Distributed Software Development with One Hand Tied Behind the Back: A Course Unit to Experience the Role of Communication in GSD

Marco Kuhrmann
University of Southern Denmark
Odense, Denmark
kuhrmann@mmmi.sdu.dk

Jürgen Münch
Reutlingen Univeristy
Böblingen, Germany
j.muench@computer.org

***Abstract*—Software development consists to a large extent 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 role of different communication patterns in distributed agile software development. In particular, students gain awareness about the importance of communication by experiencing the impact of limitations of communication channels and the effects on collaboration and team performance. The course unit presented uses the controlled experiment instrument to provide the basic organization of a small software project carried out in virtual teams. We provide a detailed design of the course unit to allow for implementation in further courses. Furthermore, we provide experiences obtained from implementing this course unit with 16 graduate students. We observed students struggling with technical aspects and team coordination in general, while not realizing the importance of communication channels (or their absence). Furthermore, we could show the students that lacking communication protocols impact team coordination and performance regardless of the communication channels used.**

***Index Terms*—experimentation; communication; distributed software development; agile software development**

I. Introduction

Today, hardly any software product or IT service is developed at only one place or by only one team. Therefore, deep knowledge about the communication within (virtual) teams is crucial to successfully carry out software development projects. At the *IEEE International Conference on Global Software Engineering* (ICGSE), communication in distributed software teams was one of the most frequently mentioned concerns [1] thus worth paying attention and building awareness in software development and project management courses.

Context and Related Work: Communication is crucial, as for instance found by Begel and Nagappan [2], who surveyed employees at Microsoft finding a frequent collaboration across time zones. They report that communication difficulties around coordination are the most critical and difficult to solve issues. Al-Ani and Edwards [3] analyzed communication patterns and techniques, and concluded that communication models tend to be hybrids of existing communication models and that team communication evolves over time. From the managerial perspective, communication in distributed teams exposes project managers to challenges, which are discussed by Casey and Richardson [4]. Authors state distance (specifically coordination, visibility, communication and cooperation) within a virtual team challenging project management and conclude that *"project management of a virtual team must be carried out in a different manner to that of a team in a single-site location"*. Stapel et al. [5] present FLOW—an approach to (formally) plan a project-specific communication strategy and to assess the conformance of the implemented strategy in a project. Additionally, over the years, different communication techniques to support distributed projects were analyzed, as for instance instant messaging [6]–[8] or open conversation spaces [9]. Furthermore, some theoretical frameworks and respective tool support were adopted to distributed development, such as the media synchronicity theory [10] or the activity theory [11]. Niinimaki et al. [12] analyzed the use of text-based communication finding that notably technical staff and people that question their language skills prefer text-based communication to direct communication.

In the work presented here, we use different findings and experiences (e.g., from research cited above, or GSD courses, such as [13]–[17]) to create a setting in which students can experience the central role of communication in distributed projects. In particular, we define two setups in which a "direct" communication via Skype and a text-based communication via e-mail are utilized to communicate and coordinate a team.

Contribution: In this paper, we present a course unit, which is focused on building awareness regarding the importance of communication in distributed projects. The course unit combines different techniques from Agile requirements engineering, development (practices), and testing. These elements are combined in a 4-hour exercise in which student teams have to develop a small application. The student teams face different challenges: *limitation of available communication channels* to build general awareness about the role of communication, *missing management frameworks* and *resource limitations* to force the student teams to define proper work patterns, and *task*

overloading to enforce a sound project setup including goal definition, prioritization, and task assignments. The course unit presented was implemented as part of an advanced course on "Agile Project Management & Software Development" (APM; [18], [19]). We provide a full description of the course unit to allow for transfer and implementation in other contexts, and we discuss our findings and lessons learned.

Outline: Section II presents the course unit by presenting its position in the APM course, learning goals, theoretical and practical elements, and general organization. Section III presents results from the implementation and a discussion of lessons learned. We conclude this paper in Sect. IV.

II. General Course Unit Design

We provide an overview of the course context in Sect. II-A. Learning goals are presented in Sect. II-B. The theoretical background and the instrument to implement the practical parts are presented in Sect. II-C and Sect. II-D respectively.

A. Course Context

The course "Agile Project Management & Software Development" (APM; [18], [19]) follows the pattern presented in [20], [21], and is illustrated in Figure 1 that shows the general course organization and 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.

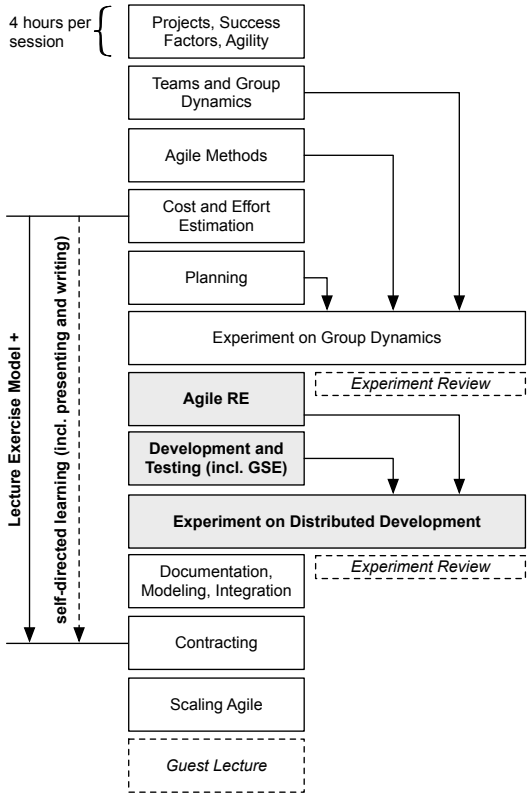

Fig. 1. Basic organization of the APM course. The distributed programming exercise covers requirements engineering, design, development and testing activities, and is carried out in virtual teams.

B. 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 communication within virtual teams on project management and software development activities, in particular, team performance and result quality (Figure 1 shows the other course units providing required input and context). Hence, we define the following learning goals:

Learning Goal 1 *How to efficiently work in a virtual team?* This goal addresses the students' ability to quickly come together, develop and implement work strategies, to set priorities, and to collaboratively develop a software.

Learning Goal 2 *What is the impact of communication?* Students are put into virtual teams and each team is further structured into groups, which work in separated rooms. Each team is exposed to limitations of the communication vehicles available and students are not allowed to use any other form of communication with their remotes. This goal addresses the learning about the central role of rich communication in (distributed) software development.

Learning Goal 3 *What are the risks of poor communication, and how to handle them?* Students face a situation in which they first have to overcome the limitations of the communication to form a working team. This goal mainly addresses gaining experience regarding the need for rich communication and to provide an environment in which students can experience the impact of missing/lacking communication.

C. Theory

The theoretical background of this exercise is given by the different communication patterns as for instance described in the paper's context (Sect. I). In particular, distance, mixed language setups with English as *lingua franca*, and tool-supported communication are used to provide proper means to understand the challenges of communicating within distributed teams. As a second component, interaction patterns derived from the values and principles of the Agile Manifesto [22] as well as theories and practices on team development (e.g., [23], [24]) build the basis to organize and run the mini-projects.

D. Practice: A Controlled Experiment

To achieve the learning goals, the practical parts follow the structure of *controlled experiments* [25] to ensure proper

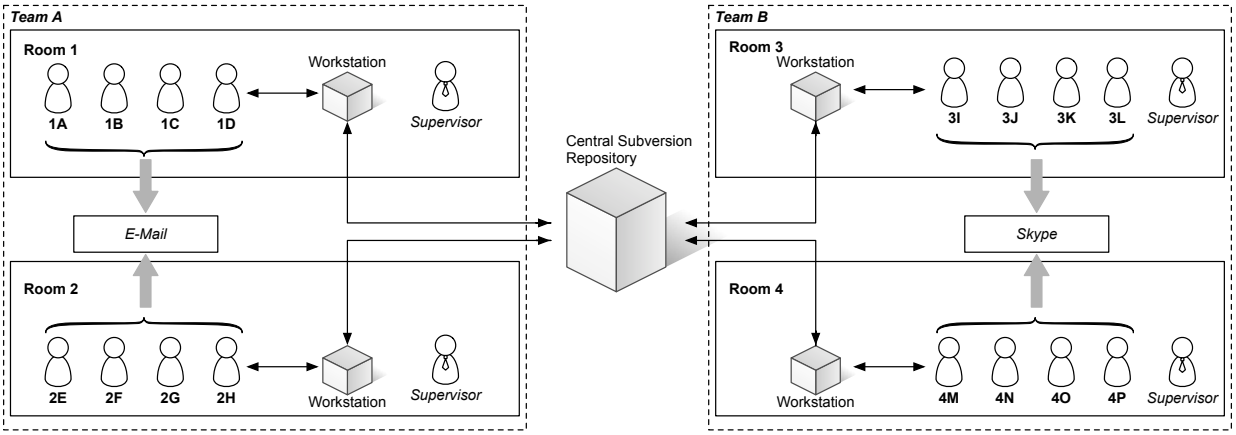

Fig. 2. Team setup for the distributed development exercise: two virtual teams develop a small Java application each, whereby the available resources (i.e., workstations) and communication channels are limited. The distributed project was simulated by separating the teams (one group per room).

TABLE I
OVERVIEW OF THE DATA COLLECTED (C: DATA BASED ON CODE SUBMITTED TO A REPOSITORY; Q: DATA FROM A QUESTIONNAIRE).

Question	Scope	Scale
C: *Code-based Analysis*		
1) Is the user story worked on?	DoD	YN
2) Is the user story implemented?	DoD	YN
3) Are the user's options displayed?	DoD	YN
4) Is the menu structured appropriately?	DoD	YN
5) Is the any new (added) information also displayed in other tasks?	DoD	YN
Q: *Team-related Questions*		
Goals and tasks were properly discussed in the team	R	LS
The decision-making process in the team was efficient	R	LS
The communication in the team was open	R	LS
If there were problems in the team, they were handled properly	R	LS
I was always an integrated part of the team	R	LS
The overall team performance was	R	LS
The atmosphere in the team was	R	LS
Goals and tasks were properly discussed in the virtual team	T	LS
The decision-making process in the virtual team was efficient	T	LS
The communication in the virtual team was open	T	LS
If there were problems in the virtual team, they were handled properly	T	LS
I was always an integrated part of the virtual team	T	LS
The overall virtual team performance was	T	LS
The atmosphere in the virtual team was	T	LS
Q: *Individual Perception/Rating (per student, personal rating)*		
Drawback #1, Drawback #2, Drawback #3		FT
Positive Aspects		FT
Negative Aspects		FT
Other things I want to say		FT

organization of the different treatments. As shown in Figure 1, the experiment session is carried out in a 4-hour block, which is specifically prepared in class. The experiment's overall organization is illustrated in Figure 2. In subsequent paragraphs, we provide details on the setup.

a) Task: In order to focus on the effects communication has on performance and quality, we defined a task, which was introduced to the students as follows: *You are asked to develop a Java **console application** that can manage user stories. The following tasks are to be implemented in the given order*[1]*. The stories do **NOT** need to be persisted when quitting the program. For the implementation of the tasks Team A1 and Team B1 will focus on developing the **user interface** (console user interface), whereas Team A2 and Team B2 will implement the logic.*

Right here, we have to mention that the task described above is a *psychological trick*. By asking the students to develop as many user stories as possible and by defining a basic project organization, we mimic the "normal" lab pattern and draw the students' attention to the code to be delivered and away from the actual exercise goals [18]. The purpose is to build awareness of the actually essential problems, and that ignoring those seriously impacts the overall performance and quality—making this assignment a *failure by design* project with only little chances to succeed (see Sect. III-D).

b) Data Collection and Analysis: In this task, we collected quantitative and qualitative data. Quantitative data was collected using submitted source code and questionnaires. This data was used as performance and quality measure. As part of the questionnaires, also qualitative data was collected, e.g., to work out perceived difficulties (Table I[2]). Data collection was

[1]The task comprises in total 18 user stories and five criteria forming the *Definition of Done* (DoD). The user stories describe a system to manage user stories for an agile software project. The DoD, among others, comprises menu options and structure, tested functionality, or evolving features in the incremental development (feature population).

[2]For the team-related and individual data, questions were scoped to the group level (room (R), cf. Figure 2) and to the team level (T). Scales and data types in the questionnaire ware free text (FT) or 5-point Likert scales (LS) with 1=fully disagree (or very poor) to 5=fully agree (or very good), and 3 as neutral value. Delivered code was rated according to the *Definition of Done* (DoD) using yes/no (YN) decisions.

TABLE II
RESULTS OF THE COMPLETION OF THE DEVELOPMENT TASKS PER COMMUNICATION VEHICLE ACCORDING TO THE DOD.

User Story	Skype					E-Mail				
	1	2	3	4	5	1	2	3	4	5
01	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
02	✓	✓				✓	✓	✓	✓	✓
03	✓					✓	✓	✓	✓	✓
04	✓					✓	✓	✓	✓	✓
05	✓					✓	✓	✓	✓	✓
06	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
07	✓		✓	✓	✓	✓	✓	✓	✓	✓
08	✓					✓				
09										
10						✓				
11						✓				
12										
—										
18										

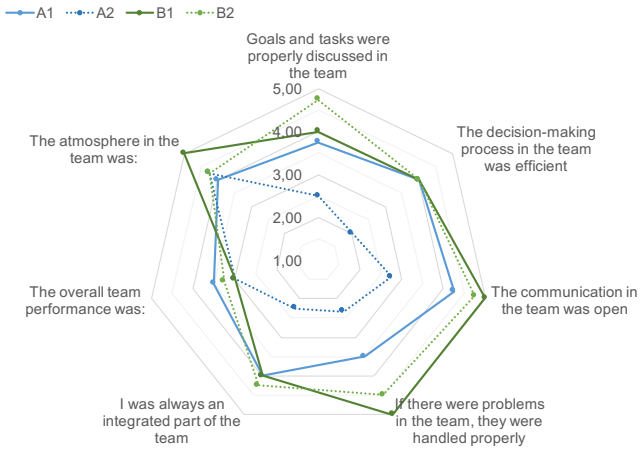

Fig. 3. Team-related questions (group level); average numbers per groups in different rooms (Figure 2 and Table I).

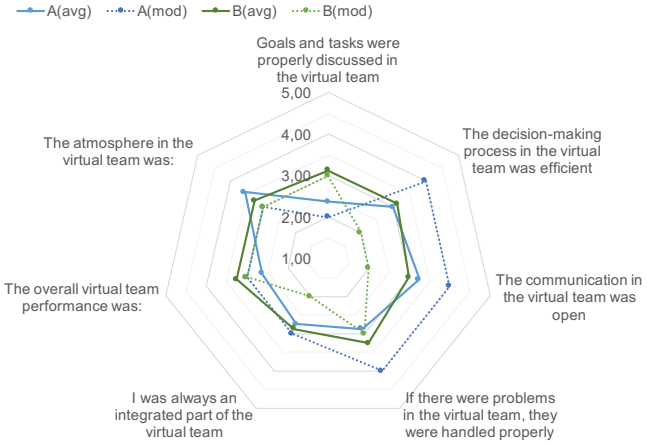

Fig. 4. Team-related questions (team level); average and modes per virtual team (Figure 2 and Table I).

carried out after the session, while the observers wrote minutes (team, time-stamp, observation). All data was transcribed into a spreadsheet for further analyses.

c) Execution: The "experiment" was conducted in the configuration from Figure 2 at the Technische Universität München. In total, 16 graduate students from the Software Engineering program participated in the session. The session, including set up and feedback, took approximately four hours, whereas the coding part was a 90-minutes time-box.

III. RESULTS

We present the data collected in class to set the scene before discussing lessons learned. The presentation of the data follows the structure from Table I, i.e., Sect. III-A is concerned with the code delivered, Sect. III-B presents results regarding the (perceived) team collaboration, Sect. III-C describes the experiences students made and discusses their (individual) perceptions and, Sect. III-D discusses our lessons learned.

A. Team Performance based on Code Delivered

Students were given 18 user stories, which were assessed given the *Definition of Done* (DoD) as presented in Table I. Table II provides an overview of the user stories implemented by the different teams. The table shows that the Skype team (Team B) started working on eight user stories. After 90 minutes, three user stories were (fully) implemented and tested. Team A (e-mail only) started to work on 10 user stories and, implemented seven. Analyzing the plain performance, the Team A worked more efficiently than Team B, as this team implemented more than twice the number of user stories. However, as "pure" performance is not in the focus of this exercise, in the next sections, we review the students' perception of the mini-project.

B. Team Collaboration

We provide two perspectives to investigate the students' perception: Figure 3 provides an overview of the team-related questions (Table I, scope R). The second perspective is given in Figure 4 in which we evaluate the ratings concerned with the overall virtual team (Table I, scope T). These questions were presented as statements to be rated on a Likert scale.

Figure 3 shows a generally higher agreement with the given statements in Team B (Skype). Furthermore, the (average) values' deviation is not as high as in Team A. On the other hand, the two groups of Team A show a remarkable deviation in the average values, especially regarding the efficiency of decision-making processes, open communication, and "team spirit" (*I was always an integrated part...*). That is, especially the ratings from Team A indicate to a higher (communication) distance between the groups within the virtual team, which was limited to e-mail as communication vehicle. Contrary, Team B that used Skype did not show a large distance.

TABLE III
SUMMARY OF THE DRAWBACKS IDENTIFIED BY THE STUDENTS (NOTE: 8 STUDENTS PER TEAM; 4 PER GROUP).

	Mentions	
Drawbacks	**E-Mail**	**Skype**
Communication	3	2
Time	3	2
Architecture and interfaces	4	2
Development environments, language and resources	5	5
Distributed team coordination	5	4
other		*2*
Total mentions	20	17

While Figure 3 provides an overview of the individual groups, Figure 4 provides the overview of the students' ratings regarding the overall atmosphere and performance in the virtual team as a whole. For both teams, the figure provides the average values and the mode values. The average values show both teams agreeing on efficient decision-making processes, open team communication, problem handling, team spirit, and general atmosphere. However, team performance as well as the discussion of goals and tasks were rated better in Team B. Considering the mode values, Figure 4 shows a different picture. For instance, in Team A, the majority of the team members considered the decision-making process, open communication, and problem handling in the team more positive than average, which indicates to a strong sub-team structure in which teams worked in a more self-contained or isolated style, and the individual teams experienced the work pattern differently. Team B however shows the exact opposite and, moreover, the ratings indicate that several team members did not feel themselves being part of a team.

C. Individual Experiences

Table I (individual perception/rating) summarizes the questions for the individual perceptions of the students regarding the mini-project. In particular, we were interested into the major drawbacks the students experienced. For this, we transcribed the questionnaire data and categorized the drawbacks mentioned.

Table III summarizes the drawbacks in the categories drawn from the students' answers and shows the development environment, (programming) language and resources to be the biggest pain points in both teams. Furthermore, the coordination of the distributed team was considered paining the teams, as well. At the same time, (general) communication issues or time pressure were not considered affecting the teams' overall performance. The numbers summarized in Table III thus show the students were more concerned with the technical infrastructure and the general team coordination rather than with communication within and among the teams.

D. Discussion and Lessons Learned

In this section, we relate the findings to each other, discuss the findings and, discuss the lessons learned from implementing the course unit presented in the paper at hand.

1) Findings in the Context: Putting all the findings together, we first have to admit that our expectations were not met: Initially, we expected the Skype team to have a better performance, which was not the case. In fact, the overall team performance (taking delivered code as reference) of the e-mail team was significantly higher. In the e-mail team, we explicitly found a more isolated way of work, however, also in the Skype team, we could find "lost souls". For instance, in the Skype group 1 (room 3), we received the positive comment *"... each person had something to do and we could work in parallel... "*, whereas in the other group one student mentioned: *"... the setup made my participation in the tasks itself unnecessary. I sat by doing nothing most of the time."*, and another one from group 1 complained: *"... backend used 30min for modeling without communicating it... "* That is, even though the overall mood in the team was perceived positive, some problems in the communication were observed. The complaints of the e-mail teams were not that frequent. One student mentioned: *"sometimes too much communication"*, yet the major pain point was the *"preparation of the logistics"*.

In a nutshell, students realized that "something" was affecting their work and they sporadically complained about communication, yet they did not make communication a first-class citizen, i.e., did not name communication issues as (potential) root cause for the trouble experienced.

From the perspective of the client, the assignment goals were only partially achieved, as the teams were unable to deliver solutions of sufficient functionality and quality. In the feedback session, we decided to play it hard and asked the students for explanations of the project failure. We got the answers from the questionnaires (see above), but rejected those statements and asked for the "real" reason, which made the students re-think the situation. Eventually, we resolved the situation by uncovering the *hidden agenda* and stating that the projects failed within the first five minutes of work and, to emphasize this statement, we asked the students: *"Did you agree upon a communication protocol, i.e., who talks when to whom and about what?"* All groups realized this issue and, in consequence, that they were "just" working somehow to compensate for this mistake in the initial project set up phase eventually reaching a certain level of frustration and unable to deliver proper solutions.

2) Lessons Learned: The implemented course unit proved an adequate instrument to (i) demonstrate students the importance of proper communication, (ii) work under stress with growing frustration and the inability to solve problems directly, and (iii) to give the students to opportunity to fail and analyze the reasons for their failure. In fact, we confronted the students with a situation, where basically only two approaches help to survive: (i) setting up the project with proper communication protocols or (ii) accepting the resource limitations and finding a "hero" in the teams doing the work while the others only provide support (one group was close to this approach).

The course unit was easy to set up, yet has some organizational constraints, e.g., availability of rooms and observers. Due to the task's nature, we argue that it is applicable to any

specific form of limited communication, e.g., e-mail, Skype, instant messaging, and so forth.

IV. Conclusion

In this paper, we presented a course unit to improve students' awareness of the role of communication in distributed software development. The course unit presented was implemented in an advanced course on Agile project management and software development. Students are asked to implement a small application in virtual teams that face, among others, limitations regarding the communication channels allowed. Therefore, the major challenge for the students is to overcome the limitations and to set up a working project team quickly to develop the application. Results from the initial run show that students being more concerned with technical issues, such as architecture and development resources, yet—to a large extent—ignoring the fundamental issues resulting from poorly organized communication. For instance, only three out of eight students from the team that was allowed to only use e-mail mention (slow) communication an issue, and only two students from the Skype team complain about communication. However, five (e-mail) and four (Skype) students mention serious problems regarding the overall team organization and coordination. We therefore could improve the students' understanding about *symptoms* in projects and (identifying) their actual *root causes*. In the unit's feedback session, students learned about the fundmental mistake they made in the project start and realized (and experienced) the consequences.

The presented course unit provides a simple means to teach students the role of communication by experiencing a hard-to-solve situation. As the presented setup was implemented only once, there is room for improvements. For instance, extended setups could cover more communication vehicles (or combinations thereof) or other facets improving the task's difficulty, e.g., interoperability of the solutions across teams. This, however, remains subject to future work.

Acknowledgement

We want to thank Henning Femmer and Jonas Eckhardt for their significant support in setting up and running the experiment. We also thank our colleagues, who spent time serving as group observers. The APM course was supported in part by the "Ernst Otto Fischer" teaching award/grant 2012 by Faculty of Informatics of Technische Universität München.

References

[1] C. Ebert, M. Kuhrmann, and R. Prikladnicki, "Global software engineering: An industry perspective," *IEEE Software*, vol. 33, no. 1, pp. 105–108, Jan 2016.

[2] A. Begel and N. Nagappan, "Global software development: Who does it?" in *International Conference on Global Software Engineering*. IEEE, Aug 2008, pp. 195–199.

[3] B. Al-Ani and H. K. Edwards, "A comparative empirical study of communication in distributed and collocated development teams," in *International Conference on Global Software Engineering*. IEEE, Aug 2008, pp. 35–44.

[4] V. Casey and I. Richardson, "Project management within virtual software teams," in *International Conference on Global Software Engineering*. IEEE, Oct 2006, pp. 33–42.

[5] K. Stapel, E. Knauss, K. Schneider, and N. Zazworka, "Flow mapping: Planning and managing communication in distributed teams," in *International Conference on Global Software Engineering*. IEEE, Aug 2011, pp. 190–199.

[6] T. Niinimäki and C. Lassenius, "Experiences of instant messaging in global software development projects: A multiple case study," in *International Conference on Global Software Engineering*. IEEE, Aug 2008, pp. 55–64.

[7] T. Jaanu, M. Paasivaara, and C. Lassenius, "Near-synchronicity and distance: Instant messaging as a medium for global software engineering," in *International Conference on Global Software Engineering*. IEEE, Aug 2012, pp. 149–153.

[8] Y. Dittrich and R. Giuffrida, "Exploring the role of instant messaging in a global software development project," in *International Conference on Global Software Engineering*. IEEE, Aug 2011, pp. 103–112.

[9] K. Dullemond, B. van Gameren, and R. van Solingen, "Virtual open conversation spaces: Towards improved awareness in a gse setting," in *International Conference on Global Software Engineering*. IEEE, Aug 2010, pp. 247–256.

[10] T. Niinimaki, A. Piri, C. Lassenius, and M. Paasivaara, "Reflecting the choice and usage of communication tools in gsd projects with media synchronicity theory," in *International Conference on Global Software Engineering*. IEEE, Aug 2010, pp. 3–12.

[11] P. Tell and M. A. Babar, "Activity theory applied to global software engineering: Theoretical foundations and implications for tool builders," in *International Conference on Global Software Engineering*. IEEE, Aug 2012, pp. 21–30.

[12] T. Niinimaki, A. Piri, and C. Lassenius, "Factors affecting audio and text-based communication media choice in global software development projects," in *International Conference on Global Software Engineering*. IEEE, July 2009, pp. 153–162.

[13] N. Mullick, M. Bass, Z. Houda, P. Paulish, and M. Cataldo, "Siemens global studio project: Experiences adopting an integrated gsd infrastructure," in *International Conference on Global Software Engineering*, ser. ICGSE. IEEE, Oct 2006, pp. 203–212.

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

[15] R. Prikladnicki and L. Pilatti, "Improving contextual skills in global software engineering: A corporate training experience," in *International Conference on Global Software Engineering*, ser. ICGSE. IEEE, Aug 2008, pp. 239–243.

[16] C. Deiters, C. Herrmann, R. Hildebrandt, E. Knauss, M. Kuhrmann, A. Rausch, B. Rumpe, and K. Schneider, "Glose-lab: Teaching global software engineering," in *International Conference on Global Software Engineering*, ser. ICGSE. IEEE, Aug 2011, pp. 156–160.

[17] F. Fagerholm, N. Oza, and J. Münch, "A platform for teaching applied distributed software development: The ongoing journey of the helsinki software factory," in *Intl. Workshop on Collaborative Teaching of Globally Distributed Software Development*. IEEE, 2013, pp. 1–5.

[18] 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*, 2016.

[19] M. Kuhrmann, H. Femmer, and J. Eckhardt, *Overcoming Challenges in Software Engineering Education: Delivering Non-Technical Knowledge and Skills*. IGI Global, 2014, ch. Controlled Experiments as Means to Teach Soft Skills in Software Engineering, pp. 180–197.

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

[21] M. Kuhrmann, D. M. Fernández, and J. Münch, "Teaching software process modeling," in *International Conference on Software Engineering*, 2013, pp. 1138–1147.

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

[23] B. W. Tuckman, "Developmental sequence in small groups," *Psychological Bulletin*, vol. 63, no. 6, pp. 384–399, 1965.

[24] N. Gorla and Y. W. Lam, "Who should work with whom?: Building effective software project teams," *Communications of the ACM*, vol. 47, no. 6, pp. 79–82, Jun. 2004.

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