A Socio-Technical Framework for Face-to-Face Teaching in Large Software Development Courses

Marlo Haering
Department of Informatics
University of Hamburg
Hamburg, Germany
haering@informatik.uni-hamburg.de

Walid Maalej
Department of Informatics
University of Hamburg
Hamburg, Germany
maalej@informatik.uni-hamburg.de

***Abstract*—In face-to-face teaching, students work in pairs on programming exercises and present their solutions to tutors. This setting fosters social skills. Students benefit from immediate feedback loops and personalized explanations. However, with an increasing number of students, it becomes challenging to scale this approach to very large courses due to the logistic and organizational effort. In this paper, we first report on significant challenges that we identified while conducting face-to-face teaching in a software development course with more than 600 students and 50 tutors. Second, we introduce a preliminary socio-technical framework for face-to-face teaching to facilitate logistical aspects, monitor the students exercise progress, and improve the students' learning experience.**

***Index Terms*—face-to-face exercises, software development education, learning analysis**

I. Introduction

Our Applied Software Technology group at the University of Hamburg hosts the introductory programming course *Software Development 1* (SE1). SE1 consists of two parts: a weekly 90-minute lecture and a weekly 3-hour face-to-face exercise session. The course is compulsory for all computer science majors. Students with other majors are also eligible to register for SE1 including mathematics, pedagogy, physics, or psychology. SE1 is an entry-level course in programming and requires no prior knowledge. We host SE1 annually in winter terms.

The exercises are structured thematically in accordance with the lecture. At first, we introduce new concepts with examples in the lecture. In the subsequent practice week, the students work in pairs on an exercise sheet in the laboratory rooms as shown in Figure 1. The programming exercises are inspired by Barnes and Kölling [1] and typically require the students to complete a well-defined implementation task in a prepared project.

During this session, tutors (teaching assistants and student tutors) are present for two main purposes. On the one hand, they support the students in case of questions or uncertainties and on the other hand they approve the solutions of the students. After the students completed a subtask, they have to present their solutions to a tutor. The tutor either accepts or declines the solution. A tutor accepts the solution if it (1) solves the task, (2) the students present and explain their solution with their own words, and (3) answer subsequent

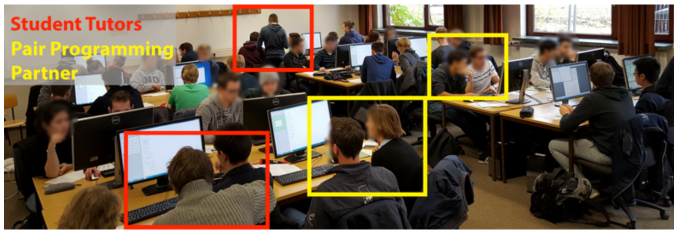
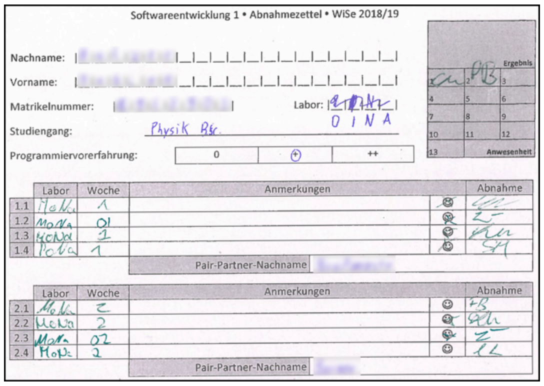


Fig. 1. Laboratory room with pair programming students (yellow frame) and student tutors (red frame). The tutors check the solutions of the students.



Softwareentwicklung 1 • Abnahmezettel • WiSe 2018/19

Nachname:	[redacted]
Vorname:	[redacted]
Matrikelnummer:	[redacted]
Labor:	~~[illegible]~~ 0 I N A
Studiengang:	Physik B.Sc.
Programmiervorerfahrung:	0 [✓ +] ++

		Ergebnis
1 [illegible]	2 [illegible]	3
4	5	6
7	8	9
10	11	12
13	Anwesenheit	

	Labor	Woche	Anmerkungen		Abnahme
1.1	MoNa	1		☺	[illegible]
1.2	MoNa	01		☺	[illegible]
1.3	MoNa	1		☺	[illegible]
1.4	MoNa	1		☺	[illegible]
			Pair-Partner-Nachname	[redacted]	

	Labor	Woche	Anmerkungen		Abnahme
2.1	MoNa	2		☺	[illegible]
2.2	MoNa	2		☺	[illegible]
2.3	MoNa	02		☺	[illegible]
2.4	MoNa	2		☺	[illegible]
			Pair-Partner-Nachname	[redacted]	

Fig. 2. Part of a student's evaluation sheet: student details (top), attendance table (top right), and one table per exercise sheet, each with four subtasks.

questions. Tutors check off each passed subtask by signing the student's evaluation sheet, which is shown in the tables in Figure 2. The tutor declines the solution if the presentation or the solution is insufficient. In this case, the students have to improve the shortcomings and present their corrected solution. Students have to have all exercises accepted to pass SE1. Course instructors prepare the teaching material, monitor the students' progress, and manage the logistic aspects. The course evaluation showed that students appreciate this format.

Face-to-face exercises encourage the students to reproduce their just learned knowledge by explaining it with their own words to a tutor. Thereby, tutors identify and correct misunderstandings early and provide immediate feedback with personalized explanations. Students have to program in pairs as previous research has shown an improved code quality and better student grades [2], [3]. Furthermore, this fosters communication and presentation skills as well as social interactions among students and with their tutors as we consider these important skills for prospective software developers. In contrast to a solely technical based framework that checks students' task automatically, we retain the social interaction between students and tutors. Therefore, we label our framework "socio-technical" to highlight that the operational face-to-face tutoring stays an essential part in our framework [4].

In recent years, the number of students has risen steadily and has now reached more than 600, which are mostly first semester students. Applying this approach to large software development classes is challenging because it does not scale easily. We summarize the main challenges that we identified in Section II. In Section III, we introduce our preliminary socio-technical framework to facilitate the logistical workload, to enable learning analysis for course instructors, and to keep the face-to-face component in the exercise sessions.

II. Challenges

In this section, we describe the challenges that we identified during our previous SE1 courses and partly outline how we currently cope with them.

A. Scalability and Logistics

In recent years, the number of students in our SE1 course has steadily increased (~300 students in 2007-09, ~400 students in 2010-12, ~500 students in 2012 and ~600 students in 2016). Additionally, computer science is becoming more and more important across various disciplines. In particular, SE1 is also eligible for non-computer scientists. We expect that the number of students will increase to maybe more than 800 students in the next 2-3 years. A higher number of students will make it challenging to keep conducting face-to-face exercises in SE1 as it requires more laboratory rooms and more tutors.

The logistics of the SE1 exercises comprise various parts about tutor management, teaching material, and student communication. For the preparation of the SE1 exercises, we recruit student tutors in addition to teaching assistants. The number of required tutors scales directly with the number of students. We contact students via different channels including other courses, mailing lists, or face-to-face. The students apply via an online form for a tutor position. We mainly select students based on their tutoring experience, recommendations by previous tutors, and prior grades.

Students register for one of eight different time slots during the week for the face-to-face exercises. The preferences of the students differ each semester. The course instructors try to assign a sufficient number of tutors to each exercise session, depending on the availability of tutors and the number of registered students. Furthermore, a teaching assistant is assigned for each exercise session for supervision and answering questions of the tutors.

Most of the exercise sessions require multiple rooms, for instance, one of our sessions with more than 120 students spreads across seven rooms. Tutors change rooms to check the occupancy in other rooms. Based on our experiences, we found that a ratio of one tutor to seven students is sufficient. For tutoring, we usually require ~40 student tutors and ten teaching assistants. Most of the tutors apply for more than one exercise session.

The programming experience among the tutors is heterogeneous. We observed that experienced tutors have higher demands on the solutions of the students. However, it is challenging to maintain consistent acceptance criteria across all tutors for the evaluation. Therefore, we set up a two-hour introductory meeting with all tutors before the semester starts. Additionally, we provide a briefing session each week for the tutors to collect feedback about the previous exercise week, walk through the upcoming exercise sheet, and discuss the acceptance criteria.

B. Alternating Teaching Staff

The general aim is to maintain knowledge about the logistics independently of an individual course instructor so that different employees can take over course management activities at any time. This is particularly important as the logistics contain fine-grained tasks with crucial deadlines and the staff is typically rotating after two years. Therefore, we explicitly document single steps and their deadlines for each part. Always two course instructors take responsibility for the SE1 exercises to ensure reliability. Furthermore, the fluctuation among the tutors is high with ~60% new tutors each semester. Experienced tutors are beneficial to support new tutors and teaching assistants in many aspects of the practice. To mitigate the knowledge loss, we require student tutors to follow along with other tutors during the semester for three hours in total to enable a knowledge transfer between experienced and new tutors.

C. Short-term Replacement and Rush Management

The occupancy of each exercise session is inconsistent as students visit other time slots when they could not finish the exercises in time. Also, tutors are regularly absent due to illness. Due to these unexpected factors, single exercise sessions exceed a proportional ratio between tutors and students. This leads to long waiting times for students until a tutor is available for their requests. In this case, tutors request tutoring support on demand from teaching assistants to overcome a temporary shortcoming.

As each of the student tutors have their course schedule, we provide two redundant briefing sessions per week. The student tutors have to attend one of these sessions to prepare for the upcoming exercise sheet. Two teaching assistants carry out two separate briefing sessions for the student tutors so that they can substitute each other. The teaching assistants prepare the briefing session and discuss the current version of the exercise sheet.

D. Continuous Progress Monitoring and Feedback on Course

To pass the SE1 exercises, students have to pass the requirements on each exercise sheet. During the semester the tutors continuously monitor how many students currently comply with the conditions to pass the SE1 exercises. As the tutors check off the approvals on the evaluation sheet, it is currently time-consuming to analyze each sheet manually and to continuously monitor the students' progress.

Moreover, course instructors collect feedback on the SE1 exercise part via three different channels: (1) from the tutors in the briefing sessions on their impression about how the students cope with the exercises, (2) from students via an anonymous online survey, (3) from students via an anonymous evaluation of the whole course at the end of each semester. Course instructors aggregate the feedback and formulate concrete change requests to improve both the process as well as each weekly exercise sheet. The complexity of these changes varies from typos to replacing subtasks or source code projects.

E. Participants with Different Prior Knowledge

The students in SE1 are from diverse studies with different prior knowledge. Because the participants increasingly come from subjects outside of the computer science department and outside of the MIN faculty, we have to adapt to different prior knowledge levels. Therefore, in the lecture and especially in the exercises, we must be able to act in a target-group-oriented and adaptive manner to reach all students. Consequently, it is a challenge to design the course for students without any previous experience and at the same time motivate advanced programmers. In an ideal case, students could learn from each other.

We still explore and experiment with different prior knowledge combinations and other mechanisms to motivate experienced programmers and educate beginners. We observed two different patterns when students, having an unequal level of prior knowledge work together. Some experienced students allow their partners to contribute to the solution and they actively explain concepts to their partners. Unfortunately, the other pattern is that experienced programmers finish the exercises quickly alone, explain the solution to their partner, and finally get the solution checked by a tutor and leave early. We highlight this as a negative example in the lecture and ask the tutors to spot this pattern.

III. Socio-technical Teaching Framework

In this section, we introduce our preliminary socio-technical face-to-face teaching framework that partly solves the challenges we described in the previous section. We describe the framework from the perspective of the three roles: students, tutors, and course instructors.

A. Students Web Interface

Students access the framework via a web interface. They access the current state of their evaluation sheet, exercise sheets, lecture slides, and further references. At the beginning of each exercise session, each student pair logs into the system and enters their seat and pair programming partner. During the exercise session, students enqueue requests for the tutors, for instance, an evaluation of a subtask or a question. Based on the number of predecessors in the queue and the number of available tutors, the framework indicates an estimated waiting time for the students until the next tutor will be available.

B. Tutors App

Tutors log into the framework via their smartphone. They have access to an extended version of the exercise sheet with additional remarks, sample solutions, and sample questions to ask students during the evaluation. During the exercise session, they process the queue of student requests. An available tutor queries the next request from the student queue and visits their workplace, which might be in a different room. The tutor has access to the evaluation sheets of the students and sees comments by tutors on previous evaluations. The tutor evaluates the solution and rates different categories as, for instance, presentation, code quality, and speech share. Thereby, we keep the operational face-to-face teaching by the tutors as a social component in our framework. Following tutors utilize previous evaluation data and take previous comments into account. For instance, if a tutor notices that one of the students shows less participation in the presentations, the tutor could directly address questions to that student.

In our course, the attendance of the weekly exercise sessions is mandatory. The framework automatically flags students as "attended", if a tutor checks a solution to a subtask. Otherwise, students have to enqueue to contact a tutor for requesting attendance manually.

We observed that restrained students hesitate to ask tutors when they get stuck on an exercise. Our framework monitors the progress of each student pair in the background and enqueues a request for slow progressing students automatically. Thereby, tutors intervene and offer support early when students fall behind.

C. Course Instructors Dashboard

Course instructors access the framework via a web interface. For the tutor recruitment, they set up the questions for the application form, which is an integrated part of the framework. Course instructors monitor both the application process as well as the distribution of students in the exercise sessions during the registration period. Applicants fill out the form with their personal information, additional references, information about previous grades, as well as their availability and how many exercise sessions they plan to supervise. The framework automatically indicates unbalanced exercise sessions and notifies the course instructors. Course instructors constantly monitor the ratios between tutors and students for each time slot and cap the participation limits for students to register for the exercise sessions.

Based on the data, collected in the exercise sessions, course instructors acquire an insight into the current status of the SE1 exercise part. The dashboard for course instructors offers an insight into the following metrics:

- **Ratio of students compliant with conditions.** The students in SE1 require to fulfill the conditions of the SE1 exercises. Our framework shows the proportion of students who are currently compliant with the SE1 exercise conditions, how many are behind, and how many dropped out.

- **Attendance.** Course instructors get an overview of when students attend the exercise sessions. How many students have to catch up with exercise sheets when they were not able to complete the exercise sheet in one exercise session? We further use this information to fine-tune the balancing of tutors among the time slots.
- **Complexity of exercises.** As tutors check each subtask of an exercise sheet individually, course instructors get an insight into the time it takes students to complete each subtask. Based on this metric, course instructors scope and fine-tune each subtask so that solving a complete exercise sheet is manageable in a three hour exercise session.
- **Pair programming partner matching.** Course instructors get an overview about the pair programming partner combinations. Which pair programming partner combinations are common? What is their level of prior-knowledge? Course instructors use these insights to adjust future pair programming partner matching.
- **Quality assurance for tutors.** Each evaluation of a student solution is different. It is crucial that tutors have uniform requirements for approving a subtask. Course instructors are interested in questions such as: How many evaluations does each tutor perform? How much time does a tutor spend on checking each subtask? What is the accept and reject ratio for each tutor? Course instructors can discuss significant evaluation differences in the tutor briefing sessions.

IV. Related Work

Other researchers discussed teaching approaches for teaching software development face-to-face to benefit from immediate feedback loops. Kothiyal et al. [5] applied Think-Pair-Share in a large introductory programming course. They showed that they obtained a sustained engagement of 83%. This lecture integrated two Think-Pair-Share phases while students could ask an instructor for help. Krusche et al. [6] reported on their experiences with an interactive learning approach. They pause the lecture in between and conduct in-class exercises to minimize the delay between theoretical knowledge and practical application. They further show in another study [7] that active participation in these exercises leads to a better exam grade. Krusche and Seitz [8] developed an automatic assessment system for large computer science courses to cope with the manual assessment effort. Our approach differs from this approach by adding a social component in which the students have to present their solutions to a tutor. Iacob and Faily [9] redesigned their second-year undergraduate course on software engineering for a large cohort. They also pointed out main challenges when hosting this course and concluded that scalable exercise sessions are needed to teach software engineering skills.

V. Discussion and Conclusion

Face-to-face exercises entail a huge logistic effort, which covers diverse aspects including tutor management, improving teaching material, and student communication. We report on major challenges that we identified in hosting the exercises of SE1, a beginner level software development course with more than 600 students with different levels of prior knowledge from different majors. We conduct face-to-face exercises assisted by 50 tutors who assist and evaluate the students and identified challenges including, sustaining hidden knowledge in case of alternating teaching staff, reacting adaptively when tutors are overwhelmed by a high number of students, monitoring the workload of students, and dealing with heterogeneous participants.

We suggest a preliminary socio-technical teaching framework which keeps the face-to-face tutoring as an essential social part. Tutors enter the evaluation of the students' solutions in an app. This data enables course instructors to gain insight into the practical exercise sessions to further improve the learning experience for students in subsequent SE1 courses. So far, we implemented a proof of concept, but in the future, the complete framework has to become robust, secure and tested extensively to be deployed as it processes sensitive student data.

Acknowledgment

We thank Axel Schmolitzky, Heinz Züllighoven, Fredrik Winkler, and Guido Gryczan for implementing this face-to-face teaching approach in SE1 at the University of Hamburg.

References

[1] D. J. Barnes and M. Klling, *Java lernen mit BlueJ: Eine Einfhrung in die objektorientierte Programmierung*. Pearson Deutschland GmbH, 2009.

[2] C. McDowell, L. Werner, H. Bullock, J. Fernald, C. McDowell, L. Werner, H. Bullock, and J. Fernald, "The effects of pair-programming on performance in an introductory programming course," *ACM SIGCSE Bulletin*, vol. 34, pp. 38–42, Feb. 2002.

[3] N. Nagappan, L. Williams, L. Williams, M. Ferzli, E. Wiebe, K. Yang, C. Miller, and S. Balik, "Improving the CS1 Experience with Pair Programming," in *Proceedings of the 34th SIGCSE Technical Symposium on Computer Science Education*, SIGCSE '03, (New York, NY, USA), pp. 359–362, ACM, 2003.

[4] G. Baxter and I. Sommerville, "Socio-technical systems: From design methods to systems engineering," *Interacting with computers*, vol. 23, no. 1, pp. 4–17, 2011.

[5] A. Kothiyal, R. Majumdar, S. Murthy, and S. Iyer, "Effect of Think-pair-share in a Large CS1 Class: 83% Sustained Engagement," in *Proceedings of the Ninth Annual International ACM Conference on International Computing Education Research*, ICER '13, (New York, NY, USA), pp. 137–144, ACM, 2013.

[6] S. Krusche, N. v. Frankenberg, and S. Afifi, "Experiences of a Software Engineering Course based on Interactive Learning," in *SEUH*, 2017.

[7] S. Krusche, A. Seitz, J. Brstler, and B. Bruegge, "Interactive Learning: Increasing Student Participation Through Shorter Exercise Cycles," in *Proceedings of the Nineteenth Australasian Computing Education Conference*, ACE '17, (New York, NY, USA), pp. 17–26, ACM, 2017.

[8] S. Krusche and A. Seitz, "ArTEMiS: An Automatic Assessment Management System for Interactive Learning," in *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*, SIGCSE '18, (New York, NY, USA), pp. 284–289, ACM, 2018.

[9] C. Iacob and S. Faily, "Redesigning an Undergraduate Software Engineering Course for a Large Cohort," in *Proceedings of the 40th International Conference on Software Engineering: Software Engineering Education and Training*, ICSE-SEET '18, (New York, NY, USA), pp. 163–171, ACM, 2018.