DOI: http://dx.doi.org/10.1145/2889160.2889194

When Teams Go Crazy: An Environment to Experience Group Dynamics in Software Project Management Courses

Marco Kuhrmann
University of Southern Denmark
Mærsk Mc-Kinney Møller Institute
Odense, Denmark
kuhrmann@acm.org

Jürgen Münch
University of Helsinki
and Reutlingen University
Helsinki, Finland
juergen.muench@cs.helsinki.fi

ABSTRACT

Software development consists to a large extend of human-based processes with continuously increasing demands regarding interdisciplinary team work. Understanding the dynamics of software teams can be seen as highly important to successful project execution. Hence, for future project managers, knowledge about non-technical processes in teams is significant. In this paper, we present a course unit that provides an environment in which students can learn and experience the impact of group dynamics on project performance and quality. The course unit uses the Tuckman model as theoretical framework, and borrows from controlled experiments to organize and implement its practical parts in which students then experience the effects of, e.g., time pressure, resource bottlenecks, staff turnover, loss of key personnel, and other stress factors. We provide a detailed design of the course unit to allow for implementation in further software project management courses. Furthermore, we provide experiences obtained from two instances of this unit conducted in Munich and Karlskrona with 36 graduate students. We observed students building awareness of stress factors and developing counter measures to reduce impact of those factors. Moreover, students experienced what problems occur when teams work under stress and how to form a performing team despite exceptional situations.

Categories and Subject Descriptors

D.2.9 [**Software Engineering Management**]: Life cycle, Productivity, Software quality assurance

Keywords

Software Project Management, Experimentation, Group Dynamics, Tuckman Model, Agile

1. INTRODUCTION

Software Engineering is an interdisciplinary field and thus requires educating students in a way that enables them to efficiently and effectively work in interdisciplinary teams. Especially in agile software development, the *team* has become that unit ensuring a project's success by applying self-organization, fast communication, and close collaboration [18]. Therefore, it is essential to not only teach students the Software Engineering basics, such as software design, implementation, integration, or test, but also to build awareness of how software is developed in teams, and what fosters and what compromises software projects beyond the technical aspects. A number of studies analyze for instance the impact of team structure [31], team members' personality [5, 14], or soft factors in general [27] on the team performance. Team performance, among other things, influences cost and quality of the product under development [12], and project managers must also consider the relation of teams and project risks [15, 29]. That makes understanding of how teams are formed and how they work (including all the "bad" things that can happen) an important topic in Software Engineering education.

However, project management is hard to teach: The basic knowledge is taught in class and then students practice selected project management techniques in exercises, labs, or smaller projects. Yet, even though considered of high practical value, those projects also have the goal to deliver some software to (external) clients, i.e., there is a certain pressure to deliver code, which builds the common ground for the teams, but also shifts the focus to coding and away from other activities, such as collaboration, documentation, or learning from failures. Hence, students have little opportunities to experience the real stress factors that make project managers suffer; or even make projects fail. This makes software project management as a subject fairly abstract, even though teachers quite often have extensive knowledge to share.

Problem Statement & Objective. Project management as part of Software Engineering education is often taught on a fairly abstract level delivering fundamental knowledge about, e.g., organization and planning activities, estimation, and controlling. The team as the vital element is introduced and different theories, models, and strategies of how to set up and work in teams are taught. Yet, students usually have few options to experience how those theories and models manifest in practice.

Our objective is to provide an environment in which the software development activities move to the background and students can focus on experiencing different situations in self-organizing teams from a project management perspective. Therefore, we want the students to experience different stress factors, settings in which people explicitly attack a team and how to handle such situations under stress, and we want to provide the students with the opportunity to experience (close-to-)failure situations.

Contribution. The present paper provides a detailed description of a course unit to be included in software project management courses. The course unit uses the Tuckman model [28] as theoretical framework and comprises a practical part that is organized according to the structure of controlled experiments [30]. The overall goal of the course unit is to provide students with an environment in which they can experience stressful situations. The presented unit was conducted twice at two different universities, and was considered a valuable exercise. Different quantitative as well as qualitative analyses (e.g., video analysis) show this unit a valuable learning experience for the students.

Outline. The remainder of the paper is structured as follows: In Sect. 2, we discuss related work. Section 3 provides context information about the overall course structure and the course unit design in particular. Section 4 presents the analysis, discussion, and a summary of lessons learned. We conclude the paper in Sect. 5.

2. RELATED WORK

Basili et al. [3] were among the first to present a framework and a process for experimentation for Software Engineering. Experimentation was mainly used for research purposes in Software Engineering (with all challenges and risks as discussed by Runeson [26]), but got only little appreciation as a teaching tool. Nonetheless, experimentation as a means to improve teaching has proved successful over the years in many disciplines. For instance, Parker [24] mentions experiments became widespread teaching tools in economics in the 1990s. Nowadays, many economists use experiments as educational tools and mention several benefits, e.g., they are distinctive and more participative, and students are likely to remember lessons associated with them. Finally, Parker mentions experiments are considered fun. Another important source for classroom experiments is the SERC Portal for Pedagogy in Action, created by Ball et al. [2]. The repository includes a comprehensive list of experiments from different disciplines that can be used for replication in classroom settings. In addition, it contains references to scientific studies that provide empirical evidence about the expected positive effects of experiments as teaching tools.

However, an important conclusion Dillon [10] draws from his overview of advantages and disadvantages of experiments is that successful observation of a phenomenon as part of an empirical study should not be an end in itself. Rather, students should have enough time to get familiar with the related ideas and concepts associated with the phenomenon. This leads to a discussion on the suitability of experiments in teaching. In order to make experiments a useful tool, Parker [24] recommends considering three criteria: (i) the experiment must be aligned with the central topic of the course, (ii) the concept to be taught through the experiment should not be easily understood without the experiment or already obvious, and (iii) students need to be able to quickly learn the necessary prerequisites for participating in the experiment. Finally, as we mentioned previously in [19], teachers must not be ignored, as conducting experiments in classroom settings generates (extra) effort.

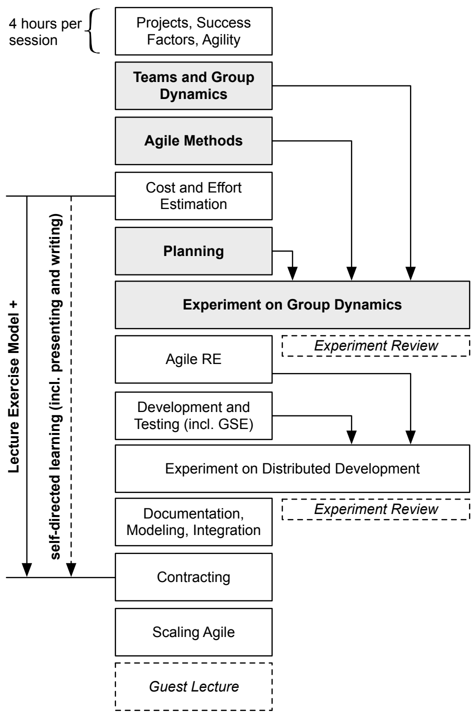
4 hours per session
Projects, Success Factors, Agility
Teams and Group Dynamics
Agile Methods
Cost and Effort Estimation
Planning
Experiment on Group Dynamics
Agile RE
Experiment Review
Development and Testing (incl. GSE)
Experiment on Distributed Development
Documentation, Modeling, Integration
Experiment Review
Contracting
Scaling Agile
Guest Lecture
Lecture Exercise Model +
self-directed learning (incl. presenting and writing)

Figure 1: Basic organization of the APM course.

Experimentation in Software Engineering education is usually used at the level of engineering processes, i.e., quality assurance and test [13, 16], Global Software Engineering (GSE; [17, 25]), process modeling [20, 22], or Software Engineering in general, e.g., [4, 9, 23]. Finally, newer approaches integrate (continuous) experimentation and close collaboration between academia and industry with the principles of Lean software and product development, e.g., [11, 21]. The present paper extends the body of knowledge with an experiment-based course unit in the field of software project management. We provide a course unit design and results from two instances focusing on self-organizing teams that allows students exploring and experiencing effects of group dynamics, which occur oftentimes in (increasingly agile organized) high-performance project teams.

3. GENERAL COURSE UNIT DESIGN

In this section, we provide the design of the course unit on group dynamics. We, provide an overview of the overall course context in Sect. 3.1. Learning goals are presented in Sect. 3.2. The theoretical model and the instrument to implement the practical parts are presented in Sect. 3.3 and Sect. 3.4 respectively.

3.1 Course Context

We present the overall design of the course "Agile Project Management & Software Development" (APM), which follows the pattern presented in [19, 20]. Figure 1 illustrates the general course organization and the course's content. The course consists of 4-hour sessions and comprises three phases: In phase 1, the scene is set, topics are introduced, and students are assigned their "special topics", which they have to prepare for phase 2. In the second phase, the course pattern changes: instead of "classic" lectures and exercises, the sessions follow a workshop model, whereas the workshops are composed of lecture- and exercise parts to which students actively contribute by presenting their "special topics", and the workshops also contain creativity tasks and discussion rounds. Furthermore, in this phase, selected sessions are devoted to more comprehensive exercises, such as the unit presented in the paper at hand. The third phase deals with wrapping up the course, provides time for guest lectures, and allows for preparing the exams.

3.2 Course Unit Learning Goals

As part of an advanced course on project management, the goal of this course unit is to enable students understanding the impact of group dynamics on project management, especially, team performance and result quality (Figure 1 shows the other course units providing required input and context). In particular, we define the following learning goals:

Learning Goal 1: *How to come to a working team quickly?* This learning goal addresses the students' ability to quickly come together, develop and implement work strategies, to set priorities, and to figure out what does (not) work.

Learning Goal 2: *What is the impact of staff turnover?* Students are randomly selected for teams and team setups are changed over time. Hence, students have to find themselves a place in a new team and to bring in their knowledge and experiences gained in another team, and students are forced to detect changes and re-organize their work pattern accordingly *and* quickly.

Learning Goal 3: *What stress factors can impact team performance and quality, and how to handle them?* Students are exposed to different stress factors—caused by the unit setup itself (e.g., time or lacking resources) or provoked by the supervisors (e.g., enforced staff turnover). Students shall experience work under heavy stress and, later, link their experiences to performance data to see the direct impact. Students shall learn how (not) to act in such situations.

3.3 Theory: The Tuckman-Model

The theoretical model taught to the students to explain group dynamics is the Tuckman model [28], which was widely tested since its presentation in the 1960s. The Tuckman model describes the basic processes of group dynamics in teams in five major phases *forming*, *storming*, *norming*, and *performing* and, finally, *adjourning* (Figure 2).

These phases can be found in every project team (more general in every group). Moreover, these phases do not only happen during the initial formation of new teams (learning goal 1), but also when new people join an already established team (learning goal 2). Project managers should therefore have some knowledge about such models, as for instance too high fluctuation in the project team seriously affects performance—in the worst case it can bring a team to a full stop, as the team is busy with fighting conflicts (re-forming and storming phases; learning goal 3).

3.4 Practice: A Controlled Experiment

In order the achieve the aforementioned learning goals, we organized the practical parts following the structure of *controlled experiments* [30] to ensure proper organization of the different treatments. As shown in Figure 1, the experiment session is carried out in a 4-hour block, which is specifically

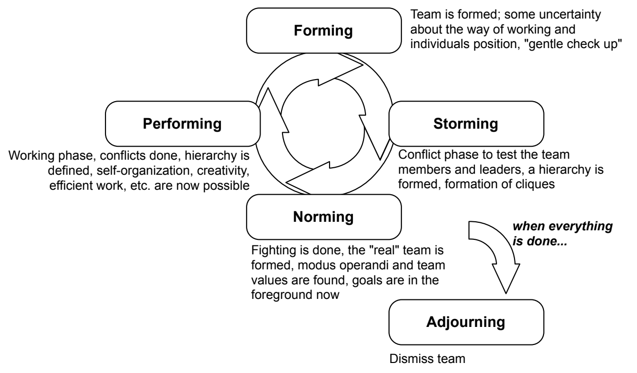
Forming
Team is formed; some uncertainty about the way of working and individuals position, "gentle check up"
Performing
Working phase, conflicts done, hierarchy is defined, self-organization, creativity, efficient work, etc. are now possible
Storming
Conflict phase to test the team members and leaders, a hierarchy is formed, formation of cliques
Norming
Fighting is done, the "real" team is formed, modus operandi and team values are found, goals are in the foreground now
when everything is done...
Adjourning
Dismiss team

Figure 2: Overview of the Tuckman model on processes in team formation (based on [28]).

prepared in class. The experiment's overall organization is illustrated in Figure 3. In subsequent sections, we provide details on the setup. Note: The purpose of this experiment is not to conduct a scientific investigation rather than reusing a proven structure with regards to (general) organization. Data collection does not aim at gaining new scientific insight. Its purpose is to support the learning goals by showing the effects of different treatments to the students.

3.4.1 Experiment Description and Execution

In order to focus on the effects group dynamics have on performance and quality, we defined a very simple task, which was introduced to the students as follows:

Task: *"The following items[1] need to be sorted by color. You can decide yourselves in which way you work. The only thing that counts is that your team sorts as many items in 2, 4, or 8 minutes as you can. Document your outcomes as a table containing: type, color, number, and so on. Your team can keep and eat all correctly sorted items after the* ***last*** *run."*

Data Collection and Analysis In this task, we collected quantitative as well as qualitative data. Quantitative data was collected using task-specific *controlling sheets* of which two per run and per team were created (Figure 3). This data was used as performance and quality measure (Table 1) to present students with an evaluation of their performance, and to also visualize the actual impact of group dynamics that students experienced[2].

The qualitative data was used to analyze if students could detect all stress factors to which they were exposed (Table 3). Furthermore, the qualitative data—especially the video data—was used to support the reflection and discussion of the lessons learned.

Before a run started, teams were provided with the first controlling sheet that they had to fill during the run. When

[1]As items to sort, we opted for *M&M's* for the following reasons: There are different types available, all types share the same color palette, the type "crisp" is easy to differentiate from type "choc", but hard to differentiate from type "nuts", such that students really must pay attention. Any other items fulfilling these characteristics are feasible for this experiment.
[2]The used Tuckman model allows for predicting of how the performance curves should look like, and students can reflect on their self-generated data and link those to their experience rather than just analyzing external data.

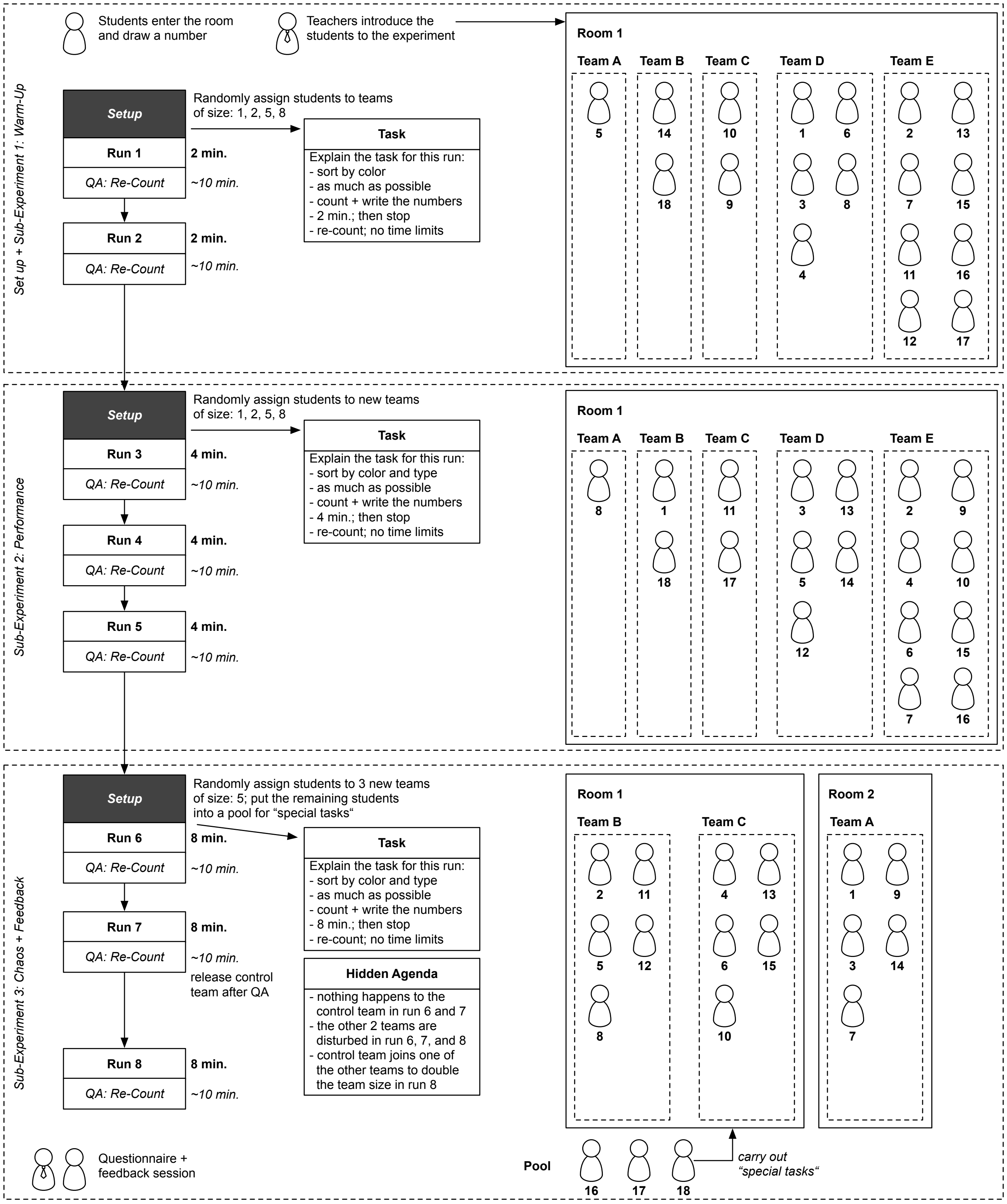
Students enter the room and draw a number
Teachers introduce the students to the experiment
Set up + Sub-Experiment 1: Warm-Up
Setup
Randomly assign students to teams of size: 1, 2, 5, 8
Run 1
2 min.
QA: Re-Count
~10 min.
Run 2
2 min.
QA: Re-Count
~10 min.
Task
Explain the task for this run:
- sort by color
- as much as possible
- count + write the numbers
- 2 min.; then stop
- re-count; no time limits
Room 1
Team A
5
Team B
14
18
Team C
10
9
Team D
1
6
3
8
4
Team E
2
13
7
15
11
16
12
17
Sub-Experiment 2: Performance
Setup
Randomly assign students to new teams of size: 1, 2, 5, 8
Run 3
4 min.
QA: Re-Count
~10 min.
Run 4
4 min.
QA: Re-Count
~10 min.
Run 5
4 min.
QA: Re-Count
~10 min.
Task
Explain the task for this run:
- sort by color and type
- as much as possible
- count + write the numbers
- 4 min.; then stop
- re-count; no time limits
Room 1
Team A
8
Team B
1
18
Team C
11
17
Team D
3
13
5
14
12
Team E
2
9
4
10
6
15
7
16
Sub-Experiment 3: Chaos + Feedback
Setup
Randomly assign students to 3 new teams of size: 5; put the remaining students into a pool for "special tasks"
Run 6
8 min.
QA: Re-Count
~10 min.
Run 7
8 min.
QA: Re-Count
~10 min.
release control team after QA
Run 8
8 min.
QA: Re-Count
~10 min.
Task
Explain the task for this run:
- sort by color and type
- as much as possible
- count + write the numbers
- 8 min.; then stop
- re-count; no time limits
Hidden Agenda
- nothing happens to the control team in run 6 and 7
- the other 2 teams are disturbed in run 6, 7, and 8
- control team joins one of the other teams to double the team size in run 8
Room 1
Team B
2
11
5
12
8
Team C
4
13
6
15
10
Room 2
Team A
1
9
3
14
7
Pool
16
17
18
carry out "special tasks"
Questionnaire + feedback session

Figure 3: Overview of the experiment (incl. general organization, task overview, subject and room assignments). Every run generates two controlling sheets per team to measure the team performance; one from the run and another one from the re-count.

Table 1: Data to be collected regarding team performance and quality of work, whereas the error rate serves as quality indicator.

Var.	Description
wi	Written items counted per team t in the i^{th} run under time pressure.
ci	Re-counted items per team t in the i^{th} run without time pressure.
err_{abs}	Absolute error computed from wi and ci by: $\text{err}_{abs,t,i} = \lvert \text{wi}_{t,i} - \text{ci}_{t,i} \rvert$; for further analysis, absolute errors are further distinguished in:
err_{bn}	Bottleneck errors that occur if students ran out of time thus not documenting all counted items (missing data).
err_{cl}	Clerical errors that occur if students wrote down "wrong" numbers.

Table 2: Special tasks to affect teams' performance.

Task	Description
ST_1	Go outside.
ST_2	Go to Team B.
ST_3	Go to Team C.
ST_4	Eat some counted/sorted items.
ST_5	Start a discussion on "How should we sort?"
ST_6	Take a cup [sorted items] and put the contained items back into the bowl [unsorted items].
ST_7	Take a cup from another team and put the contained items back into the bowl.
ST_8	Join a team and start some conversation without doing anything else.

the run's time elapsed, the sheets were collected and teams were provided with a fresh sheet to re-count the sorted items without pressure. After the final run, students were provided with a short questionnaire. This questionnaire had two parts: the first part asked for things that went well in the sub-experiments, and part two was asking for things that did not respectively. Besides, observers wrote minutes (team, time-stamp, observation) and (partially) video taped the runs to add more data for qualitative analyses and video analysis/feedback.

Quantitative data was transcribed into a spreadsheet for further analyses using Microsoft Excel and R. Qualitative data was also transcribed into text documents and spreadsheets that served the qualitative analysis. Video data was collected and stored for later analysis.

Execution The "experiment" was conducted twice in the configuration from Figure 3. The first instance was conducted at the Technische Universität München (end of 2012), the second instance was conducted at the BTH Karlskrona (end of 2013). Since the same number of students attended the experiment at both sites, we replicated the experiment setup at BTH. That is, in both instances, 18 graduate students from the respective Computer Science/Software Engineering programs (with required pre-knowledge in Software Engineering and software project management) participated in the class session. The whole session, including set up and feedback session, took approximately four hours[3].

[3] And about 15kg of M&M's were required per instance...

Table 3: Stress factors applied to the teams.

Factor	Description
Noise level	All students were put into one room. Because of the communication within the teams, a continuously increasing noise level was created. Just in sub-experiment 3, the control group was led to another room without any other noise sources.
Boring task	Students had to sort, count, and report M&M's again and again—for about 2 hours.
Bottleneck (resources)	Every team was provided with one controlling sheet only. This bottleneck effectively helped limit/influence work distribution.
Time	Although the basic task is very simple (sort, count, and document), significant stress was put on the teams by restricting the time to just a few minutes.
Missing strategy	All teams were told to organize themselves, i.e., no strategy was suggested by the teachers and students were given no time prior the experiments to discuss proper strategies. The actual strategies had to be developed on the fly.
Overloading	The setup invited work overloading, e.g., for experts in teams.
Disturbances	The special tasks in the "chaos runs" aimed at confronting the students with different external effects (cf. Table 2).
General staff turnover	In the "chaos runs", instructors had to follow the playbook in which detailed orders were issued when to remove how many team members from a team and (optionally) to replace them by new team members (either from the pool or from another team, cf. Table 2).
Loss of key personnel	As special kind of turnover, we enforced some loss of key personnel, e.g., the instructors' playbook explicitly contains a command: *"Find this person that is documenting the counted items and remove it from the team."* This was done *without* informing the rest of the team.

3.4.2 "Special" Tasks

As shown in Figure 3, the third sub-experiment explicitly aims to put pressure on the teams. Therefore, we designed a playbook that comprised some "special tasks" to disturb and even "attack" the teams. Table 2 provides a summary of the special tasks. Students were told to execute a task immediately without talking or doing anything else.

The tasks from Table 2 were executed according to a playbook and aimed at affecting the teams' performance. For instance, ST_1, ST_2, or ST_3 were preferably given to those students that documented the numbers, i.e., applying these tasks to a team aimed at removing a mission-critical person from the team[4] thus forcing the team to re-organize. The tasks ST_4, ST_6, and ST_7 aimed to provoke quality problems (errors) in a team, and to force teams to develop countermeasures (see also Sect. 4.2). ST_5 aims to bring in a new team member that questions the current way of working and raises a debate on principles. Finally, ST_8 brings in team members to provoke unproductive work time. These tasks

[4] This aims to simulate the "bus factor" [8] in the project.

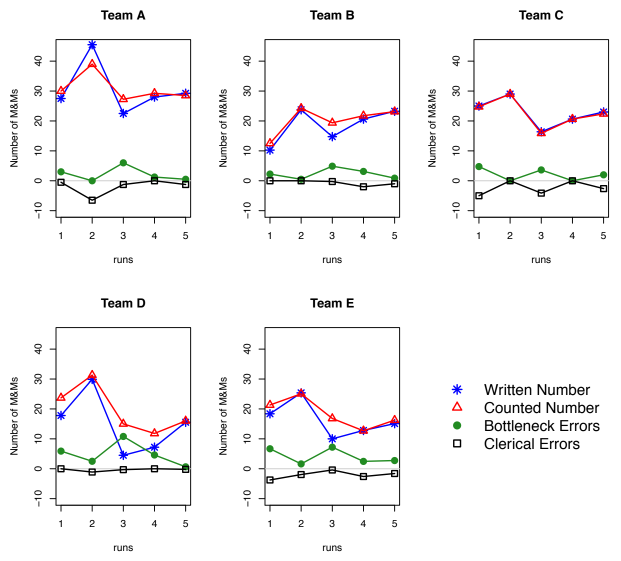

Figure 4: Performance/quality measurement results from sub-experiments 1 and 2 (per person; TUM).

mainly focus on learning goal 3 and contribute to the stress factors that we exposed the teams to (Table 3).

3.4.3 *Reflection*

When the experiment was finished, students were first asked to provide written feedback about the experiment and their experience. After collecting the feedback sheets, in an approximately 15-minute wrap-up session, the experiment was briefly discussed, some details were provided by the teachers, the "hidden agenda" was revealed (i.e., students were informed about stress factors and so forth), and students were asked to reflect on the experiment. In the next class[5], the reflection was continued by discussing the experiment in detail and presenting the outcomes of the quantitative analysis and selected video tapes to link the theoretical framework to what actually happened in the experiment.

4. ANALYSIS AND DISCUSSION

The unit was given twice in Munich and Karlskrona and was fully monitored. We analyze and discuss these instances using the collected performance and qualitative data, which is based on the different controlling sheets, video recordings, observer minutes, and student feedback.

4.1 Performance Analysis

In a nutshell, the Tuckman model says that teams fight several conflicts before they start performing. That is, team performance curves should show a certain behavior after the initial team setup, team reconfiguration, and over time. The purpose of collecting performance data is to generate graphs that allow students to reproduce the Tuckman model and to understand the impact of such group dynamic processes. In this section, we demonstrate, how we presented the performance data to the students and discussed the outcomes.

[5]**Note:** For organizational reasons, we couldn't run the full classroom reflection at BTH.

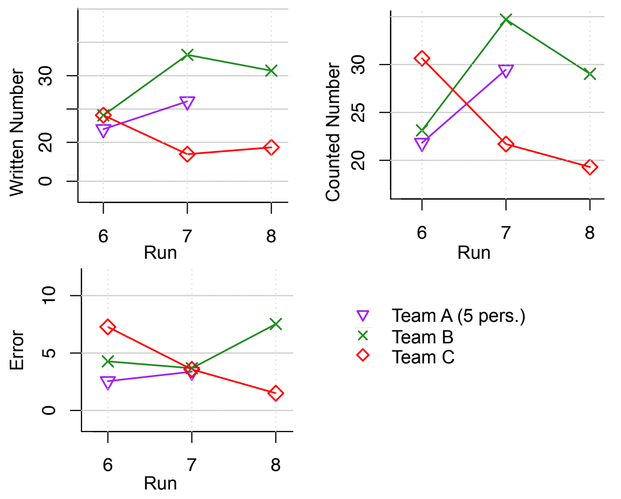

Figure 5: Performance/quality measurement results from sub-experiment 3 (per person; BTH).

4.1.1 *Training and Performance Runs*

The sub-experiments 1 and 2 (Figure 3) serve as training and performance phases in which students learn the task and carry out the task under "normal" conditions, i.e., without enforced extra stress factors. Figure 4 shows the collected performance data for the TUM teams (runs 1-5). To make the performance of the teams more comparable, this presentation is boiled down to the individual person's performance within a specific team setup. That is, the variables wi, ci, err_{bn}, and err_{cl} are presented on a per-person basis.

Figure 4 illustrates one prediction of the Tuckman-Model, which was a key aspect to be learned: changing the team, performance will drop. The data shows this for each team (from run 2 to 3). Figure 4 also shows the students being able to find and apply strategies to perform and minimize the error (in average, the error per person is, regardless of the actual team setup, below three items per run and per person). Furthermore, the figure also shows the teams performing at a comparable level (written/counted items). Another observation is the performance in relation to team size: the bigger the team the more effort is spent on coordination. In particular, the two 2-person teams get close to the 1-person team (no communication/coordination effort at all), while the bigger teams stay behind (individual performance is below the 1- and 2-person teams).

In the Feedback Session: The classroom discussion was initiated by a reflection on the Tuckman model. We then asked to students about their expectations regarding the performance. Then we presented the actual performance curves from Figure 4 and "quantified" the subjective perception. This first part of the performance analysis aimed at improving the understanding of the "normal" team behavior, i.e., what happens when a new team is formed and a new project with a somewhat familiar task is started. Furthermore, students should experience that, e.g., team size and communication effort impact performance and quality (cf. Figure 3; teams size). In particular, students recognized the decreasing per-person performance in growing teams. In the discussion, students found the increased effort caused in the need to find proper strategies to coordinate larger teams.

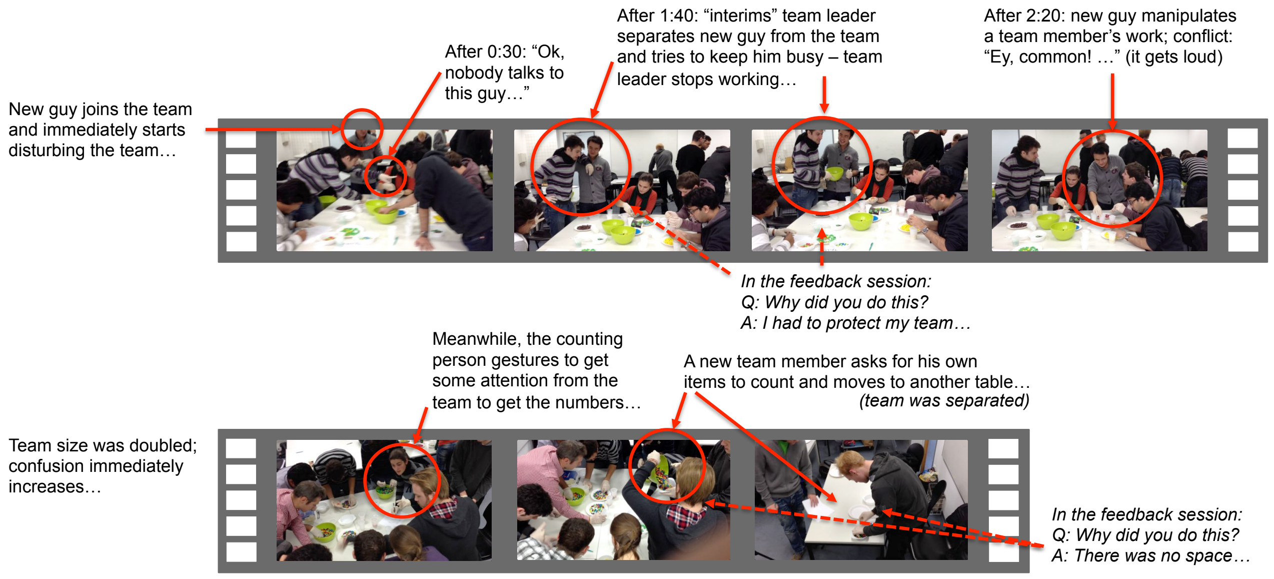
New guy joins the team and immediately starts disturbing the team…
After 0:30: "Ok, nobody talks to this guy…"
After 1:40: "interims" team leader separates new guy from the team and tries to keep him busy – team leader stops working…
After 2:20: new guy manipulates a team member's work; conflict: "Ey, common! …" (it gets loud)
In the feedback session: Q: Why did you do this? A: I had to protect my team…
Team size was doubled; confusion immediately increases…
Meanwhile, the counting person gestures to get some attention from the team to get the numbers…
A new team member asks for his own items to count and moves to another table… (team was separated)
In the feedback session: Q: Why did you do this? A: There was no space…

Figure 6: Selected examples from the video analysis (run 8, teams TUM B and TUM C).

4.1.2 *The "Chaos Runs"*

In sub-experiment 3, the team setup was changed. Team A was separated from the other teams and was taken to another room (control group; just doing the task without any disturbance). Teams B and C remained in the room and faced several "attacks" to influence their performance.

Figure 5 shows the results from the third sub-experiment of the BTH teams for the variables wi, ci, err_{abs}. For Team A, the figures show the expected behavior regarding the variables wi, ci, err_{abs}, i.e., some performance and quality improvement from run 6 to run 7. Yet, the performance and quality indicators for Team B and Team C show a different behavior. The most striking observation is—regardless of the experiment instance—that performance and quality changed. For instance, Team B shows an increasing error rate err_{abs}, i.e., disturbing the teams affects quality. Another observation of BTH's Team C shows that, on the one hand, a decreasing error rate, but, on the other hand, a reduction of the overall performance. That is, disturbing the teams impacts the teams' performance.

In the Feedback Session: In the retrospective, we presented the students these graphs discussing what happened. Having the knowledge about the Tuckman model, students could explain the visualized effect. However, students were surprised about the level of performance reduction. Nevertheless, despite all disturbance, students were still able to work on their tasks. In the feedback session, we also asked the students to explain what happened and what were the stress factors and disturbances in detail. This discussion was then continued with the video analysis.

4.2 Video Analysis

Both unit instances were video taped in order to enable students to see their action and to link the theoretical model to their actual behavior. In total, we collected several hours of video material. In the following, we provide some insights by presenting selected examples and link them to the experiment's playbook, and the different stress factors and actions taken to disturb the teams. Based on Figure 6, we discuss two selected examples from the video analysis.

4.2.1 *Example 1: Taking Responsibility*

The upper part of Figure 6 shows a situation from TUM's Team C, which was caused by a new guy that was told to execute ST_6 and ST_7 (Table 2). Initially, the new guy was welcomed with a "Hi!" and it was tried to integrate him in the team. Yet, some 25 seconds later, the new guy started to eat the sorted items. This was quickly recognized and led to the statement that this guy should be ignored from now on. However, the new guy continued messing up the sorted items and, eventually, one team member took responsibility becoming the team leader. At first, the team leader was only observing the new guy and tried to correct the errors by searching and putting back the 'stolen' items, but, finally, he stopped working and tried to separate the new guy from the team. Thinking the situation was resolved, work went on, but the new guy got back to his task. As the deadline was approaching, the team's mood turned and people became loud and aggressive.

In the Feedback Session: In the feedback session, we presented this scene to the students asking the protagonist for his motivation. He stated that he had to protect the team. We also discussed consequences for the team when he stopped working. The group discussion considered the behavior a compromise: reduced work force to achieve high quality vs. high performance with increased quality risk.

4.2.2 *Example 2: Too many People, too few Space*

The lower part of Figure 6 shows TUM's team B in the moment the team size was doubled to 10 people. In this situation, 5 new people had to be merged with the already working team (cf. Tuckman model, Sect. 3.3), which caused serious confusion and increased the stress level significantly. For instance, the counting person was not only pointing to those that should provide their numbers, but she used more gestures to stop others talking trying to keep some kind of

Table 4: Stress factors and actions as detected by the students, and the number of mentions per experiment (TUM/BTH – 68/81 feedback notes; the table shows if a factor/action was found and in what sub-experiment).

Factor/Action	Found?	...in?	TUM/BTH	Students' comments (selection)
Noise level	✗	–	0/0	*The general noise level was not detected at all.*
Boring task	✓	3	1/0	"After a couple times of sorting by type and color it got a bit boring."
Bottleneck (resources)	✓	1	1/0	"Bottleneck: result sheet"
Time	✓	1	7/10	"Short time", "[...] and time was limited", "No time for counting and result reporting", "counting until 30 secs before time limit, then writing down was the bottleneck", "we only counted 2 colors in the end, because we underestimated the needed time"
Missing strategy	✓	1	15/18	"It took same time to work out a strategy", "No communication rules [...] inefficient counting process.", "very chaotic responsibilities after mixing the team members; no clear instructions for new members because the rest of the team was counting under pressure", "not enough time to talk how it should work; everyone did it in his own way"
Overloading	✓	2	4/5	"the one writer was very busy and overloaded [...]", "writer was overloaded because of too many announced numbers", "everyone was shouting and not waiting [...]"
Disturbances	✓	3	5/2	"new team members in the final run", "round 3: team too large and no coordination", "People got angry after the disturbances [...] more errors were made; more chaos"
General team turnover	✓	3	6/1	"joining the new team in the middle of experiment scared the other partners.", "when someone left the team or change the table, nobody knew if his glass was already counted or not", "change the people to another group because you don't remember your add of colors"
Loss of key personnel	✓	3	1/1	"M&Ms were forgotten because the responsible person was removed [...]"

reporting structure. At the same time, the new people tried to find some space around the table to join the work. However, one person ended up taking some items to sort and count to another table (outside the team's work space), as there was just not enough space.

In the Feedback Session: For this scene, the classroom discussion ended up with the finding that just adding more and more people to projects that are short in time does not help anything (see also F. Brooks' "The Mythical Man Month" [6]): the writer was stressed and there was not enough space for all team members to work collaboratively.

4.3 Feedback Analysis

Students provided feedback by answering a free-form questionnaire (Sect. 3.4.1) in which they should summarize their own positive and negative observations from the experiment. In subsequent sections, we first analyze the questionnaires for the different stress factors, if students detected them, and how the stress factors were experienced. In the second part, we looked for those aspects that were considered positive and that contributed to the learning goals of this course unit.

4.3.1 *Detected Stress Factors*

Table 4 provides a summary of the students' observations. The table shows whether and when a stress factor or some action was detected by the students and identified cumbersome. Furthermore, the table lists the number of mentions, i.e., how many times a stress factor or action was mentioned—multiple mentions were possible and students often mentioned different factors together.

Based on the feedback sheets, we found 32 (TUM) and 38 (BTH) mentions of stress factors. In total, we found 8 out of the 9 pre-defined factors (Table 3) detected by the students. Time pressure, the bottleneck created by only one available controlling sheet per team, and the missing strategy were detected immediately in the first sub-experiment. Overloading was mentioned the first time in sub-experiment 2 when the task's complexity was increased, and the remaining factors were realized in the third sub-experiment. In summary, students considered time pressure (7 out of 10 mentions) and missing strategies (15 out of 18 mentions) impacting their performance the most. Furthermore, beyond the pre-defined stress factors, the BTH teams also found task complexity (2 mentions) and communication aspects (2 mentions) affecting efficient work[6].

The general noise level was, however, not detected at all—even when we enforced this stress factor by playing loud music in the background (students noticed that there was some music playing, but did not complain). Moreover, the increasing noise from shouting the numbers to the writing person was not considered cumbersome from the noise perspective rather than from chaos and overloading, i.e., noise was somehow noticed, but stress and chaos were considered more affecting the work.

Therefore, we consider *learning goal 3* (Sect. 3.2) achieved,

[6]**Note:** Both groups were composed of students from different countries and cultures. That is, that even in this small co-located settings, GSE-related effects occur, such as described in [1, 7]. Students stated language barriers becoming problematic under pressure, as for instance heavy accents challenged the communication in the teams.

Table 5: Positive mentions and lessons learned (TUM: 36/68 mentions, BTH: 43/81 mentions).

Task	TUM	BTH
Strategy and Team Development	23	24
Start with initial Strategy	4	6
Efficiency of Small Teams	6	3
Benefits of Bigger Teams	1	3
Impact of more Time	0	5

as students were able to detect different stress factors and to develop proper approaches to deal with exceptional situations (cf. Figure 6).

4.3.2 *Students' Lessons Learned*

The other part of the questionnaire aimed at finding positively perceived events. As we had no target categories defined in advance, we transcribed and analyzed the questionnaire responses, and we found five aspects frequently mentioned: strategy and team development, starting a run with an initial plan, impact of time, and benefits of smaller/bigger teams. Table 5 provides an overview of the positively perceived effects of the different sub-experiments/runs.

Comparing Table 5 with Table 4 we see that, on the one hand, the missing strategy was considered the most critical point, but on the other hand, students mentioned strategy and team development as a very positively experienced aspect. Starting with sub-experiment 2, students joined teams having initial ideas and strategies of how to optimize work toward maximum performance (learning effects from the training phase). In runs 3 and 6 (first runs in new team setups), we observed the students spending time (usually some 30 seconds) to discuss and agree on an approach to work. In these teams, opportunities for knowledge transfer as part of the team's improvement were also mentioned positive in the questionnaires and feedback session.

From the questionnaires, we also see the *learning goals 1* and *2* achieved. Students found strategies to quickly team up, and to define, apply, and improve working strategies. Furthermore, students were eventually able to deal with staff turnover and to compensate for lost staff and to integrate new staff. In the feedback and discussion sessions, we could successfully link the theoretically predicted performance and quality drops with the students' experiences.

5. CONCLUSION

In this paper, we presented a course unit on group dynamics in project teams. Such teams are key to todays software development in which cross-functional teams develop software in short cycles. As working in such teams challenges team members not only from the technical but also the social perspective, the presented course unit contributes to project management courses by providing an environment in which students can directly experience the impact of team size and composition, stress factors, and exceptional situations to project performance and result quality.

The course unit uses the well-known Tuckman model as theoretical framework to explain group dynamics in team formation and development, and it borrows from controlled experiments to organize the practical parts. Eventually, students are exposed to different stress factors to learn about the meaning of team formation processes, especially when teams work under pressure.

We conducted the course unit twice with two groups of 18 graduate students each at two different universities. In both instances, students could detect almost all stress factors, and students were able to develop and improve collaboration strategies quickly, even when the team setup was changed, and students were able to develop strategies to handle exceptional situations, e.g., staff turnover and attempts to attack the project. The learning goals set could be achieved, and in the final exams, students showed their ability to understand a critical project scenario and to develop proposals to solve conflicts in teams. Finally, in the course feedbacks, students working in industry stated the course unit valuable due to the applicability to their work environment.

The presented course unit has been transferred from Munich to another university during an Erasmus exchange, and is scheduled for being set up in 2016. We experienced especially the practical parts fairly easy to set up, as it only requires rooms, tables, supervisors, and items to sort. Within less than four hours, students gain practical experience of stress factors and team formation. A replication kit provides all material required to implement the unit.

Therefore, the presented course unit and the complementing material help establishing and optimizing courses that demand to practically explain and show theories or conventional wisdom, to experience real world problems, and to learn how to deal with such problems in a differentiated manner. Furthermore, the course unit contributes a practical exercise to otherwise more theoretical and abstract subjects. For instance, given the project management context, the presented unit allows for focusing on different management- and team-related aspects without relying on code-centered work thus avoiding the risk of "skipping" the management tasks to get the software done.

The course unit was appreciated by the students, and it was also considered fun. Given the positive learning effects and feedback we received, we plan to replicate this unit in upcoming courses. Furthermore, as this is the second successful implementation of our teaching pattern [19], we positively experienced the feasibility of our pattern which motivates us to adopt this pattern for further courses as for instance courses on software test[7], which provide a rich ground for defining appropriate experiments. In this context, we are also setting up collaboration among different universities across Europe, e.g., to exchange our course- and experiment designs, conduct replications (research-driven), and to arrange guest lectures.

Acknowledgement

We want to thank Henning Femmer and Jonas Eckhardt for their significant support in setting up the experiment. We also owe thanks to Katrin Kehrbusch for her support in data transcription and analysis. Furthermore, we thank Darja Šmite and her team of Ph.D. and graduate students, who hosted and supported the experiment instance at BTH. Finally, the experiment was supported in part by the "Ernst Otto Fischer" teaching award/grant 2012 by Faculty of Infor-

[7]The application for the topic software quality and software test is currently in progress in the courses *Design of Software Systems* and *Software Quality Management* at the University of Southern Denmark.

matics of Technische Universität München and the Erasmus Teaching Mobility program.

6. REFERENCES

[1] A. Avritzer, S. Beecham, R. Britto, J. Kroll, D. Sadoc Menasche, J. Noll, and M. Paasivaara. Extending survivability models for global software development with media synchronicity theory. In *International Conference on Global Software Engineering*, pages 23–32, 2015.

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

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

[4] G. Bavota, A. De Lucia, F. Fasano, R. Oliveto, and C. Zottoli. Teaching software enginerring and software project management: An integrated and practical approach. In *International Conference on Software Engineering*, pages 1155–1164, 2012.

[5] J. H. Bradley and F. J. Hebert. The effect of personality type on team performance. *Journal of Management Development*, 16(5):337–353, 1997.

[6] F. P. Brooks. *The Mythical Man-Month*. Addison-Wesley Longman, 1995.

[7] V. Casey and I. Richardson. Project management within virtual software teams. In *International Conference on Global Software Engineering*, pages 33–42, 2006.

[8] J. O. Coplien and N. B. Harrison. *Organizational Patterns of Agile Software Development*. Prentice Hall, 2004.

[9] D. Dahiya. Teaching software engineering: A practical approach. *ACM SIGSOFT Software Engineering Notes*, 35(2):1–5, 2010.

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

[11] F. Fagerholm, A. S. Guinea, H. Mäenpää, and J. Münch. Building blocks for continuous experimentation. In *International Workshop on Rapid Continuous Software Engineering*, pages 26–35, 2014.

[12] F. Fagerholm, M. Ikonen, P. Kettunen, J. Münch, V. Roto, and P. Abrahamsson. Performance alignment work: How software developers experience the continuous adaptation of team performance in lean and agile environments. *Information and Software Technology*, 64:132–147, August 2015.

[13] 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.

[14] N. Gorla and Y. W. Lam. Who should work with whom?: Building effective software project teams. *Communications of the ACM*, 47(6):79–82, June 2004.

[15] J. Jiang and G. Klein. Software development risks to project effectiveness. *Journal of Systems and Software*, 52(1):3–10, 2000.

[16] E. Kamsties and C. Lott. An empirical evaluation of three defect-detection techniques. In *Europ. Software Engineering Conference*, pages 362–383, 1995.

[17] E. Keenan, A. Steele, and X. Jia. Simulating global software development in a course environment. In *International Conference on Global Software Engineering*, pages 201–205, 2010.

[18] Kent Beck et al. Manifesto for agile software development. Available from http://www.agilemanifesto.org, 2001.

[19] M. Kuhrmann. A practical approach to align research with master's level courses. In *International Conference on Computational Science and Engineering*, pages 202–208, 2012.

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

[21] J. Münch, F. Fagerholm, P. Johnson, J. Pirttilahti, J. Torkkel, and J. Järvinen. Creating minimum viable products in industry-academia collaborations. In *Lean Enterprise Software and Systems Conference*, pages 137–151, 2013.

[22] Ocampo, A. and Münch, J. Rationale modeling for software process evolution. *Journal on Software Process: Improvement and Practice*, 14(2):85–105, 2009.

[23] W. Pádua. Measuring complexity, effectiveness and efficiency in software course projects. In *International Conference on Software Engineering*, pages 545–554, 2010.

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

[25] I. Richardson, A. E. Milewski, and N. Mullick. Distributed development: an education perspective on the global studio project. In *International Conference on Software Engineering*, pages 679–684, 2006.

[26] 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.

[27] G. P. Sudhakar, A. Farooq, and S. Patnaik. Soft factors affecting the performance of software development teams. *Team Performance Management: An International Journal*, 17(3/4):187–205, 2011.

[28] B. W. Tuckman. Developmental sequence in small groups. *Psychological Bulletin*, 63(6):384–399, 1965.

[29] L. Wallace, M. Keil, and A. Rai. How software project risk affects project performance: An investigation of the dimensions of risk and an exploratory model*. *Decision Sciences*, 35(2):289–321, 2004.

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

[31] H.-L. Yang and J.-H. Tang. Team structure and team performance in is development: A social network perspective. *Inf. Manage.*, 41(3):335–349, Jan. 2004.