: Summary of the expected deliverable types (related to the teams from Figure 4).

Since there were no case studies among the topic proposals[1], team 6 was asked to focus on (case) study reporting and to offer respective knowledge to the practice teams. The topics 7, 8, and 9 are true cross-cutting topics, i.e., all practice teams have to discuss threats to validity, have to carry out some sort of data analysis, and have to visualize their findings. Therefore, the cross-cutting concerns teams are (potentially) consulted by all the other teams.

3.5 Deliverables and Examination

Each team has to deliver a number of deliverables, which are summarized in Table 4.

4 Evaluation and Discussion

We report our experiences and lessons learned from the initial implementation of the course. Furthermore, we provide some discussion using the in-course feedback collected in two evaluation rounds.

4.1 Course Evaluation

The evaluation presented in this section is based on two evaluation rounds, which were carried out in the seventh session (mid-term) and the closing session (final). The evaluation was conducted using the questionnaire presented in Table 5.

[1] As this was the first time the course was run this way, case study research was excluded from the portfolio due to the expected effort of running a "true" case study. Also, only one experiment group was accepted. Yet, these methods will be included in upcoming course instances as soon as there is sufficient experience available regarding the options to integrate these methods properly.

Id	Question	Type
General Criteria		
GC_1	Please rate the course according to the following criteria: general cause complexity, course speed, cause content volume, and the appropriateness of the course in terms of ECTS	LI
GC_2	Please rate the following course components: lecture, exercise, relation to practice	LI
Free-form comments (1-minute paper)		
FF_1	Please name up to 5 points you considered good	Text
FF_2	Please name up to 5 points you considered bad and that need improvement	Text
FF_3	Anything else you want to say?	Text
Course-integrated Mini-Projects		
MP_1	What is your general opinion about the mini-projects?	0/1
MP_2	Please evaluate the statements: changed my view on science, improved learning experience, better understanding of concepts, helpful for later career, and built expertise to share.	LI
MP_3	If you had the choice, would you have more focus on mini-projects or more classic exercises?	LI
MP_4	Is there anything else you want to say?	Text
GC and MP Extension (final evaluation only)		
MP_5	Looking back, the mini-projects contributed to my learning experience.	LI
MP_6	Looking back, the team work within the mini-projects was good.	LI
MP_7	Looking back, the cross-team collaboration among the different project groups was good.	LI
GC_3	What is your major take-home asset from the course?	Text

Table 5: Questionnaire used for the mid-term evaluation (simplified version for space limitations; LI=Likert scale, 0/1=decision on a statement).

The questionnaire comprises quantitative as well ans qualitative questions: the general criteria (GC) serve the general analysis whether students consider the course fair[2]. The second part of the questionnaire comprises the *1-minute-paper* part (FF) in which students are asked for providing feedback to capture the current mood in the course and to support the course's improvement. Finally, in the third part, the perceived value of the course-integrated mini-projects (MP) is evaluated. The subsequent sections provide the quantitative and qualitative analysis of the mid-term feed-

[2] Note: These questions are kept stable since [27] in order to also validate the teaching model proposed; see also [24].

back, and present the evaluation of the mini-projects and the perceived value.

4.1.1 Standard Quantitative Evaluation

In total, 68 students are active in the course of which 39 students participated in the mid-term evaluation and 38 in the final evaluation respectively. The subsequent discussion is focused on the final evaluation, and results from the mid-term evaluation are presented, but only used for discussing changing perceptions over time.

The question GC_1 addresses the general rating of the course and whether the students consider the course appropriate. Figure 6 shows the absolute rating and shows that students consider the course's volume *high* to *very high* (19 out of 38), but at the same time, the majority of the students consider complexity and speed fair. In summary, 24 out of 38 students consider the appropriateness of the ECTS for this course *fair* to *absolutely appropriate*.

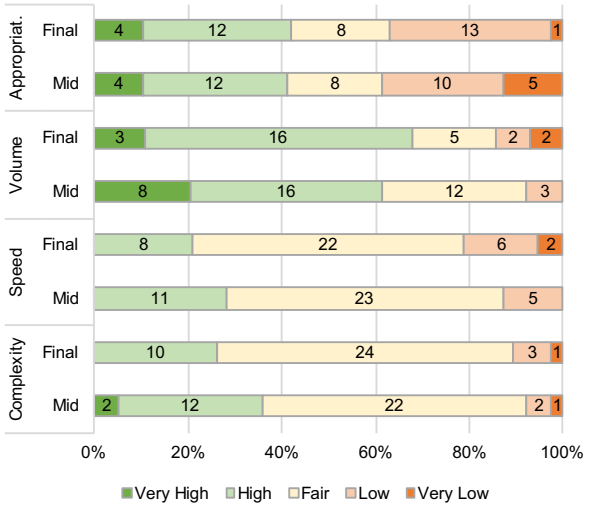

Figure 6: Evaluation of the general criteria part GC_1.

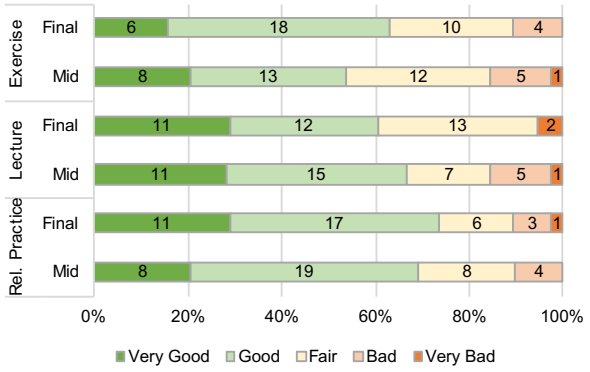

Figure 7: Evaluation of the general criteria part GC_2.

Question GC_2 aims at computing an overall grade for the course and considered the three components lecture, exercise, and the relation of the course to practice. Figure 7 shows the absolute mentions, and the figure shows that the majority of the students rate the course *good* to *very good*. The overall grades from the parts GC_1 and GC_2 are shown in Table 6 (with 1.00 as best and 5.00 as worst grade).

Criterion	Mid-Term			Final	
	MD	Avg		MD	Avg
Course Complexity	3	2.69	↘	3	2.87
Course Speed	3	2.85	↘	3	3.05
Content Volume	2	2.26	↘	2	2.58
Appropriateness	2	3.00	↗	4	2.87
Lecture	2	2.23	→	3	2.21
Exercise	2	2.44	↗	2	2.32
Relation to Practice	2	2.21	↗	2	2.11

Table 6: Overall rating of the course (MD: modal values; Avg: average ratings; mid-term: n=39, final: n=38; arrows indicate the trend from the mid-term to the final evaluation).

4.1.2 Qualitative Standard Evaluation

For the qualitative evaluation, the *1-minute-paper* part of the questionnaire is used (Table 5, questions FF_x). In particular, for question FF_1, 35/32 (mid/final) students provided (positive) feedback; for FF_2, 32/29 students provided feedback regarding negative points/aspects to be improved, and, finally, for FF_3, 10/6 students provided further comments. In total, we received about 130 statements for the mid-term evaluation and about 125 comments in the final evaluation, which we group and analyze in the following. The statements of the students were categorized based on keywords; the threshold for a category was set to three mentions.

Category	Mid-Term		Final	
	Pro	Con	Pro	Con
Group work, feedback, communication	11		7	
Content and understanding	8		9	
Mini-projects	8	5	11	4
Work pattern	8	4	10	1
Content/material volume		12		4
Class size		3		3
Relevance		4		4
Guest lectures*	3	1	6	5
Volume for ECTS*		6		4

Table 7: Categorized and condensed qualitative feedback (free-form text questions) Categories marked with '*' were added during the final evaluation.

Table 7 provides the condensed qualitative feedback in nine categories. Group work, in particular, the involvement of students, the communication and quick feedback cycles were considered positive. Also, the content collection and the understandability of

the content was considered positive; whereas the volume of the content was considered critical (comment, mid-term: "*maybe 20% less would be more manageable.*"). The chosen way of work, i.e., expert teams in combination with the mini-projects, shows an indifferent picture. On the one hand, students appreciate this approach as it allows for focusing, continuous work on one subject, and building expertise in specific methods. However, on the other hand, students consider certain aspects critical. For instance, the focus brought by the mini-projects allows for obtaining detailed knowledge on one method, yet, the students are concerned about the other approaches, which were not in their respective scope. One argument presented was a non-optimal synchronization among the expert teams. Arguments presented in favor of the work model were real research and cross-team collaboration, arguments against this work model regarded a late availability of the research kits and the size of the projects. Few students also suggested to reduce the mini-projects to "normal" assignments that would allow for covering more topics/methods: "*Mini-projects are basically good, but more variety would be nice.*").

Another critical aspect of the course was the amount of reading material provided. While two students explicitly mentioned "*learning to analyze scientific papers*"—and later on also "*write*"—positive, six (mid-term) students considered the material to read and analyze too much, yet, in the final evaluation, this aspect was not mentioned anymore. In the mid-term evaluation, six students stated the number of ECTS points for this course to small; in the final evaluation, four students (still) think the the amount of credit points inappropriate. Furthermore, students from other study programs than *Software Engineering, MSc* were enrolled. Hence, the relevance for their specific education lines was questioned. Also, three students explicitly questioned the relevance to their current studies and future activities. Finally, the course had almost 70 students enrolled, which is almost thrice the class size for which the pattern was applied so far. And this class size was mentioned critical, especially the "*crowded class room*".

4.1.3 Evaluation of the Course-integrated Mini-Projects

For the question MP_1 (*What is your general opinion about the mini-projects?*), 36 students mentioned that they like the mini-project approach, and two students mentioned that they would prefer the classic lecture-exercise model to the mini-project approach. Figure 8 further shows that, eventually, five out of 38 students tend towards applying more classic teaching elements, i.e., the classic lecture-exercise model (question MP_3). Yet, 11 students would prefer putting even more focus on the mini-projects.

Figure 9 shows the absolute mentions for MP_2. The figure shows that the mini-projects are considered valuable to improve understanding of concepts and

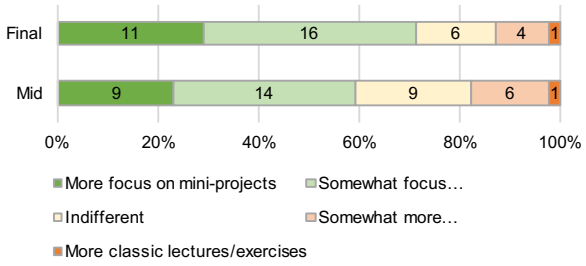

Figure 8: Perception of the mini-project approach compared to the classic lecture-exercise model (mid-term: n=39; final: n=38).

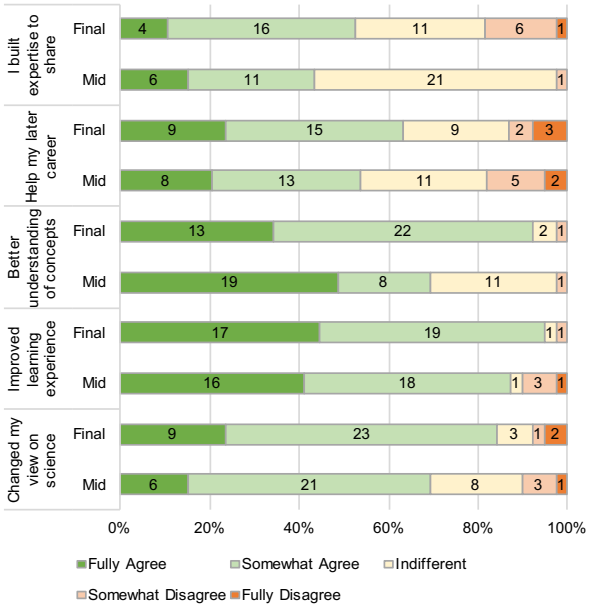

Figure 9: Evaluation (absolute) of the general criteria section MP_2 (mid-term: n=39; final: n=38).

to improve the general learning experience. Since the mini-projects aim at building expert teams, i.e., teams that build a specific expertise to share with other teams, it is important to see the students' perspective regarding this goal. The figure shows that, finally, 20 out of 38 students think they have built a respective expertise to share, yet, 11 students are *indifferent*. Compared to the numbers from the mid-term evaluation, the data shows the students evaluating their gained knowledge and experience better to the end of the course. Another point of interest is the students' perception of scientific work. Quite often, students have little contact with scientific work until the late stages of their studies, which makes scientific work somewhat abstract and hard to align with the students' day-to-day work[3]. Thus, it is of certain in-

[3] In [26], we already mentioned that a strong focus on projects might influence the willingness of students to accept and apply methods/techniques not directly addressing the actual project goals.

terest to learn whether the "practical" scientific work changed the students perception and if they see positive impact for their later professional development. Figure 9 shows 32 students (fully) agreeing that the course changed their view, and 24 also see impact on their later career—both increased toward the course's end. A statement from the free-text form (mid-term evaluation) provides a good summary: *"in my opinion lecturing scientific method without working with the methods it's only knowing about the methods, not learning them."*

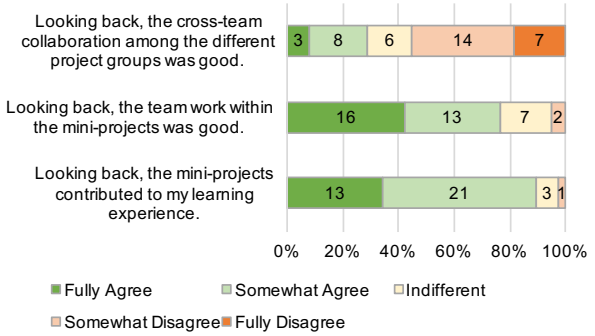

Figure 10: Final evaluation of the mini-project approach by the students (n=38).

Finally, Figure 10 provides a reflection. The general perception of the mini-projects and the team work is positive. However, the students considered the cross-team collaboration not optimal, which shows room for improvement. Studying the feedbacks shows that some teams just "disappeared" and the other teams could not interact anymore, yet, this requires further analysis of the evaluation data.

5 Conclusion

In this paper, we presented a course design to teach empirical software engineering, which follows the principle *learn scientific work by doing scientific work*. The concept presented aims at implementing such a course with larger classes, and, in order to manage a large number of students, utilizes *expert teams* to allow for specialization and fostering cross-team collaboration. Expert teams are supposed to build a specialization beyond the base knowledge and to bring in this specialized knowledge in a cross-team collaboration, e.g., a (theoretical) expertise on a specific method is used to consult practice teams that apply this method and, vice versa, practice teams that report experience to a theory team of how the method "feels" like in practice.

A reference implementation was run at the University of Southern Denmark in the fall semester 2016 with about 70 students enrolled. The students formed 20 teams carrying out mini-projects, which are of theoretical, practical, or cross-cutting nature, and that addressed a variety of different empirical methods. All practical projects comprised "real" research tasks sponsored by internal or external researchers, and that come from either ongoing research or from completed research that was considered worth replication. An evaluation by the students shows the course and the work pattern considered appropriate. In particular, students valued the collaborative work on real research tasks. Yet, they were also concerned about the effectiveness of the work pattern, in particular, students are concerned about potentially missing insights to further methods. However, the initial evaluation shows a generally positive attitude towards the expert team and mini-project approach.

However, since it was the first time that (i) this course was run this way and (ii) the used teaching model [27] was yet not implemented at this scale, the initial implementation revealed potential for improvements. For instance, the class size was considered critical, whereas the heterogeneity of the class needs to be considered, too. For future implementations, the course should be limited to one study program only. This would allow for better tailoring the course for the respective audience. Due to the explorative nature of the reported course instance, no teaching assistants were involved, which resulted in a dramatically high workload. For future instances, teaching assistants should be involved to reduce the workload, e.g., to speed up organization processes like topic sponsor acquisition or research kit preparation. Furthermore, the volume of the course contents needs adjustment.

Finally, several aspects await an in-depth analysis, e.g., analysis of the work load and the work distribution. That is, is the topic selection and task assignment fair? Is the cross-team collaboration working as expected? Future work will therefore focus on analyzing the communication within the course (based on approx. 650 emails, more than 50 meetings in total, paper and presentation reviews, and confidential written evaluation of the cross-team collaboration by the students). Also, an independent quality assurance of the students' deliverables beyond the examination (e.g., supplemental material like extra article sources, or quality and completeness of research data) is an option to better understand the appropriateness of the tasks and the suitability of the topic composition, and helps improving the course's goal definitions.

Acknowledgement

We owe special thanks to all the students actively participated in the course and who accepted the challenge to be the "guinea pigs", and that jumped into cold water and conducted real research. We also want to thank the different topic sponsors, who shared their research topics and (ongoing) work in this course.

At SDU's engineering faculty, project-based learning is foundational principle, which continuously puts students in project situations and leaves little space to reflect on topics such as scientific methods, since those do not directly and immediately contribute to the product development.

References

[1] S. Ball, T. Emerson, J. Lewis, and J. T. Swarthout. Classroom experiments. Available from http://serc.carleton.edu/sp/library/experiments/index.html, 2012.

[2] V. Basili, R. Selby, and D. Hutchens. Experimentation in software engineering. *Trans. on Software Engineering*, 12(7):733–743, 1986.

[3] V. R. Basili. The experimental paradigm in software engineering. In *Proceedings of the International Workshop on Experimental Software Engineering Issues: Critical Assessment and Future Directions*, volume 706 of *LNCS*, pages 3–12. Springer, 1993.

[4] J. D. Blackburn, G. D. Scudder, and L. N. V. Wassenhove. Improving speed and productivity of software development: a global survey of software developers. *Trans. on Software Engineering*, 22(12):875–885, Dec 1996.

[5] M. Broy and M. Kuhrmann. *Projektorganisation und Management im Software Engineering*. Xpert.press. Springer, 2013.

[6] B. Brügge, S. Krusche, and L. Alperowitz. Software engineering project courses with industrial clients. *Trans. Comput. Educ.*, 15(4):17:1–17:31, Dec. 2015.

[7] R. O. Chaves, C. G. von Wangenheim, J. C. C. Furtado, S. R. B. Oliveira, A. Santos, and E. L. Favero. Experimental evaluation of a serious game for teaching software process modeling. *Trans. on Education*, 58(4):289–296, Nov 2015.

[8] M. Ciolkowski, C. Differding, O. Laitenberger, and J. Münch. Empirical investigation of perspective-based reading: A replicated experiment. Technical Report 13/97, International Software Engineering Research Network (ISERN), 1997.

[9] D. S. Cruzes, N. B. Moe, and T. Dybå. Communication between developers and testers in distributed continuous agile testing. In *International Conference on Global Software Engineering*, pages 59–68. IEEE, Aug 2016.

[10] E. Dale. *Audiovisual methods in teaching*. Dryden Press, 3 edition, 1969.

[11] K. Dikert, M. Paasivaara, and C. Lassenius. Challenges and success factors for large-scale agile transformations. *J. Syst. Softw.*, 119(C):87–108, Sept. 2016.

[12] J. Dillon. A Review of the Research on Practical Work in School Science. Technical report, King's College, 2008.

[13] D. M. Fernández and S. Wagner. Naming the pain in requirements engineering: A design for a global family of surveys and first results from germany. *Inf. Softw. Technol.*, 57:616–643, 2015.

[14] A. Fink. *The Survey Handbook*. Sage Publications Inc., 2 edition, 2002.

[15] D. Fucci and B. Turhan. A replicated experiment on the effectiveness of test-first development. In *International Symposium on Empirical Software Engineering and Measurement*, pages 103–112. IEEE, Oct 2013.

[16] D. Fucci, B. Turhan, and M. Oivo. Impact of process conformance on the effects of test-driven development. In *International Symposium on Empirical Software Engineering and Measurement*, pages 10:1–10:10. ACM, 2014.

[17] D. Fucci, B. Turhan, and M. Oivo. On the effects of programming and testing skills on external quality and productivity in a test-driven development context. In *International Conference on Evaluation and Assessment in Software Engineering*, pages 25:1–25:6. ACM, 2015.

[18] J. W. Jacobson, M. Kuhrmann, J. Münch, P. Diebold, and M. Felderer. On the role of software quality management in software process improvement. In *International Conference on Product-Focused Software Process Improvement*. Springer, Nov 2016.

[19] B. A. Kitchenham, D. Budgen, and P. Brereton. *Evidence-Based Software Engineering and Systematic Reviews*. CRC Press, 2015.

[20] B. A. Kitchenham and S. L. Pfleeger. *Personal Opinion Surveys*, pages 63–92. Springer London, London, 2008.

[21] M. Kuhrmann, P. Diebold, and J. Münch. Software process improvement: A systematic mapping study on the state of the art. *PeerJ Computer Science*, 2(1):1–38, 2016.

[22] M. Kuhrmann, P. Diebold, J. Münch, and P. Tell. How does software process improvement address global software engineering? In *International Conference on Global Software Engineering*, pages 89–98. IEEE, Aug 2016.

[23] M. Kuhrmann, D. M. Fernández, and A. Knapp. Who cares about software process modelling? a first investigation about the perceived value of process engineering and process consumption. In *International Conference on Product-Focused Software Process Improvement*, volume 7983 of *LNCS*, pages 138–152. Springer, 2013.

[24] M. Kuhrmann, D. M. Fernández, and J. Münch. Teaching software process modeling. In *International Conference on Software Engineering*, pages 1138–1147, 2013.

[25] M. Kuhrmann and J. Münch. Distributed software development with one hand tied behind the back: A course unit to experience the role of communication in gsd. In *1st Workshop on Global Software Engineering Education (in conjunction with ICGSE'2016)*. IEEE, 2016.

[26] M. Kuhrmann and J. Münch. When teams go crazy: An environment to experience group dynamics in software project management courses. In *International Conference on Software Engineering*, ICSE, pages 412–421. ACM, May 2016.

[27] Kuhrmann, M. A practical approach to align research with master's level courses. In *International Conference on Computational Science and Engineering*. IEEE, 2012.

[28] T. Kulvicius, M. Tamosiunaite, J. Ainge, P. Dudchenko, and F. Wörgötter. Odor supported place cell model and goal navigation in rodents. *Journal of Computational Neuroscience*, 25(3):481–500, December 2008.

[29] J. Linåker, S. M. Sulaman, R. M. de Mello, and M. Höst. Guidelines for conducting surveys in software engineering. Technical report, Lund University, January 2015.

[30] Münch, J., Pfahl, D., and Rus, I. Virtual software engineering laboratories in support of trade-off analyses. *International Software Quality Journal*, 13(4), 2005.

[31] J. J. Nutaro. *Building Software for Simulation: Theory and Algorithms, with Applications in C++*. Number ISBN: 978-0-470-41469-9. John Wiley & Sons, Ltd., 2010.

[32] J. O'Keefe and N. Burgess. Geometric determinants of the place fields of hippocampal neurons. *Nature*, 381:425–428, May 1996.

[33] J. Parker. *Using laboratory experiments to teach introductory economics*. Working paper, Reed College, http://academic.reed.edu/economics/parker/ExpBook95.pdf, accessed 2014-10-23.

[34] N. Paternoster, C. Giardino, M. Unterkalmsteiner, T. Gorschek, and P. Abrahamsson. Software development in startup companies: A systematic mapping study. *Inf. Softw. Technol.*, 56(10):1200–1218, Oct. 2014.

[35] K. Petersen, R. Feldt, S. Mujtaba, and M. Mattson. Systematic mapping studies in software engineering. In *International Conference on Evaluation and Assessment in Software Engineering*, pages 68–77. ACM, 2008.

[36] K. Petersen, S. Vakkalanka, and L. Kuzniarz. Guidelines for conducting systematic mapping studies in software engineering: An update. *Inf. Softw. Technol.*, 64:1–18, August 2015.

[37] P. Runeson. Using students as experiment subjects–an analysis on graduate and freshmen student data. In *International Conference on Empirical Assessment in Software Engineering*, pages 95–102, 2003.

[38] P. Runeson and M. Höst. Guidelines for conducting and reporting Case Study Research in Software Engineering. *Empirical Software Engineering*, 14(2):131–164, 2009.

[39] P. Runeson, M. Höst, A. Rainer, and B. Regnell. *Case Study Research in Software Engineering: Guidelines and Examples*. John Wiley & Sons, 2012.

[40] A. Sarma, X. Chen, S. Kuttal, L. Dabbish, and Z. Wang. Hiring in the global stage: Profiles of online contributions. In *International Conference on Global Software Engineering*, pages 1–10. IEEE, Aug 2016.

[41] M. Tamosiunaite, J. Ainge, T. Kulvicius, B. Porr, P. Dudchenko, and F. Wörgötter. Path-finding in real and simulated rats: assessing the influence of path characteristics on navigation learning. *Journal of Computational Neuroscience*, 25(3):562–582, 2008.

[42] G. Theocharis, M. Kuhrmann, J. Münch, and P. Diebold. Is Water-Scrum-Fall reality? On the use of agile and traditional development practices. In *International Conference on Product Focused Software Development and Process Improvement*, volume 9459 of *LNCS*, pages 149–166. Springer, Dec 2015.

[43] C. Wohlin. Empirical software engineering: Teaching methods and conducting studies. In *Proceedings of the International Workshop on Empirical Software Engineering Issues: Critical Assessment and Future Directions*, volume 4336 of *LNCS*, pages 135–142. Springer, 2007.

[44] C. Wohlin, P. Runeson, P. A. da Mota Silveira Neto, E. Engström, I. do Carmo Machado, and E. S. de Almeida. On the reliability of mapping studies in software engineering. *J. Syst. Softw.*, 86(10):2594–2610, 2013.

[45] Wohlin, C., Runeson, P., Höst, M., Ohlsson, M. C., Regnell, B., and Wesslén, A. *Experimentation in Software Engineering*. Springer, 2012.