Virtual by Design: How a Work Environment can Support Agile Distributed Software Development

Pernille Lous
IT University of Copenhagen
Copenhagen, Denmark
pelo@itu.dk

Paolo Tell
IT University of Copenhagen
Copenhagen, Denmark
pate@itu.dk

Christian Bo Michelsen
IT University of Copenhagen
Copenhagen, Denmark
chmi@itu.dk

Yvonne Dittrich
IT University of Copenhagen
Copenhagen, Denmark
ydi@itu.dk

Marco Kuhrmann
Clausthal University of Technology
Clausthal, Germany
kuhrmann@acm.org

Allan Ebdrup
Debitoor
Copenhagen, Denmark
aeb@debitoor.com

ABSTRACT

Even though agile methods have been flourishing in the last decades, their implementation in (globally) distributed arrangements still present hard challenges. Due to this tension, practices are either modified or added to compensate with the additional control required by the setup. In this paper, we present a case study about a company that managed to incrementally design a process that does not compromise the foundations of the agile philosophy by embracing the characteristics of distributed development. We show how a virtual work environment has been crafted by continuously improving practices and carefully selecting technologies to allow each team member to fully participate regardless of the actual physical location. Aware of the single nature limitation of the reported case, we present extensive information to frame the context allowing meaningful comparisons by researchers and providing concrete examples for practitioners.

CCS CONCEPTS

• **Software and its engineering** → **Agile software development**; **Programming teams**;

KEYWORDS

Agile development, global software engineering, virtual teams

ACM Reference Format:
Pernille Lous, Paolo Tell, Christian Bo Michelsen, Yvonne Dittrich, Marco Kuhrmann, and Allan Ebdrup. 2018. Virtual by Design: How a Work Environment can Support Agile Distributed Software Development. In *ICGSE '18: ICGSE '18: 13th IEEE/ACM International Conference on Global Software Engineering , May 27–29, 2018, Gothenburg, Sweden.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3196369.3196374

1 INTRODUCTION

Global software engineering (GSE) and agile software development (ASD) are streams in the software development industry that are increasingly gaining momentum—notably in their combination *Agile Distributed Software Development* (ADSD) from hereon. Yet, this combination has shown to be fairly complicated to implement [23, 32] if the core of the followed process ought to be adhering to the agile values and principles expressed in the Agile Manifesto [1]. Even though evidence in the literature provides examples of cases combining GSE and ASD, implementations that do not compromise the agile philosophy and achieve the extra level of control required by distributed development are rare. Specifically, cooperation in GSE is considered challenging due to the limited communication, the harder coordination, the inadequate collaboration, and the insufficient awareness among the sites involved in the endeavour [13, 16]. Independent of whether the team is distributed, dispersed, or partially dispersed [38, 39], geographical, temporal, cultural, and linguistic distances are characteristics hampering cooperation that need to be alleviated by special precautions [6, 21, 23].

The case reported in this paper describes a different approach: instead of compensating for the distribution, the whole work environment has been crafted to embrace the distributed setup.

Case Subject. Debitoor has 40 employees, and its business is based on an online invoicing and accounting software solution for small businesses and companies. Fourteen employees, which are distributed across four countries in Europe, are directly involved in the development of the product and represent our case subject. This study does not focus on the system architecture (microservices in the specific case), as this decision was never challenged. Instead, we focus on the team and study the practices and the ecosystem of tools used by the virtual development team.

Problem Statement and Objectives. Our research aims to study the virtual work environment at Debitoor and to show how the challenges of ADSD reported in the literature (as aggregated in [23]) are avoided by design. In particular, we are interested in answering the following research question: *"How can a work environment support agile distributed software development?"*

Contribution. By reflecting on a one-year long investigation of Debitoor, we provide insights regarding the practices, the ecosystem of tools used, and their configuration, which showcases how a work environment can be crafted to support a virtual agile distributed software development team. Our study provides inspiration to practitioners and—as a case study reporting a successful implementation of a virtual work environment—provides detailed information about the context to lay the foundation for future research.

Outline. The remainder of this paper is organized as follows: Section 2 presents related work, and Section 3 describes the research design. Results are presented in Section 4, before we discuss implications in Section 5.

2 RELATED WORK

Agile software development in GSE constitutes numerous challenges regarding proper implementation of key components of the agile philosophy. For instance, in their literature review, Lous et al. [23] study whether agile methods—specifically Scrum—are fit for GSE. They name 45 challenges in 19 categories that practitioners face when using Scrum in distributed settings. Challenges have been identified in the Scrum "core process" (e.g., on-/off-site Scrum Master, team cohesion, and attendance in meetings), as well as in the "extended practices" (e.g., cultural and technical challenges, stakeholder separation, and distributed requirements engineering). Lous et al. [23] conclude that scaling Scrum for GSE is the most challenging problem. A similar conclusion is drawn by Paasivaara et al. [32], who study a large-scale agile transformation at Ericsson. Among other things, authors conclude that larger teams might have an increased need for specialization, which limits team interchangeability and, thus, flexibly of staffing, and they also consider out-of-the-box agile frameworks inappropriate. In this regard, Diebold et al. [9] provide a study that particularly investigates the actual way of adopting Scrum to specific a context. They find Scrum barely used by the book, but many practitioners modifying Scrum on-demand. A broader perspective is provided by Kuhrmann et al. [22]. In their study, authors investigate the use and combination of different development approaches. Their major findings comprise that independent from size or industry sector, companies combine different traditional and agile development methods/practices. Participants in the study mentioned project/product management and commitment, and improved project flexibility (across project sites) as major drivers. These exemplarily selected studies concern the process perspective and how distributed teams approach these challenges. In 2014, an IEEE Software special issue on virtual teams [39] discussed the particularities of work patterns, challenges coming along with virtual teams, and experiences from implementing virtual teams [10, 14].

Modern technology helps distributed virtual teams collaborating. For instance, different communication technologies like video conferencing or instant messaging help creating a virtual work environment [29]. Yet, building awareness regarding "real-time" distributed workspaces is a topic discussed since the mid 1990's [19]. In 2012, Portillo-Rodríguez et al. [34] conducted a systematic review to develop a big picture of tools and technologies used in GSE. In total, they identified 132 tools that have been used in distributed software projects mainly addressing communication, coordination, and control. That is, tools used to compile a distributed workspace mainly address the issues coming along with distributed team setups, notably concerned with enabling teams to collaborate across different sites. Yet, most of the tools address specific issues, such as management of artifacts in distributed projects [4, 15], collaborative software development practices, e.g., using pair-programming [20, 44], dashboards [3], and many more.

In their systemic review, Šmite et al. [43] conclude that *"majority of the studies represent problem-oriented reports focusing on different aspects of GSE management rather than in-depth analysis of solutions for example in terms of useful practices or techniques"*. A reflection on the ICGSE conference series by Ebert et al. [12] provides an overview of the fields of interest finding *project management*, *collaboration and teams*, and *processes and organization* the most frequently addressed issues. The study at hand thus contributes to the body of knowledge by providing experiences of *"solutions [...] of useful practices"* using a practical case on processes and tools and their integration in a virtual workspace in distributed teams.

3 RESEARCH DESIGN

Given the exploratory nature of this research, a mixed-method approach with a concurrent triangulation strategy *"to confirm, cross-validate, and corroborate findings"* [11] was chosen. As suggested by Miles et al. [25], we collected data from a single subject through observations, interviews, questionnaires, and off-site data collection. Table 1 presents a summary to characterize the empirical context to ease future comparisons. The taxonomy has been adapted from the one suggested in [42]. Further details on the data collection procedures are presented in Section 3.1, and the data analysis procedure is presented in Section 3.2. A brief description of the study subject is presented in Section 3.3. Finally, Section 3.4 discusses threats to validity and measures taken to improve the validity of our study.

Table 1: Characterization of the empirical context.

Attribute	Value
Year	2017
Empirical focus	Empirically based
Empirical background	Industry
Industry sector	Accounting
Subject of investigation	Practitioners
Study results	Successful practices
Empirical research method	Case study
Source of empirical evidence	Observ., interview, survey
Location	Offshore
Legal entity	Outsourcing (insourcing)[1]
Geographic distance	Distant (within Europe)
Temporal distance	Small (1 hour max)
# of sites	4 (2x Denmark, Ukraine, Lithuania)
Team size	14 (12 developers, 1 PO, 1 CTO)

3.1 Data Collection

The data collection was performed in 12 visits to Debitoor between November 2016 and October 2017. During this period, the research team was granted access to the entire ecosystem of tools used by

[1]Note: Debitoor's business arrangement with the Ukraine part of the team is outsourcing. However, all members of the development team consider the relation much more in line with what would be expected in an insourcing arrangement.

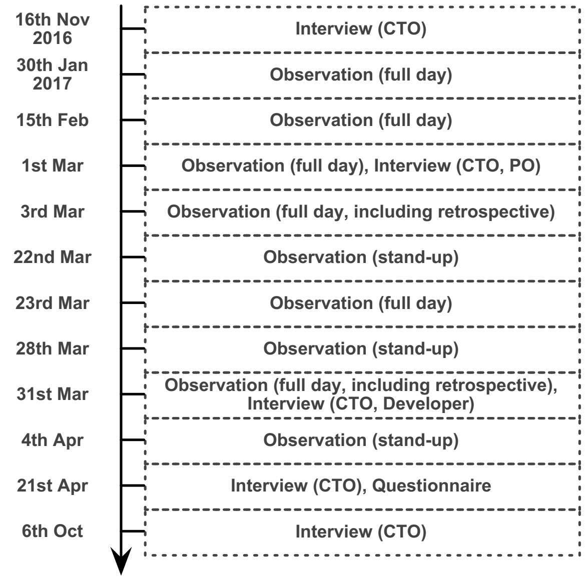

Figure 1: Overview of the research activities performed during the study.

the development team: Slack, Google Sheet/Slide, Github, and the Waffle.io board (see Table 4). Figure 1 visualizes a breakdown of the different activities, i.e., observations, interviews, and a survey.

3.1.1 Observations. Nine observations were conducted at the Danish site. Three visits were fully concerned with the 9:15 CET stand-up meeting. The remaining six full-day observations included the stand-up meetings, an observation of the daily work practice in the company, and we also observed two full retrospective meetings. Debitoor and the members of the development team had been open to collaboration and we were allowed to join the team in any activity. This included lunches and coffee breaks, which were extremely useful to engage in one-on-one conversations with specific individuals to clarify and discuss any interesting observation. Field notes were taken during all observations and were re-written after the observation ended according to the process described in [25].

3.1.2 General Interviews. In total, seven semi-structured interviews have been performed throughout the research period. One with the product owner (PO), one with a developer from Ukraine, and five with the CTO. Semi-structured interviews were chosen to ensure the openness of the conversation, while allowing the topics of interest to be fully explored [36]. All interviews have been carried out using the interview guide proposed by Yin [45]. Each interview lasted about one hour and was recorded for post analysis. All but one interview, which was performed through a Slack-call (Ukraine developer), were conducted face-to-face in a meeting room at the Danish site. Besides discussing general aspects of the work practices at Debitoor, the interviews were all organized with key members to explore recurring insights captured during the observations. An exception was the first interview, as it was the project kick-off in which more general aspects of Debitoor were explored and discussed.

Table 2: Overview of the questionnaire (questions and data types collected).

Q1	*Do you perceive the developers in Kiev/Copenhagen as one or two teams?* [Multiple choice: 1 team, 2 teams, free text]
Q2	*Please rate your satisfaction with regards to the amount of interaction between Kiev/Copenhagen during working hours.* [1: I would prefer less interaction; to 5: I would like to have more interaction than we already have]
Q3	*If the entire team would be located in one site: how much would you change the way of doing stand-ups/retrospective?* [1: I would change a lot and make it more co-located; to 5: I would not change anything. I prefer the way it is now]

3.1.3 Short Interview. To understand the particularities of the communication network within the development team, a short interview was performed with each member of the team to capture the perceived communication paths. This interview comprised one question only: *"Who do you contact directly regarding work? This includes private messages on Slack, but excludes group channels, social interactions not regarding workor contact outside work."*

3.1.4 Questionnaire. An online questionnaire was designed to gather further details from the team members regarding their perception of specific topics. Besides other advantages, this collection technique allowed us to reach all sites of the development team equally: Denmark, Lithuania, Ukraine, and the single home office in Denmark. Table 2 shows the three questions of interest. Two of the questions use a 5-point Likert scale, and one question asks the participant to select options.

3.1.5 Off-site Data Collection. Finally, our main contact at Debitoor was available for off-site data collection via email throughout the study. This contact was leveraged several times using the approach suggested in [25].

3.2 Data Analysis

Based on the field notes, specific moments of the observations were selected for in-depth analyses. The audio recordings from this selection and all interviews were manually transcribed by two of the authors. The resulting transcription—including field notes—was coded using qualitative coding[2]. The coding was used to create clusters, which then were abstracted into themes. Once the data analysis of the field material was concluded, we obtained a detailed picture of the set of practices followed at Debitoor, the technologies utilized by the development team, and how the former are supported by the latter.

To better understand the implications of the findings from the observations, these have been related to the challenges faced in GSE. In particular, as presented in Section 2, we used the catalog presented in [23]. After consolidating the set of 45 initial challenges by removing duplicates and unifying closely related ones, we were left with 33 unique ones, of which, eight were used to explore how Debitoor manages to successfully designed their work environment using a hybrid method adhering to the agile philosophy

[2]The tool Atlas.ti was used to perform the coding activities.

with a virtual team by mitigating—if not ignoring—the challenges reported.

3.3 The Virtual Development Team at Debitoor

As introduced in Section 1, the subject of this study is the software development team of Debitoor, which consists of fourteen members distributed across Denmark (headquarter in Copenhagen), Ukraine, and Lithuania. Figure 2 visualizes the team setup of the studied case. Even though the business arrangement with the Ukraine team is—formally—outsourcing, all members of the development team consider the relation much more in line with what would be expected in an insourcing arrangement, which leads to a virtual team.

To better characterize the team: (i) together with seven developers, the product owner (PO) is located in Ukraine and travels regularly between Denmark and Ukraine; (ii) together with three developers, the CTO works in the Danish office, and he is considered to be a part of the development team as his daily work is mainly in service of it; (iii) one developer in Denmark works remotely from his home office and rarely visits the office in Denmark due to perfume allergy, hence, he is considered as an independent site (i.e., teleworking); and, (iv) one developers works remotely from Lithuania.

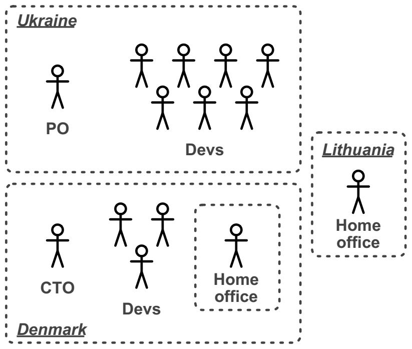

Figure 2: Overview of the development team of Debitoor.

3.4 Threats to Validity

We discuss the threats to the validity of this study following the basic scheme that distinguishes among construct validity, internal validity, external validity, and reliability [45]. *Internal validity* will be skipped, for with this study as we, based on qualitative data, not attempt to prove logical relationships among the concepts presented.

Construct Validity. Threats to construct validity were minimized by using different sources of data including observations, semi-structured interviews, project artifacts, and off-site data collection. As suggested by [25, 36], this triangulation allowed us to clarify and confirm insights by accessing a wide range of perspectives.

Reliability. To ensure reliability [45] and to reduce the risk of single-researcher bias, all field activities, but three interviews, were performed by at least two researchers. With regards to the data analysis, this was initially performed by two researchers until the coding scheme stabilized and, afterwards, meetings were arranged including often up to four researchers to discuss, clarify, and proceed in the generation of clusters and themes. Additionally, an open communication with the company contact was kept to verify constantly the findings, hence, decreasing the risk of misinterpretation.

External Validity. The main limitation of single case studies is their generalizability [45]. This study is no exception; however no statistical significance or generalizability of the results were sought. Our results and discussions are grounded in the case analyzed and are affected by the specific context. We have therefore provided detailed information of the context to allow future studies to replicate or compare results.

4 RESULTS

In this section, we describe the set of practices implemented at Debitoor. Section 4.1 introduces these practices that represent a unique way of working, which is not covered by any available standard process. In Section 4.2, we elaborate on the interplay between tools and practices that Debitoor uses. Additionally, a list of tools used by the development team to work on the code artifacts and cooperate among the distributed team members is presented in Table 4. To help the reader, in the remainder of this section, we highlight **practices** in boldfaced and *tools* in italic text.

4.1 Practices and Tools

The process model at Debitoor is significantly influenced by the agile philosophy, however it is not restricted to a specific method such as Scrum or XP. A schematic representation is provided in Figure 3, and each of its comprising practices is described in the following.

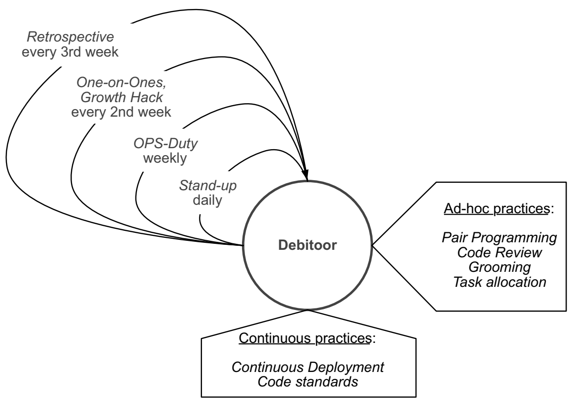

Figure 3: Schematic representation of the software process followed at Debitoor.

Table 3 provides a detailed description of the practices from Figure 3 followed at Debitoor. The practices are complemented by a set of tools Debitoor uses. These tools are summarized in Table 4. Beyond the practices from Table 3, Debitoor implements a rather uncommon approach to requirements engineering and division of labour. As there are no explicit iterations, deadlines are less relevant,

Table 3: Overview of the practices used at Debitoor.

Practices	Description
Three-week Cadence	In line with the 12th principle of the Agile Manifesto [1], which is related to the theme of 'inspect and adapt', the set of practices are reviewed at regular intervals to ensure that the team—part of this process—can be as effective as possible. In particular, the development team performs a **retrospective** meeting (inspired by Scrum) every third week.
Two-week Cadence	To ensure the well-being, satisfaction, and happiness of each team member, the CTO conducts **one-on-ones** lasting approximately half an hour every second week. These meeting are short conversations between the CTO and each member of the team. This practice is an opportunity to express dissatisfaction or concern due to events happening either inside or outside the company. Additionally, every second week, the development team holds **growth hacks**—events in which the developers are encouraged to work on anything they can conceive that could benefit Debitoor in any way.
Weekly Cadence	Each week, a new team member is assigned to **OPS-duty**. This practice has evolved over several iterations due to requests expressed during the retrospectives, and it is now involving all team members from all sites.
Daily Cadence	**Stand-up** is also a practice inspired by Scrum that is adopted at Debitoor. This ceremony happens daily at 09.15.
Continuous Practices	In line with the 3rd principle [1], which is related to delivery frequency, Debitoor invested a significant effort in setting up a complete **continuous deployment** pipeline, which includes, among others, automated checks to enforce the adherence to **coding standards** every time a commit is performed.
Ad-hoc Practices	In line with the 8th principle [1], which is related to technical excellence, several technical practices are in use at Debitoor. These occur on-demand, hence, triggered by team members when needed. In particular, quality assurance (QA) at Debitoor is entirely delegated to the development team. No team member or external team is dedicated to QA. Contrarily, team members can request when they deem necessary to be supported by a colleague either in real time through **pair programming** (practice taken from XP) or asynchronously via **code reviews**.

Table 4: Overview of the tools used in the studied ecosystem.

Tool	Description
Slack	At Debitoor, the enterprise version of Slack is the only tool used for communication within the team. This includes synchronous and asynchronous communication. The adoption of Slack completely removed the use of e-mails for internal communication. Slack supports four types of communication: calls (one-to one audio), channels (one-to-many text), private messages (PM) (one-to-one text), and screen sharing. In the remainder of this paper, these four types of communication will be kept separate to emphasize the feature of Slack used in each specific practice.
Waffle.io board	Waffle.io is an automated project management tool for GitHub that automatically updates changed tasks created in GitHub. The Waffle.io board is a task board that is used to keep track of the development process when a developer makes progress on a task. The stages used at Debitoor are: "Next Awesome Thing", "In Progress", and "In Production". Debitoor uses the paid version and, for historic reasons, Debitoor kept it even after project features became available on GitHub.
GitHub	GitHub is the choice for version control (paid business plan), and it is used on a daily basis by all the team members. This choice also impacted others, e.g., Waffle.io was picked due to its simplicity and functionalities of course, but mainly as it needed to be compatible with GitHub.
Google Products	At Debitoor, mainly two Google products are used: Google Sheets and Google Slides. As it will be more clear in the next section, these are primarily used to support retrospective meetings, stand-ups, and grooming. Minor usage of Google Sheet also includes activities like monitoring of licenses and similar less relevant purposes.
TeamCity	TeamCity is a continuous integration tool, and it is used at Debitoor for automated testing. This software is also used to check conformity of commits to code standards.
Mirror	Two monitors are set up in the two main offices, i.e., Denmark and Ukraine. The monitors are connected to a camera on the top which live-streams the office to the other site. The stream is running continuously to simulate a virtual window. Sound has been deactivated eventually, as it was considered too intrusive and disturbing.
Others	Google e-mails are used to create calendar invites as this coordination feature is not supported by Slack.

and these are in general not communicated to the development team. Exceptions to this practice only occur in the presence of a large marketing campaign for which specific features need to be ready. Besides these events, the only members of the team aware of upcoming deadlines are the CTO and the PO. Also related to the absence of iterations, planning and estimation of features are performed in unusual manners. In particular, **task management** is performed by the PO and CTO. Team members are organically assigned to tasks based on their knowledge or their wish to follow a feature, e.g., to improve their set of skills or their knowledge of the code base. Accordingly, **grooming** is performed by the PO in collaboration with the team members that will follow through the implementation of a specific task. *"We have this rule: the people who are going to do the tasks, they groom together. I mean, there is no reason to sit there and do grooming for an hour about something, that you should not be a part of yourself—CTO."*

4.2 Configuration

Based on the practices (Table 3) and the set of tools (Table 4) presented, we describe how the former are supported by the latter providing details on their use. This description will be grouped by categorizing the practices into three different groups: *technical practices*, *automation*, and *organisational practices*. Table 5 presents an overview of the observed interplay. To avoid repeating information, we start by describing an additional group representing the

Table 5: Tools used to support the different practices.

	Organizational						**Automation**	**Technical**		
	Stand-up	*Retro-spective*	*One-on-one*	*Task allocation*	*Grooming*	*Growth hacks*	*Continuous deployment [Coding standards]*	*Ops-duty*	*Code review*	*Pair programming*
Google Slide	✓			✓	✓					
Google Sheet		✓								
Slack call	✓	✓	(✓)	(✓)		(✓)			✓	✓
Slack channels						✓		✓		
Slack PM									✓	✓
Waffle.io board				✓						
Mirror				(✓)					(✓)	(✓)
GitHub						✓	✓	✓	✓	✓
Screen sharing									✓	✓
Calendar	✓	✓	✓							
TeamCity							✓			
FaceToFace			(✓)	(✓)		(✓)				

configuration used to support general communication both when scheduled and when performed ad-hoc.

4.2.1 General communication. Common to the majority of the practices is the configuration used to communicate. When required, team members would use their headset to participate to a *Slack call*, while sitting at their desk. Additionally, we observed two slight variations that, depending on the meetings, are either scheduled, recurring, or ad-hoc.

Scheduled or recurring meetings are almost always coordinated by blocking a time slot in the *calendar*. Once the time arrives, participants join an audio conference using *Slack calls* and, depending on the type of meetings, they might use other tools to provide additional shared workspaces like *Google Slide* or *Google Sheet*. This configuration was observed during stand-ups, retrospectives, and remote one-on-ones.

Contrary, ad-hoc meetings are triggered via *Slack private messages*. These meetings usually require access to the code base via *GitHub* and might require the *screen sharing* functionality of *Slack*. Finally, if the parties involved are located both in the Ukraine and Denmark, it is not uncommon for the initiator to use the *mirror* to verify whether the others are available at their desk: *"[...] if I am trying to reach someone and they are not answering then I check if they are sitting at their place. If they are not, okay then I have to wait; and if they are, I can tell one of the others: 'Can you please poke?'—CTO."* This configuration was observed during **pair programming** or **code review** sessions as well as **task allocation** meetings.

4.2.2 Technical Practices. Technical practices are those practices that directly involve the handling of code artifacts thus directly interacting with the product's code base through *GitHub*. These practices are: OPS-duty, code review, and pair programming.

OPS-duty involves the smallest set of technologies. As mentioned before, **OPS-duty** is a weekly rotating responsibility that requires a team member to be in charge of dealing with external requests for the development team, e.g., the customer support and other relevant teams at Debitoor. Such external groups have direct access to a *Slack channel* named #OPS-channel and, if for instance a serious issue was identified or reported by a customer, this issue is listed on the #OPS-channel to be taken care of by the development team member on **OPS-duty** this week.

The other two technical practices, i.e., **pair programming** and **code review**, share the same tool configuration. This includes *GitHub* and the *screen sharing* feature of *Slack*. The communication is established as a regular ad-hoc communication (see above). Because code reviews are not mandatory, it is well understood by all team members that, if a review is requested through a private message, it is important. According to Debitoor's internal culture, code reviews are the only reason that justifies the interruption of a colleague. Quoting the CTO: *"once they are done, it is up to the developers to do code reviews and ask for code reviews and do automated testing and do whatever it takes to make sure that this thing [the code] is production quality. We do not have any testers or QAs so they [the developers] make sure that the quality is great—CTO."*

4.2.3 Automation. A significant effort has been devoted to introduce **continuous deployment**. Currently, the deployment pipeline includes several steps that automate test execution and the adherence to **coding standards**. This is managed through *TeamCity*.

4.2.4 Organizational practices. Among the organizational practices, there are scheduled and ad-hoc events of which **stand-up** and **retrospective** meetings are particularly interesting. Even thought their label is inspired by Scrum, these practices have been evolved through extensive experimentation to fit the team's needs.

Stand-up meetings are conducted with developer sitting at their own personal computer wearing a headset. A *Google slide* presentation is used as main tool to drive the meetings. The duration of the meetings is carefully tuned to never exceed 15 minutes. The initial slide is automatically generated by using calendars data to make people aware of vacation periods as well as data from *GitHub* to alert the team about potentially critical issues in the system captured from mining error messages in the production logs. Finally, developers have the ability to add an individual slide to present interesting aspects of currently active tasks for knowledge sharing purposes: *"[...] so one of the principles in Debitoor is that we are not enforcing people a lot of things, right, they should share things by themselves—Ukraine developer."*.

Sharing the configuration with the stand-ups, **retrospective** meetings differ in that they use a *Google sheet* document rather than a slide set for knowledge management. The meeting is lead by the developer in Lithuania and comprises five steps:

(1) Assessing results based on the action points agreed upon during the previous retrospective.
(2) Sharing good and bad thoughts since the last meeting.
(3) Generating ideas on how to solve the bad thoughts identified in the previous step.
(4) Defining a plan on how to implement the ideas.
(5) Eventually assigning tasks to developers if necessary to ensure their execution.

One-on-one meetings differ slightly from other recurring events as their nature is more personal. The CTO prefers to perform them in a more relaxed fashion than through *Slack calls*. It is not uncommon, for instance, that meetings performed with the team members in Copenhagen are done *face-to-face* while having a walk along the shores of the lakes in Copenhagen.

The management of the backlog is left to the PO. When features are close to be handed over to the development team, the PO performs the final **grooming** by creating a presentation using *Google slides*. The presentation is sent to the team members that will take charge of the feature(s). This **task allocation** is either performed via *Slack calls* or with the PO addressing the specific team member(s) *face-to-face*. Results from the meeting are recorded on the *Waffle.io board*.

Finally, **growth hacks** are quite an exception. As mentioned before, these two-day events happen every other week and, due to their spontaneous nature, can be supported by any of the configurations presented here. They very often result in some development in parallel to the main *GitHub* product projects, and they might involve individuals or a group people. Depending on their needs, the people involved might decide to rely either on virtual tools to support communication or to rely on face-to-face communication. We do not have additional insights on this practice as they were not the main focus of our study. However, based on the interviews, there seems to be no preference regarding the communication channel related to the physical location.

Table 6: Assessment of tools related to the dimensions of cooperation.

	Communication	Coordination	Collaboration	Awareness
Google Slides	✓	✓	✓	✓
Google Sheets	✓		✓	✓
Slack call (headset)	✓			
Slack channels	✓	✓	✓	✓
Slack PM	✓	✓		
Waffle.io		✓		✓
Mirror				✓
GitHub		✓	✓	
Screen sharing	✓		✓	
Calendar		✓		✓
TeamCity		✓		

Table 7: Communication network at Debitoor. Developers are identified with: D for Denmark, U for Ukraine, and L for Lithuania. The darker gray cells show how to read U1 as an example.

	CTO	*PO*	*L1*	*D1*	*D2*	*D3*	*D4*	*U1*	*U2*	*U3*	*U4*	*U5*	*U6*	*U7*
L1	✓	✓		✓	✓	✓	✓		✓		✓	✓		
D1	✓	✓						✓	✓	✓	✓	✓	✓	✓
D2	✓	✓				✓	✓	✓	✓	✓	✓	✓		
D3	✓	✓					✓	✓	✓		✓			
D4	✓	✓						✓	✓			✓		
U1	✓	✓							✓	✓	✓	✓	✓	✓
U2	✓	✓								✓	✓	✓	✓	✓
U3	✓	✓												✓
U4	✓	✓										✓		✓
U5	✓	✓												✓
U6	✓	✓												✓
U7	✓	✓												

5 DISCUSSION

This section presents three main observations made at Debitoor. Section 5.1 challenges the assumptions that a co-located work environment is better than a virtual one; Section 5.2 provides an initial validation by relating known challenges identified in the literature against Debitoor's environment; and, Section 5.3 highlights the importance of embedding in a company culture a continuous improvement focus.

5.1 Co-located vs. Virtual Work Environment

In cooperative work [37], for the purpose of collaborating on the shared code base, individuals involved need to coordinate their actions. This coordination will eventually entail engaging in direct communication for alignment and agreement or for solving doubts, misunderstandings, and, in general, lack of clues, when one of them is not sufficiently aware of what is required to work on the product. Hence, cooperation links together four terms: communication, collaboration, coordination, and awareness [13, 40]. These terms emphasize high-level requirements that a work environment needs to provide to improve cooperation in distributed software development—either through practices or software tools.

Focusing on communication as a dimension of cooperation, we argue that even though media richness theory [8] identifies face-to-face communication as the channel through which the maximum throughput of information can be obtained, task/technology fit [18] explains how a leaner media can be a better fit given a specific task. Therefore, a co-located environment in which face-to-face communication is possible, does not necessarily represent the most suited environment to support a software team. Practice has shown that communication media are used opportunistically by switching between rich and lean media, e.g., [5, 30].

Expanding this observation to the remaining dimensions of cooperation and to cooperation itself, it could be hypothesized that the traditional co-located work environment is not necessarily the ideal setup for group work when considering software development. We argue that currently available tools can be used to support carefully selected practices, making the resulting virtual work environment extremely effective for supporting cooperation in a specific context. We therefore expand Berczuk's idea [2] that *"agile is about people, but distributed agile requires good tools to help people communicate effectively over distances"* by stressing the fact that the tools used should be facilitating all dimensions of cooperation and should support ad-hoc practices. Table 6 relates the set of tools used at Debitoor with the dimensions of cooperation just discussed, by highlighting dimension(s) mostly supported by the tool.

Debitoor opted for the primacy of a virtual work environment over a co-located one. It is important to emphasize that Debitoor's environment is the result of conscious decisions often experimented and carefully tweaked, rather then a compromise that had to be accepted to allow the virtual development team to cooperate. As a result, the team at Debitoor does not feel separated by physical boundaries *(Q1. N=7. One team: 5; Other: 2)* and, if anything, when asking the team members, they might identify smaller subgroups focused on development areas in the product: *"we have several teams based on what we're working on, location doesn't matter that much—Anonymous team member."* To further investigate the communication network, we asked team members to state their perceived interactions within Debitoor. Table 7 shows the density of the communication network, and the table confirms that physical boundaries are not hampering communication.

Question *Q2* also confirms satisfaction with regards to the amount of interaction across the two main geographical locations *(Q2. N=7. Neutral (3): 6; I would like to have more interaction than we already have (5): 1).* Interestingly, challenging the team members and proposing a co-located alternative to their implementation of the stand-up and retrospective meetings, no respondent considered the absence of physical separation a problem *(Q3. N=7. Neutral (3): 2; (4): 1; I would not change anything. I prefer the way it is now (5): 4).*

5.2 Addressing Known Challenges

A holistic approach to provide an initial validation of the hypothesis that a virtual work environment can be designed to support an agile development team that is distributed across different sites is to assess the case of Debitoor against known problems in distributed agile software development identified in the literature. As mentioned in Section 2, Lous et al. [23] systematically identified challenges. Based on their results, we identified eight challenges (from the 45) related to the virtual work environment. Table 8 summarizes the challenges and relates them solutions. Specifically, for each challenge, we describe the impact according to the reference in literature that named and studied this specific challenge. Challenges and respective impact are related to the solution approach implemented at Debitoor. We discuss how Debitoor addressed specific challenges by virtual work environment, i.e., the set of practices and supporting tools, or even entirely removed a challenge.

One of the most interesting aspects of how Debitoor has set up the virtual work environment relates to communication and the culture around it. Regardless of whether it is a one-to-one or a many-to-many communication, regardless of whether it involves co-located or distributed people, communication is always mediated through tools. Team members are comfortable with executing any kind of communication while sitting at their own desk with a personal headset. As mentioned in Section 4, only one exception exist: the **one-on-one** meetings. These meetings address personal issues, and executing them in an environment outside the team space is generally preferred.

Debitoor's setup effectively creates a work environment in which physical presence is not required to participate to any project activity. As a result, meeting rooms or cost-intensive video-conferencing systems are not required, which reduces the cost of running the virtual team. Most of the infrastructure used—and *accepted*—by the team (see Table 4) are low-cost or even free solutions that have been adapted to serve the work practices appropriately. That is, the team uses the tools that its developers use anyway in an efficient and effective manner that is accepted by the team thus representing an organically grown consensus on the way the virtual team operates.

5.3 Continuous Improvement

Tweaking the work practices, i.e., pragmatically adapting practices to the actual context (see also [22]), constitutes an important success factor of Debitoor's virtual work environment. The practices at Debitoor, which we exemplarily described, are constantly under inspection for continuous improvement opportunities. The team is encouraged to challenge the current modus operandi during each retrospective, and the team is also welcomed to discuss urgent and/or paining aspects during the **one-on-one** meetings. From the very beginning on, the CTO's mission has been to reduce waste [28] as much as possible, and to allow for the continuous evolution of and experimentation with the practices.

This constant strive for effectiveness captures the essence of the 12th principle of the Agile Manifesto [1]. Inspect and adapt is one of the core engines that through articulation-work [41] and meta-work [17] generates hybrid approaches [22]. Additional exploration of this topic can be found in [24].

6 CONCLUSION AND FUTURE WORK

Distributed agile software development is a stream that increasingly gains momentum. The combination is hard and complicated to implement, and organizations often tend to compromise agility in favor of additional control under the assumption that this is required to cope with the challenges created by the physical separation.

In this paper, we have extensively described and provided examples from a small Danish single-product company that managed to design and implement a virtual work environment in which tools and practices have been carefully picked and improved to support the distributed agile development team. From our one-year observation, we conclude that the smart adaptation and alignment of standard practices in combination with tools well-accepted by the development team supports the efficient and effective operation of distributed development projects. Neither of the team members would change the current project setup. Even though our findings

Table 8: Challenges related to the virtual work environment identified from [23], their impact, and their presence at Debitoor.

Challenge	Impact	Observed Solution at Debitoor
Lack of attendance	The lack of attendance at the daily stand-up creates a feeling of lost importance of the meeting [7].	Daily stand-up meetings at Debitoor are a peculiar practice that has been continuously modified until it reached the format described in Section 4.2.4. In particular, the participation to the daily stand-up (and, in general, all meetings) is not a barrier, as: (i) by conducting it via online platforms at their workstation and not in meeting rooms, the meeting does not require physical presence; and, (ii) by collaboratively agreeing on a different format in terms of content, it becomes more appealing to team members.
Meetings at only one site/ keep meetings jointly	Having meetings at only one site isolates the rest of the distributed team members and lowers the feeling of being united as a team [31].	Since all meetings in the development team are conducted online using computer-mediated communication, this challenge is not faced at Debitoor.
Lacking team cohesion	Lack of team cohesion is a common problem among distributed teams, in which the members do not necessarily perceive themselves as part of on-/off-site team. This might affect the shared views of goals [35].	The improved cooperation facilitated via Slack PM, calls, and channels together with the company culture promoted by the CTO (see, e.g., the use of the mirror) clearly mitigated this issue. Additionally, all team members have access to the same amount of documentation through GitHub, the Waffle.io board, and Google Products. Due to the way the virtual work environment has been designed, team cohesion has not been observed to be an issue. An insight that was confirmed by the team members through the questionnaire (see Q1).
Create transparency among sites	Lack of transparency among sites is likely to be one of the most subtle and dangerous threats in GSE, which is related to the idiosyncratic distances of the arrangement [6, 33]. It causes a series of negative effects, e.g., poor knowledge sharing and lack of team cohesion.	Even though the individual cultures and different mother tongues within the team, even though physical and temporal distances are a fact at Debitoor, the virtual work environment is designed to maximize transparency. The most visible elements of the environment that facilitate such transparency are the communication centralized around Slack open to all team members, the daily stand-up meetings, the Waffle.io board, and the mirror.
Balance between formal and informal communication and documentation	Agile development relies on informal face-to-face communication, but GSE traditionally requires more formal communication [7, 35].	Formal documentation at Debitoor takes place only in the Waffle.io board and GitHub. During stand-up and retrospective meetings, knowledge is captured and archived via Google products. However, most communication is informal and is supported by the Slack channels or Slack private messages. Far from claiming that Debitoor's approach to documentation is the silver bullet, we certainly observed that through these rather simple artifacts, the team managed to reach an efficient balance between formal and informal documentation.
Knowledge fragmentation	Generalizing the findings from [7] (i.e., synchronizing use cases and user stories when they are managed through different software tools), the issues related to knowledge being fragmented into several repositories can lead to problems like traceability.	One of the goals of Debitoor that impacted the design of the virtual work environment was to reduce waste by following the 10th agile principle [1], hence, achieve simplicity. The tool ecosystem at Debitoor comprises a limited number of software tools, which together with the found *balance between formal and informal documentation* (see Challenge above), allows members of the development team at Debitoor to not experience this problem.
Practices and tools adoption	The adoption of common practices and tools, among all sites, has been reported to be a success factor (e.g., [26]), which, if not addressed might generate friction and issues across sites.	This problem has never been experienced at Debitoor as all changes at Debitoor are a shared responsibility, which are discussed, decided, and implemented together during retrospective meetings.
Cost and impact of maintenance of team knowledge	Knowledge sharing needs to be prioritized to work efficiently, but establishing and maintaining knowledge sharing across sites is costly [27].	Knowledge sharing at Debitoor is continuously achieved via the established practices, and dedicated Slack channels have been created to allow communities of practices to thrive. Therefore, Debitoor is able to create and maintain knowledge across the different sites with little to no cost.

do not yet allow for generalization, however, we claim to have presented a prototype scenario of how to organize and run distributed projects efficiently. Key to the success of Debitoor is an appreciation of the development team's needs in terms of work practices and supporting tools. No out-of-the-box process was dogmatically implemented, and no huge standard tool that dictates certain procedures was used. We observed a general agreement on the work environment and the components it was made of. However, we have to acknowledge that our findings relate to a very specific context (see Table 1). Hence, we do not claim to have presented findings that can be extended to other contexts.

In this paper, we focused on the practices implemented at Debitoor and the tools used to support the practices. Specifically, we selected those data from our one-year observation that helped us understanding team-related issues, e.g., regarding communication and collaboration. Yet, the study revealed further insights, e.g., regarding effectiveness of specific practice-tool combinations or the efficiency of particular interaction patterns. An extended in-depth analysis of such aspects to develop an even better picture of the presented case and to help improving the conclusions draws, however, remains subject to further work.

REFERENCES

[1] Kent Beck, Mike Beedle, A Van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, Jon Kern, Brian Marick, Robert C. Martin, Steve Mellor, Ken Schwaber, Jeff Sutherland, and Dave Thomas. 2001. Manifesto for Agile Software Development. (2001). http://www.agilemanifesto.org

[2] Steve Berczuk. 2007. Back to Basics: The Role of Agile Principles in Success with an Distributed Scrum Team. In *Agile Conference (AGILE)*. IEEE, Washington, DC, USA, 382–388.

[3] Jacob T. Biehl, Mary Czerwinski, Greg Smith, and George G. Robertson. 2007. FASTDash: A Visual Dashboard for Fostering Awareness in Software Teams. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI)*. ACM, New York, NY, USA, 1313–1322.

[4] Bernd Bruegge, Andrea De Lucia, Fausto Fasano, and Genoveffa Tortora. 2006. Supporting Distributed Software Development with Fine-grained Artefact Management. In *Proceedings of the IEEE International Conference on Global Software Engineering (ICGSE)*. IEEE, Washington, DC, USA, 213–222.

[5] Fabio Calefato, Daniela Damian, and Filippo Lanubile. 2012. Computer-mediated communication to support distributed requirements elicitations and negotiations tasks. *Empirical Software Engineering* 17, 6 (2012), 640–674.

[6] Erran Carmel and Ritu Agarwal. 2001. Tactical approaches for alleviating distance in global software development. *IEEE Software* 18, 2 (2001), 22–29.

[7] M. Cristal, D. Wildt, and R. Prikladnicki. 2008. Usage of SCRUM Practices within a Global Company. In *2008 IEEE International Conference on Global Software Engineering*. 222–226.

[8] Richard L. Daft and Robert H. Lengel. 1986. Organizational Information Requirements, Media Richness and Structural Design. *Management Science* 32, 5 (1986), 554–571.

[9] Philipp Diebold, Jan-Peter Ostberg, Stefan Wagner, and Ulrich Zendler. 2015. What Do Practitioners Vary in Using Scrum? In *International Conference on Agile Software Development (XP)*. Lecture Notes in Business Information Processing, Vol. 212. Springer, Cham, 40–51.

[10] Kevin Dullemond, Ben van Gameren, and Rini van Solingen. 2014. Collaboration Spaces for Virtual Software Teams. *IEEE Software* 31, 6 (Nov 2014), 47–53.

[11] Steve Easterbrook, Janice Singer, Margaret-Anne Storey, and Daniela Damian. 2008. Selecting empirical methods for software engineering research. *Guide to advanced empirical software engineering* (2008), 285–311.

[12] Christof Ebert, Marco Kuhrmann, and Rafael Prikladnicki. 2016. Global Software Engineering: Evolution and Trends. In *Proc. of the IEEE International Conf. on Global Software Engineering (ICGSE)*. IEEE, Washington, DC, USA, 144–153.

[13] Clarence A. Ellis, Simon J. Gibbs, and Gail Rein. 1991. Groupware: Some Issues and Experiences. *Commun. ACM* 34, 1 (Jan. 1991), 39–58.

[14] Fabian Fagerholm, Alejandro Sanchez Guinea, Jay Borenstein, and Jürgen Münch. 2014. Onboarding in Open Source Projects. *IEEE Software* 31, 6 (Nov 2014), 54–61.

[15] Jon Froehlich and Paul Dourish. 2004. Unifying Artifacts and Activities in a Visual Tool for Distributed Software Development Teams. In *Proceedings of the International Conference on Software Engineering (ICSE)*. IEEE Computer Society, Washington, DC, USA, 387–396.

[16] Hugo Fuks, Alberto Raposo, Marco A Gerosa, et al. 2008. The 3c collaboration model. In *Encyclopedia of E-collaboration*. IGI Global, 637–644.

[17] Elihu M Gerson. 2008. Reach, bracket, and the limits of rationalized coordination: Some challenges for CSCW. In *Resources, Co-Evolution and Artifacts*. Springer, 193–220.

[18] Dale L. Goodhue and Ronald L. Thompson. 1995. Task-Technology Fit and Individual Performance. *MIS Quarterly* 19, 2 (1995), 213–236.

[19] Carl Gutwin, Saul Greenberg, and Mark Roseman. 1996. *Workspace Awareness in Real-Time Distributed Groupware: Framework, Widgets, and Evaluation*. Springer London, London.

[20] Lile Hattori and Michele Lanza. 2010. Syde: A Tool for Collaborative Software Development. In *Proceedings of the International Conference on Software Engineering (ICSE)*. ACM, New York, NY, USA, 235–238.

[21] James D. Herbsleb. 2007. Global Software Engineering: The Future of Socio-technical Coordination. In *Future of Software Engineering (FOSE)*. IEEE, Washington, DC, USA, 188–198.

[22] Marco Kuhrmann, Philipp Diebold, Jürgen Münch, Paolo Tell, Vahid Garousi, Michael Felderer, Kitija Trektere, Fergal McCaffery, Christian R. Prause, Eckhart Hanser, and Oliver Linssen. 2017. Hybrid Software and System Development in Practice: Waterfall, Scrum, and Beyond. In *Proceedings of the International Confernce on Software System Process (ICSSP)*. ACM, New York, NY, USA, 30–39.

[23] Pernille Lous, Marco Kuhrmann, and Paolo Tell. 2017. Is Scrum Fit for Global Software Engineering?. In *Proceedings of the IEEE International Conference on Global Software Engineering (ICGSE)*. IEEE, Washington, DC, USA, 1–10.

[24] Pernille Lous, Paolo Tell, Christian Bo Michelsen, Yvonne Dittrich, and Allan Ebdrup. 2018. From Scrum to Agile: a Journey to Tackle the Challenges of Distributed Development in an Agile Team. In *Proceedings of the ACM International Conference on Software and System Process (ICSSP)*. ACM (in press).

[25] M.B. Miles, A.M. Huberman, and J. Saldaña. 2013. Qualitative Data Analysis. SAGE Publications.

[26] S. Modi, P. Abbott, and S. Counsell. 2013. Negotiating Common Ground in Distributed Agile Development: A Case Study Perspective. In *Proceedings of the IEEE International Conference on Global Software Engineering (ICGSE)*. IEEE, Washington, DC, USA, 80–89.

[27] Nils Brede Moe, Tor Erlend Fægri, Daniela S. Cruzes, and Jan Edvard Faugstad. 2016. Enabling Knowledge Sharing in Agile Virtual Teams. In *Proceedings of the IEEE International Conference on Global Software Engineering (ICGSE)*. IEEE, Washington, DC, USA, 29–33.

[28] Shahid Mujtaba, Robert Feldt, and Kai Petersen. 2010. Waste and Lead Time Reduction in a Software Product Customization Process with Value Stream Maps. In *Australian Software Engineering Conference (ASWEC)*. IEEE, 139–148.

[29] Tuomas Niinimaki. 2011. Face-to-face, email and instant messaging in distributed agile software development project. In *Proc. of the IEEE International Conf. on Global Software Engineering (ICGSE)*. IEEE, Washington, DC, USA, 78–84.

[30] Tuomas Niinimaki, Arttu Piri, Casper Lassenius, and Maria Paasivaara. 2010. Reflectingof communication tools in GSD projects with media synchronicity theory. In *Proc. of the IEEE International Conf. on Global Software Engineering (ICGSE-Workshops)*. IEEE, Washington, DC, USA, 3–12.

[31] Maria Paasivaara. 2011. Coaching Global Software Development Projects. In *Proceedings of the IEEE International Conference on Global Software Engineering (ICGSE)*. IEEE, Washington, DC, USA, 84–93.

[32] Maria Paasivaara, Benjamin Behm, Casper Lassenius, and Minna Hallikainen. 2018. Large-scale agile transformation at Ericsson: a case study. *Empirical Software Engineering* (11 Jan 2018). https://doi.org/10.1007/s10664-017-9555-8

[33] Maria Paasivaara, Casper Lassenius, Ville T. Heikkilä, Kim Dikert, and Chistian Engblom. 2013. Integrating Global Sites into the Lean and Agile Transformation at Ericsson. In *Proceedings of the IEEE International Conference on Global Software Engineering (ICGSE)*. IEEE, Washington, DC, USA, 134–143.

[34] Javier Portillo-Rodríguez, Aurora Vizcaíno, Mario Piattini, and Sarah Beecham. 2012. Tools Used in Global Software Engineering: A Systematic Mapping Review. *Inf. Softw. Technol.* 54, 7 (July 2012), 663–685.

[35] Balasubramaniam Ramesh, Lan Cao, Kannan Mohan, and Peng Xu. 2006. Can Distributed Software Development Be Agile? *Commun. ACM* 49, 10 (Oct. 2006), 41–46.

[36] Per Runeson, Martin Host, Austen Rainer, and Bjorn Regnell. 2012. *Case study research in software engineering: Guidelines and examples*. John Wiley & Sons.

[37] K. Schmidt. 1994. Cooperative Work and its Articulation: Requirements for Computer Support. *Le Travail Humain* 57, 4 (1994), 345–366.

[38] Helen Sharp, Rosalba Giuffrida, and Grigori Melnik. 2012. Information flow within a dispersed agile team: a distributed cognition perspective. In *International Conference on Agile Software Development*. Springer, 62–76.

[39] Darja Šmite, Marco Kuhrmann, and Patrick Keil. 2014. Virtual Teams [Guest editors' introduction]. *Software, IEEE* 31, 6 (Nov. 2014), 41–46.

[40] Igor Steinmacher, Ana Paula Chaves, and Marco Aurélio Gerosa. 2013. Awareness Support in Distributed Software Development: A Systematic Review and Mapping of the Literature. *Computer Supported Cooperative Work (CSCW)* (2013), 113–158.

[41] Anselm Strauss. 1985. Work and the division of labor. *The sociological quarterly* 26, 1 (1985), 1–19.

[42] Antônio R. D. R. Techio, Rafael Prikladnicki, and Sabrina Marczak. 2015. Reporting Empirical Evidence in Distributed Software Development: An Extended Taxonomy. In *Proceedings of the IEEE International Conference on Global Software Engineering (ICGSE)*. IEEE, Washington, DC, USA, 71–80.

[43] Darja Šmite, Claes Wohlin, Tony Gorschek, and Robert Feldt. 2010. Empirical Evidence in Global Software Engineering: A Systematic Review. *Empirical Softw. Engg.* 15, 1 (Feb. 2010), 91–118.

[44] Dietmar Winkler, Stefan Biffl, and Andreas Kaltenbach. 2010. Evaluating Tools That Support Pair Programming in a Distributed Engineering Environment. In *Proc. of the International Conf. on Evaluation and Assessment in Software Engineering (EASE)*. BCS Learning & Development Ltd., Swindon, UK, 54–63.

[45] Robert K Yin. 2013. *Case study research: Design and methods*. Sage publications.