IT UNIVERSITY OF COPENHAGEN

DOCTORAL THESIS

Collaborative Windows – A User Interface Concept for Distributed Collaboration

Author:
Morten ESBENSEN

Supervisor:
Dr. Jakob E. BARDRAM

A thesis submitted in partial fulfilment of the requirements for the degree of Doctor of Philosophy

at the

IT University of Copenhagen

August 2015

PITLAB
PERVASIVE INTERACTION TECHNOLOGY LABORATORY

Abstract

Distributed collaboration is the work arrangement in which people distributed across different locations collaborate on achieving a common goal. One particular domain of work that has embraced distributed collaboration is software development. Global software development is the special kind of software development where the production of software is carried out by geographically dispersed people. Such work however, is challenged by the distance between people and a strategy for handling the complex dependencies that exists in distributed software development is to engage in closely coupled work where close collaboration and frequent meetings drive the work. One way to achieve this way of working is to implement the Scrum software development framework. Implementing Scrum in globalized context however, requires transforming the Scrum development methods to a distributed setup and extensive use of collaboration technologies.

In this dissertation, I explore how novel collaboration technologies can support closely coupled distributed work such as that in distributed Scrum. This research is based on three different studies: an ethnographic field study of distributed Scrum between Danish and Indian software development companies and the design, implementation and evaluation of the two video-communication based collaboration tools SideBar and dBoard.

Based on these studies I present the concept of *collaborative windows*—a user interface concept for a special kind of collaborative video-mediated systems. A collaborative window is a video-based collaboration tool that implements three properties; *(i)* it employs a window metaphor to its videoconferencing features, *(ii)* it uses a content-on-video approach to superimpose interactive content on top of the video and *(iii)* it implements context-awareness to adjust its behaviour to the surrounding environment. The dissertation explains the concept and presents a design space analysis for collaborative window systems.

Acknowledgements

Four years ago I started as a Ph.D. student at the IT University of Copenhagen. Since then, I have encountered numerous people who have helped, influenced or inspired my work for which I am thankful.

First, I extend my greatest thanks to my supervisor Jakob E. Bardram. You have been a great inspiration and support in my work ever since I entered the PitLab as an undergraduate student and all the way through my Ph.D. studies. It has been a pleasure to work with you and the results presented in this dissertation have been produced as a result of your excellent guidance.

Second, I would like to thank all my former and current colleagues in the PitLab. Special thanks go to Aurelien Tabard, Juan D. H. Ramos, Steven Houben, Steven Jeuris, Paolo Tell, Sebastian Büttrich, Jakob B. Cholewa and Mathias K. Pedersen for all your help with my research and for making the PitLab a great environment in which I have enjoyed working the last six years. I also extend my thanks to Carl Gutwin and the students at the Interaction Lab at the University of Saskatchewan for having me stay there for four months and showing me the hospitality of Canada.

As a member of the NexGSD research project I would also like to thank Rasmus E. Jensen, Stina Matthiesen, Thomas Tøth and Anne-Marie Søderberg for many fruitful discussions and input on my work throughout the four years. A special thanks goes to Pernille Bjørn for help and guidance of my project and for showing me new ways of research. I have truly enjoyed working with you.

Lastly, a special thanks go to my beloved family for supporting me through my years as a Ph.D. student. A very special thanks goes to my wonderful girlfriend Line Jakobsen. Without your love, support and endless compassion throughout these last years, I would not have made it this far. I hope I may someday repay you.

Papers Included

The present dissertation consist of an introduction and a collection of the following four research papers:

[P1] **Morten Esbensen** and Pernille Bjørn: Routine and Standardization in Global Software Development. In *Proceedings of the 18th International Conference on Supporting Group Work*, **(GROUP '14)**, ACM, 2014

[P2] **Morten Esbensen**, Paolo Tell and Jakob E Bardram: SideBar: Videoconferencing System Supporting Social Engagement. In *2014 International Conference on Collaborative Computing: Networking, Applications and Worksharing*, **(CollaborateCom '14)**, IEEE, 2014

[P3] **Morten Esbensen**, Paolo Tell, Jacob B. Cholewa, Mathias K. Pedersen, Jakob E. Bardram: The dBoard: a Digital Scrum Board for Distributed Software Development. Conditionally Accepted for publication in *Proceedings of the 10th International Conference on Interactive Tabletops and Surfaces*, **(ITS '15)**, ACM, 2015

[P4] **Morten Esbensen**, Paolo Tell, Jacob B. Cholewa, Mathias K. Pedersen, Jakob E. Bardram: Facilitating Distributed Standup Meetings through Blended Video Conferencing and Task Management Technology. *Submitted, Revised and Resubmitted, Rejected at CSCW, 2016*

Full Publication List

Pre-print Manuscripts

- **Morten Esbensen**, Paolo Tell, Jacob B. Cholewa, Mathias K. Pedersen, Jakob E. Bardram: The dBoard: a Digital Scrum Board for Distributed Software Development. Conditionally Accepted for publication in *Proceedings of the 10th International Conference on Interactive Tabletops and Surfaces*, **(ITS '15)**, ACM, 2015
- **Morten Esbensen**, Paolo Tell, Jacob B. Cholewa, Mathias K. Pedersen, Jakob E. Bardram: Facilitating Distributed Standup Meetings through Blended Video Conferencing and Task Management Technology. *Submitted, Revised and Resubmitted, Rejected at CSCW, 2016*

Full Papers

- Pernille Bjørn, **Morten Esbensen**, Rasmus Eskild Jensen and Stina Matthiesen: Does Distance Still Matter?; Revisiting the CSCW Fundamentals on Distributed Collaboration. *ACM Transactions on Computer-Human Interaction*, **(TOCHI)**, 21.5 (2014): 27
- **Morten Esbensen** and Pernille Bjørn: Routine and Standardization in Global Software Development. In *Proceedings of the 18th International Conference on Supporting Group Work*, **(GROUP '14)**, ACM, 2014
- **Morten Esbensen**, Paolo Tell and Jakob E Bardram: SideBar: Videoconferencing System Supporting Social Engagement. In *2014 International Conference on Collaborative Computing: Networking, Applications and Worksharing*, **(CollaborateCom '14)**, IEEE, 2014
- Steven Houben, Søren Nielsen, **Morten Esbensen** and Jakob E. Bardram: Noosphere: An activity-centric infrastructure for distributed interaction. In *Proceedings of the 12th International Conference on Mobile and Ubiquitous Multimedia*,**(MUM '13)**, ACM, 2013

Workshop Papers

- Jakob E. Bardram, **Morten Esbensen**, and Paolo Tell: Supporting Co-located SCRUM Processes in Global Software Development. In workshop of *"Local Remote" Collaboration: Applying Remote Group Awareness Techniques to Co-located Settings*, **(CSCW '15)**

- **Morten Esbensen** and Jakob Bardram: Tool Support for Globally Distributed Scrum. In workshop of *Global Software Development in a CSCW perspective*, **(CSCW '14)**
- Steven Houben, **Morten Esbensen** and Jakob E. Bardram: A Situated Model and Architecture for Distributed Activity-Based Computing. In workshop of *Modiquitous '12, the 2nd international workshop on model-based interactive ubiquitous systems*, **(EICS'12)**

Contents

List of Figures

Chapter 1

Introduction

The present dissertation presents work on designing technologies for distributed collaboration and the concept of collaborative windows—a novel type of collaboration technology for closely coupled distributed work. In this chapter, I present the background and domain of the work, the research question and provide an overview of the dissertation.

1.1 Background

Nowadays, more and more work is carried out as distributed work. In distributed collaboration, geographically dispersed actors work together on a common task. In order to enable distributed collaboration, a number of technologies should be used to handle communication. In these technological setups, video technologies play a large role. As a high-bandwidth communication channel [50] video is well suited for many collaborative tasks in which distributed co-workers have to come together. Video-conferencing provides advantages over for example audio-only setups in terms of meeting structure, turn taking and conveying body-language [38, 52].

Video-communication is not without problems though, and one area of problems with todays videoconferencing, relates to the fact that the videoconferencing technologies often are disconnected from other collaborative tools. Previous research have shown that videoconferencing is most often conducted in combination with other activities [11]. Just as in collocated meetings, distributed meetings using videoconferencing usually revolves around a common task. And as with most tasks that need to be solved collaboratively, such tasks are solved with the aid of tools. The problem here is related to the fact that the tools needed to solve a given task at hand, and the videoconferencing tools used to mediate the communication are not integrated which requires distributed actors to manually set up and relate these tools to each other. Creating such arrangements where different kinds of tools has to be connected and setup across locations poses challenges for distributed teams [4, 14] and there is a need to create technologies that integrate video with the other essential tools used in distributed collaboration [69].

Another area of problems of videoconferencing setups is that they are mostly designed for planned meetings. A typical videoconferencing setup in a company will be located in a dedicated meeting room where meetings can be planned and executed at specific times. In collocated settings however, much interaction and information sharing amongst colleagues happen outside of such planned meetings. From the informal meetings in hallways or at the coffee machine to the awareness gathered by sitting and working collocated, much information is gained simply by being collocated. In distributed collaboration, this information is lost which in turn causes challenges for the collaboration. These issues have been addressed with the concept of 'video windows' [21]' and 'media spaces' [6]. Both concepts describe how video can be use as always-on technologies to provide a view into a distant location instead of only being turned on during planned meetings.

1.2 Domain

This research has been conducted in the domain of global software development (GSD). GSD is one work domain that has adopted distributed collaboration as a standard way of working [15]. Companies are increasingly turning to GSD in search of greater profits through access to lower wage labor, in search of new talented employees or to reach new markets [15, 29]. Transitioning to global work however, also introduces a range of problems related to the distances of time, space and culture which pose challenges to communication, coordination and control [13].

Within GSD, agile methodologies like Scrum are now increasingly becoming applied [34]. The usage of Scrum in GSD might seem like a contradiction at first [59]; while Scrum advocates close collaboration, collocation and frequent communication, GSD projects are characterized by the distribution of people over two or more locations. With a focus on close collaboration and frequent meetings however, Scrum has proved a good approach to handle the complexities of GSD [57]. Adhering to the recommended meetings and transforming them to a globalized work context, companies have been observed to overcome some of the problems traditionally associated with GSD [2]. In general, achieving a successful collaboration across distances requires companies to implement appropriate work processes—such as those recommended by Scrum transformed to a distributed environment [35]—and communication and coordination technologies to facilitate these processes. This dissertation presents work on the latter of these two; how novel collaborative technologies can support distributed agile software development.

1.2.1 Next Generation Technologies for Global Software Development

The work presented in this dissertation has been carried out as a part of the research project 'Next Generation Technologies for Global Software Development' (NexGSD)[1]. The NexGSD project is a joint research project between universities and companies that have set out to develop processes and tools supporting GSD. The project has undertaken a multidisciplinary approach to GSD looking at challenges and opportunities from ethnographical, organizational and technological perspectives. This, so far has shed light over GSD from different perspectives looking into the challenges of GSD in terms of e.g. managing trust [70] or maintaining legacy systems [48].

A core tenet within the NexGSD project is to explore GSD as a potential for companies to stretch across countries and engage in close collaboration with distant colleagues rather than a platform for outsourcing work. We refer to outsourcing as the special kind of organizational arrangement where (parts of) the construction of software is shipped off to another location to be returned once it has been completed. Differently, we refer to close collaboration in GSD if the production of software relies on close collaboration between two or more sites that each take part in and responsibility of the construction and completion of the software as a whole. Our studies have showed that successful collaborative software projects can be achieved by keeping collaboration close despite the complexities of managing such closely coupled distributed work [5, 39]. This dissertation presents work on collaborative technologies for agile GSD and follows the tenet of the NexGSD project. GSD carries a potential for engaging in close collaboration between distributed teams and the tools we design to support such work should enable close collaboration to unfold rather than support outsourcing.

1.3 Research Question and Approach

This dissertation addresses how distributed collaboration can be supported from a technological perspective and asks the question: *How do we design technologies that support closely coupled collaborative work such as that of global Scrum?*. Following the triangulation process of Mackay and Fayard [47] the dissertation approaches this questions from three perspectives: a technological, an empirical and a conceptual.

[1] `http://nexgsd.org/`

Technological. From a technological perspective, the dissertation demonstrates two tools designed to support distributed teams; SideBar and dBoard. SideBar is an initial exploration into the domain of collaborative video and is a tool designed to support social engagement in video meetings. dBoard is a collaboration tool for GSD in the form of distributed Scrum board designed to support awareness and ad-hoc and planned meetings for a distributed Scrum team. I present the design and implementation of these prototypes which were used to explore the technical aspects of the research question.

Empirical. The empirical perspective encompasses a workplace study of a distributed Scrum team working out of Denmark and India and three evaluations of SideBar and dBoard. SideBar was evaluated in a small scenario-based usability study. dBoard was evaluated in two steps; in a scenario-based usability study with Scrum practitioners and in a field deployment between two distributed collaborating software development companies. These studies show how users found the tools both easy to use, usable and how combined video and domain specific applications was appreciated. The field deployment of dBoard also showed how this kind of technology was well suited for supporting the standup meetings of the distributed Scrum team but also pointed out challenges that were introduced with a deployment of the board.

Conceptual. The conceptual perspective presents the concept of *collaboration windows*—a user interface concept for collaborative systems that are specifically designed closely coupled distributed collaboration. A collaborative window application is a video-mediated communication tool where domain-specific interactive content is overlaid on top of the video. The concept is derived from the the workplace study and the experiences of designing, implementing and evaluating SideBar and dBoard and builds on previous research on content overlay in videoconferencing [43, 51], video windows [21] and media spaces [6].

1.4 Reading Guide

This dissertation is organized into two parts. Part I is an introduction and Part II is a collection of four research papers. In Part I, I introduce and discuss the background of the research, the methods used and the concept of collaborative windows that was developed based on the research. This part is structured as follows:

Chapter 2: *Background*
This chapter sets the stage of the dissertation by examining the domain of agile global software development, how video-mediated communication is used within the domain and what challenges are faced. The chapter also explains related work which is later

used to position the concept of collaborative windows.

Chapter 3: *Research Methods*
Chapter 3 presents the methods that were used in exploring the research questions. The chapter describes the ethnographic workplace study of distributed Scrum and the design, implementation and evaluation of two technology prototypes that were developed to support distributed software development: SideBar and dBoard.

Chapter 4: *Collaborative Windows*
This chapter presents the concept of collaborative windows—a user interface concept for collaborative systems that employ a window metaphor, use content-on-video to integrate video with other tasks and implement context awareness. The concept was derived from the workplace study of distributed Scrum and the work of designing, implementing and evaluating SideBar and dBoard and captures what was learned into a unified concept. The chapter places the concept in relation to related works and presents a design space analysis of systems that build on top of this concept. The purpose of this chapter is to summarize and crystallize what has been learned from working with developing systems for globally distributed scrum.

Chapter 5: *Discussion*
This chapter discusses the opportunities and challenges of collaborative windows and how the concepts might be appropriated to domains outside distributed agile software development.

Chapter 6: *Conclusion*
Chapter 6 concludes the dissertation by providing an answer to the research question and pointing out directions of future work.

Part II is a collection of four research papers:

Paper P1: *Routine and Standardization in Global Software Development*
This papers reports on an ethnographic work place study of globally distributed Scrum between a Danish software development company and an Indian service provider. The purpose of the study was to investigate the complex net of dependencies that exists in such a project and how a globally distributed team managed to successfully handle these dependencies. This paper contributes to the dissertation in two ways. *First*, it demonstrates how GSD using Scrum organized around close collaboration and frequent communication can be used to handle the complex dependencies in such a collaboration. *Second*, it was used as the first step in the design of the dBoard as some of the challenges

we observed were related to the lack of a Scrum board for the observed distributed Scrum team.

Paper P2: *SideBar: Videoconferencing System Supporting Social Engagement*
This paper presents the design, implementation and evaluation of SideBar. SideBar was our first exploration into the domain of collaborative video. The purpose of SideBar is to give distributed co-workers opportunities to connect to each other and engage in more social activities. SideBar is a videoconferencing system that extends a traditional videoconferencing systems with tablet computers for all participants. The tablets show a mirrored videofeed from the conference camera, however, the video has been made interactive by superimposing interactive rectangles on top of faces of people in the video. Using this interactive video, users can navigate to profiles of each other and engage in one-to-one chat conversations during the meetings.

Paper P3: *dBoard: a Digital Scrum Board for Distributed Software Development*
In this paper we present the dBoard—an interactive Scrum board and video-communication tool designed for distributed Scrum teams. The dBoard superimposes a shared interactive Scrum board on top of a video-stream. Designed for both planned as well as unplanned meetings, the dBoard is able to blur and de-blur the video automatically the proximity of people and the whole system is integrated with existing task tracking system so it fits into the standart collaboration setup used in distributed software development. The paper reports on the design, implementation and user evaluation of the dBoard.

Paper P4: *Facilitating Distributed Standup Meetings through Blended Video Conferencing and Task Management Technology*
This last paper reports on a field deployment of the dBoard. We first studied the daily Scrum meeting practices of a small Danish software development company to understand and learn how the meetings supported their Scrum process and what challenges the team faced. We then deployed the dBoard in two companies; the Danish company and their collaborating partner in Flensburg, Germany for a period of 5 weeks. During this time we observed the standup meetings of the teams as they used the boards. The paper describes the outcomes of these studies that showed how the dBoard was able to provide support for their daily standup meetings while also pointing out challenges related to the physical and organizational embeddedness of the board.

Part I

Collaborative Windows

Chapter 2

Background

This chapter describes the background and context of the dissertation. In particular, since the work presented here has been carried out within the domain of distributed agile software development, the the first part of chapter explains the concept of global software development and the relation to Scrum. The second part of the chapter then explains the role of videoconferencing—a widely used communication tools for GSD—and the benefits and challenges of using videoconferencing in distributed collaboration.

2.1 Global Software Development

Global software development (GSD) is the special kind of software development where work is split across two or more geographically dispersed sites [44]. In recent years, GSD has become common practice [15] as companies have realized that GSD carries potential for savings through reduced labour costs, easier penetrations of new markets and access to larger pools of skilled labour [15]. GSD however, also faces challenges related to the geographical, temporal, cultural and linguistic distances [53] that impact communication, coordination and control [13]. In worst case, such challenges can cause a distributed collaboration to fail [7] however, if handled correctly, succesful collaboration can unfold [20] and companies might reap the benefits of GSD.

This dissertation refers to GSD as the work arrangement within software development where several people in two or more geographically dispersed locations work together to produce software. Here, it is important to note that several terms are used in the literature when talking about GSD. Distributed software development, global software development or global software engineering are all used somewhat interchangeably by different authors when describing the same domain. In this dissertation I use the term global software development and the abbreviation GSD to describe such work arrangements however, it should be noted that other terms exist and that these have been treated as being similar description for the same concept.

2.1.1 Scrum in GSD

Scrum is a software development framework that suggests how to arrange work, development teams and meetings [63]. Scrum is built on the Agile philosophy in which the it is recognized that software requirements are hard to define up-front and that the development teams should embrace change [3]. In Scrum, software requirements are formulated as user stories which are broken down into tasks and arranged in the product backlog. Development teams are organized into Scrum teams—small self-managing teams consisting of a Scrum master, developers and testers. The work is arranged into sprints—time boxed unit of typically 2-3 weeks—during which a number of user stories are implemented. The effort it takes to implement a user story is estimated by the Scrum teams and every sprint, a number of user stories corresponding to the teams velocity (i.e. the amount of estimated user stories the team can produce) are selected and placed in the sprint backlog. During the sprint, the developers in the Scrum teams assign user stories to each other and remove them from the backlog as they are completed. At the end of a sprint, the implemented stories are accepted or rejected in a demonstration meeting and the Scrum team reflects on the work processes in the retrospective meeting.

While Scrum has been designed for collocate use, it is increasingly being applied to GSD as well with succeses (e.g. [2, 57, 67]). Despite the fact that previous research suggests that the challenges of GSD should be handled through reducing collaboration [13] it is exactly the focus on frequent meetings and close collaboration of Scrum that has proven to be successful in handling the problems that arise from the distances in GSD [57]. Still however, challenges exist related to the introduction of Scrum in GSD which requires companies using Scrum in distributed environments to tailor the Scrum practices to fit with the distances [35, 56], make use of other strategies such as aligning working hours and additional informal meetings as well as make extensive use of different communication tools [34].

2.2 Video Mediated Communication

One central research theme in GSD is how technologies can support GSD [29]. To achieve successful collaboration, a wide range of tools should be used. Noll et. al.

describe how one of the primary solutions to overcome the barriers created by the distances in GSD is to make use of synchronous communication technologies [53]. Within synchronous communication, videoconferencing plays a large role as the highest communication bandwidth technology [17]. Previous research on technology support for GSD suggests that video carries value in the fact that that meetings are better structured, and that people tend to be more focussed compared to audio only [52]. Isaacs and Tang's studies of video technologies suggest, that compared to audio-only communication the benefits of video is how video allows expressing understanding, forecasting responses, managing pauses and including non-verbal communication [38]. Video-communication however, also suffers from problems. One line of problems relate to the fact that traditional videoconferencing setups where the camera is mounted on top or below the screen create a parallax that disrupts eye-contacts and pointing gestures. To solve such issues, previous research has looked into creating setups with one way transparent screens where the camera can be placed behind the screen [42, 55, 68], or recently, using 3d-sensors in additional to cameras to create 3d photorealistic representations of meeting participants that can be rendered 'correctly' on screen to correctly transmit both gaze and gestures [31, 74].

2.2.1 Video Windows and Media Spaces

One issue in distributed collaborative work is tied to all the information that which is lost when people are working geographically dispersed. From the spontaneous encounters in hallways to the knowledge of where ones colleagues are located, much information is shared among people simply by being collocated. Staying aware of remote co-workers' schedules, work and personal lives is much more difficult in distributed collaboration [12] and existing technology setups are not designed to support such awareness [37]. This problem is perhaps not directly linked to videoconferencing as a technology. Indeed, the name video-*conferencing* or video-*meeting* implies that this kind of technology is targeted planned events such as a conference and not designed for informal communication and awareness. However, this is a general problem of distributed work and it is included in this dissertation as it links to the usage of videoconferencing. In distributed work, the video serves as the closest means of achieving collocation, however, it does not support an important aspect of what it entails to be collocated.

To address these issues, the concepts of video windows and media spaces have been proposed. A video window is an always on video connection between two sites intended for people to be able to communicate with each other through a virtual window in the wall providing a video into a distant location [21]. Similarly, the concept of mediaspaces describe video and audio connected geographically dispersed sites that support awareness, chance encounters and group discussions [6].

2.2.2 Integration with other tools

Another problem of todays videoconferencing equipment is the fact that it is rarely integrated with other tools. A traditional setup consists of camera, screen, microphone and speakers—perhaps located in a dedicated meeting room. While some setups allow for bringing in documents or provide an additional screen for screen sharing, there is no explicit support for collaborative tools to be used during the meetings. While videomeetings indeed are about *meetings*—which the technology facilitates—is usually done in conjunction with other tasks [11]. When distributed actors meet using videoconferencing, it is with a task at hand. The point here is, that these tasks usually also require some tools to accomplish and that these tools are not integrated with the videoconferencing equipment. There is a need for videoconferencing technologies that integrate with the other tools used during meetings [69].

The problems of establishing a video channel and meanwhile configuring other tools is part of Bjørn and Christensen's concept of *relation work* [4, 14]. Relation work denotes all the work that goes into establishing the social-technical relations between people and people, people and artifacts and artifacts and artifacts. Establishing these connections, the authors argue, become increasingly complex in distributed collaboration. The problems of setting up videoconferencing equipment in conjunction with other tools is one instance of the complexities of relation work in distributed collaboration. Bjørn and Christensen observed how communicating information on physical task wall in global engineering through video became a cumbersome task. Indeed, the disconnection between the task wall and the video set the need for highly complex and complicated relation work to take place.

Some previous research however, have focussed on creating vide-mediated communication setups which provide support for collaboration around shared artifacts. The notion of blended interaction spaces and the accompanying BISi system for example, is a video-mediated communication system inspired by the HP Halo systems that have been engineered in software as well as hardware to create a video-mediated space in which external artifacts and data sourced can be brought into the meeting [54].

2.2.3 Beyond Being There

Despite the aforementioned problems within the context of video-mediated communication, previous research have also pointed out that video-communication provides possibilities of creating systems that provide collaborating actors with interaction possibilities that are not possible in collocated environments. System that—in theory—could be preferred even if interaction in physical proximity was possible [32]. *Beyond being there* is the concept that describe such systems that take advantage of the possibilities of modern technology to provide users with experiences and interactions beyond physical collocation. Roussel and Gueddana elaborate on this concept and propose the notion of Multiscale Communication Systems—a telecommunication concept that aims to support variable degree of engagement, smooth transition between such degrees and integration with other media [61]. Concepts such as beyond being there and multiscale communication systems suggest that we can design and implement communication-tools that provide us with functionality that we do not have in collocated settings.

2.3 Summary

In distributed collaboration, distributed actors work together on a common task. One instance of distributed collaboration is GSD where the work of producing software is carried out by people located in geographically dispersed locations. With opportunities for advantages of reduced costs and access to skilled labour, GSD has become popular but GSD is also challenged by geographical, temporal, cultural and linguistic distances. Within GSD, agile methodologies such as Scrum are now increasingly becoming applied as they have proven to be effective in helping overcome the challenges associated with the distances of GSD. In order to successfully implement Scrum in GSD, the processes need to be modified to the distances and appropriate technologies should be used for handling the collaboration. One such technology that has enabled this way of working is videoconferencing. Videoconferencing enables distributed coworkers to met in a high communication bandwidth channel, but videoconferencing still does not compare to collocation and suffer from several drawbacks. In particular, the lack of support for informal awareness and the disconnections between video-technology and other tools needed during video-meetings pose challenges for distributed collaboration. In the following chapters, I present work on collaboration technologies that seek to solve these issues.

Chapter 3

Research Methods

The purpose of this dissertation is to gain knowledge about designing collaborative tools for closely coupled distributed work such as that of distributed Scrum. In doing so, three main studies were conducted: an ethnographic work place study of globally distributed Scrum and the development of two video-based communication tools: SideBar and dBoard. All studies are detailed in Part II (the study in [P1], SideBar in [P2] and dBoard in [P3] and [P4]) so this chapter explains the studies with a focus on how they helped inform and conceptualize the notion of collaborative windows (Chapter 4).

This chapter describes the research methods that have been applied in the investigation of the research question and the three aforementioned studies. The specific activities in the research methodology can be depicted using Mackays and Fayard's framework of scientific triangulation which describes the research methods underlying HCI research and how these methods relate to each other [47]. More specifically, HCI research triangulates between theory, the design of artifacts and observations. The methods used in exploring collaboration tools for globally distributed work presented in this dissertation can be seen in Figure 3.1.

The observation perspective encompassed a workplace study [46] of distributed Scrum which was conducted to explore globally distributed Scrum and provide input for the design of technologies supporting this kind of work arrangement. The design of artifacts perspective involved the design and implementation of two collaboration tools: SideBar and dBoard. The design of these tools both followed a centered design process [1] where wireframes or working prototypes were presented to users used in order to elicitate final requirements [25]. Both systems were built as fully functional prototypes. The tools were then—from an observational perspective—evaluated in user studies and the dBoard was furthermore subjected to a field deployment [66]. Finally, the concept of collaborative windows was conceived based on the work of the three main studies.

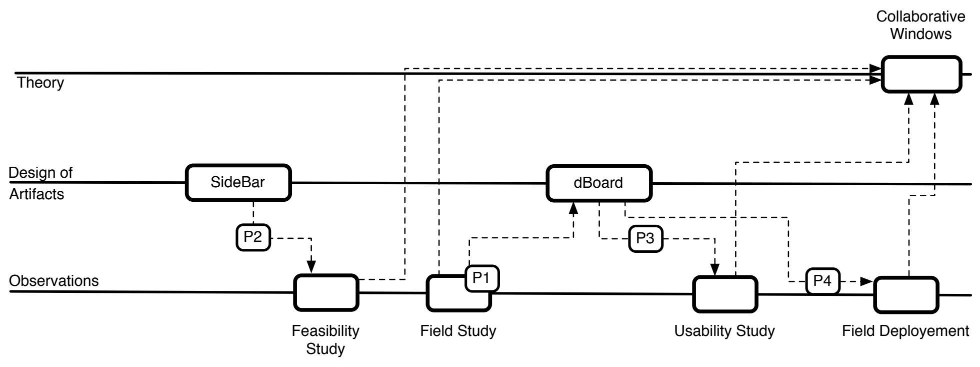


FIGURE 3.1: The research methods positioned in Mackay and Fayard's framework of triangulation [47]. The papers presented in Part II are are depicted in the figure as well (P1 - P4).

3.1 Workplace Study

In 2013 we performed an ethnographic workplace study of a globally distributed Scrum team with members located in both Denmark and India. Using observations and interviews, two researchers studied the work practices of the team for two weeks. With one researcher located in Denmark and one in India we had access to a global view of the collaboration [58]. The purpose of this study was two-fold. First, we wanted to understand how a distributed Scrum team managed to handle all the dependencies in work that arise when working in a distributed team. Second, we wanted to gather input for the design of collaborative systems supporting global Scrum. Paper [P1] explains how the team used the strategies of routine and standardization to manage the complex dependencies that arise in distributed work, so this section explains what we learned from the study that helped us design tools for distributed Scrum. We also refer to Paper [P1] for details on the study methodology. It is important to note here, that this study was conducted after SideBar (Section 3.2) was designed and evaluated, and the results thus were used for the design of dBoard (Section 3.3) and as part of the motivation for the concept of collaborative windows (Chapter 4).

3.1.1 Results

Our study of the distributed Scrum team revealed that they managed to keep an ongoing close collaboration despite the large distance between Denmark and India and that Scrum was a particularly good match for the complex work the goes into developing software across countries. Adhering to many of the suggested Scrum practices—such as daily meetings, sprint planning and estimation, and team struture—the team successfully carried out the Sprint we observed and in general expressed that the collaboration was a success. Despite the success however, we also observed problems which were mostly related to the technical setup the teams used to facilitate the collaboration.

In the case of a daily standup meeting in Scrum, for example, the lack of a dedicated space where the meetings could take place caused delays. While the Indian developers always used the same meeting room for the daily meetings, in Denmark, a room was booked on a day-to-day basis. As the Danish developers did not have a spatially stable location to go to when conducting the meetings, we observed how delays occurred. In several instances, the Danish developers were late to the meetings as they first had to figure out which meeting room had been booked for the meeting and then navigate through the (rather large) office building to that room. This could easily delay the meeting by 5 minutes, leaving little time for the meeting as only 15 minutes had been allocated in India before another team had booked the room. Without access to a common, stable location to conduct daily meetings, the meetings were delayed and had to be rushed so they could finish within the 15 minute timeslot.

The lack of integration between video and other tools also caused problems during the daily meetings. We observed how the meetings between the two distributed sites were done using videoconferencing but also required the access to their task tracking software. Indeed, these two technologies were not integrated but had to be set up independently for each meeting which had two negative consequences for these meetings. First, the setup costs of these meetings were quite high which contrasts the intention of the daily meeting. These meetings are supposed to be short and effective but we observed how this was only possible in a distributed setting if setting up the equipment was as effortless as possible. Both technologies required significant knowledge to set up—knowledge which on the Indian side only resided with one developer. If this developer was not available to set up the technology we observed how the others struggled to establish the video connection and log in to the task tracking system which delayed the meeting. Second, the lack of integration between video and task mangement tool caused a situation where only the Danish Scrum master could update the state of tasks and the control of the system thus resided with him.

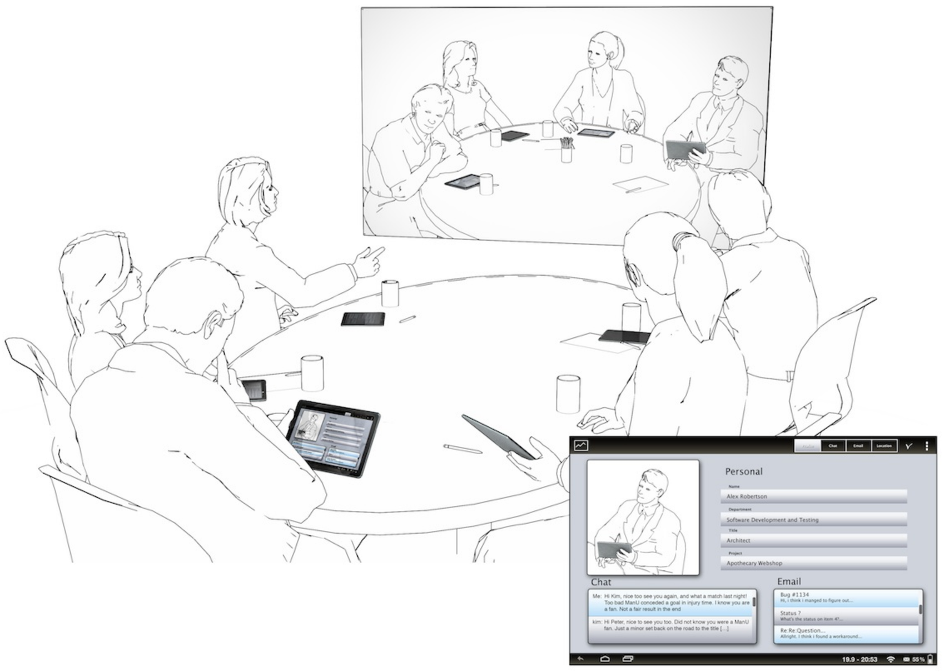
The same problems were observed during the demonstration meeting which was conducted at the end of the Sprint, only here, the consequences of not having an integrated video and demonstration tool were more severe. During the demonstration meetings, the Scrum team demonstrated the user stories that had been implemented in the sprint for the Product Owner. The Product Owner would either accepts the user story as completed or reject it in which case the user story was placed back into the development backlog for further development. What we observed was, that due to the technical limitations of the technologies used in the demonstration meetings, it was not possible for the Indian developers to participate in the meeting, and instead, Danish developers demonstrated the features that the Indian developers had worked on. The reason for this was, that the team needed a large projector to demonstrate the features which did not allow them to simultaneously have a video link to the Indian offices. At the same time, was the test-server setup located in Denmark and the Indian developers did not have access to it. The technological contraints thus meant, that the Indian developers had to inform developers in Denmark on what was to be demonstrated and how, and all demonstration was done in Denmark by the Danish team.

3.1.2 Summary

The study of a globally distributed Scrum team showed, that while the team was able to successfully carry out a globally distributed collaboration using of Scrum, some of the greatest challenges were related to the technological setups used to facilitate the collaboration. The team had access to and made extensive use of videoconferencing equipment, but when other tools were needed during meetings, problems arose and with no common place for the daily meetings to take place, meeting were delayed. The daily Scrum meetings became cumbersome to setup—which goes against their intended simple and fast intentions—and the demonstation meetings were held without the participation of the Indian developers. There seemed to be a need for smarter video-conferencing solutions that integrated with the other tools that were needed to support these specific activities.

3.2 SideBar

SideBar was out first technological exploration into the domain of interactive video. SideBar seeks to address the problem that personal connections are hard to achieve in distributed arrangements. The problems of distributed collaboration are linked to the fact that many of the processes that exists in collocated collaboration are disrupted by distances [28]. This also applies to the creation of personal ties and connections which have optimal conditions in collocated settings where everyday communication, spontaneous encounters, frequent meetings and interactions, and possibly planned team building activities all foster the creation of such connections. With the introduced distances in GSD, such facilitating processes are disrupted and studies of people in distributed teams show that their self-reported distributed social network size is smaller compared to collocated [30] and people tend to form groups with collocated colleagues [9]. Personal connections are not without importance however, and concepts such as trust, the feeling of group cohesion and connectedness have been linked to communication effectiveness and patterns in distributed environments [22, 40, 50].



Name: Alex Robertson
Department: Software Development and Testing
Title: Architect
Project: Apothecary Webshop

Me: Hi Kim, nice too see you again, and what a match last night! Too bad ManU conceded a goal in injury time. I know you are a fan. Not a fair result in the end

kim: Hi Peter, nice to see you too. Did not know you were a ManU fan. Just a minor set back on the road to the title [...]

Bug #1134
Hi, i think i manged to figure out...

Status ?
What's the status on item 4?...

Re:Re:Question...
Allright. I think i found a workaround...

Figure 3.2: The SideBar system adds tablet computer for all meetin participant. Using these tablets, partcipants can seek out information about each other and engage in one-on-one chat conversations.

3.2.1 System Description

SideBar addresses the problem that interpersonal connections are much harder to achieve in distributed collaboration compared to collocated [4]. SideBar is a videoconferencing system that aims to support social activities during video meetings using interactive video. The main feature of SideBar is that the traditional videoconferencing setup is extended with personal tablets for all meeting participants. These tablets show an interactive mirrored videofeed from the conference camera. Using facial-tracking, the tablet video is made interactive by superimposing small rectangles on top of the faces of people in the video. As the SideBar system uses face tracking not face recognition, a login procedure is necessary to make a connection between the user and the face as tracked by the image recognition algorithm. This association is made by a login procedure in which a user—when starting the video-meeting—is shown the video-stream of the local site in which she is located. She then selects herself in the interactie videofeed to make the connection between the face and the person. Figure 3.2 depicts a conceptualization from the design process of a SideBar meeting—meeting participants are using tablets during the meeting to interact with each other.

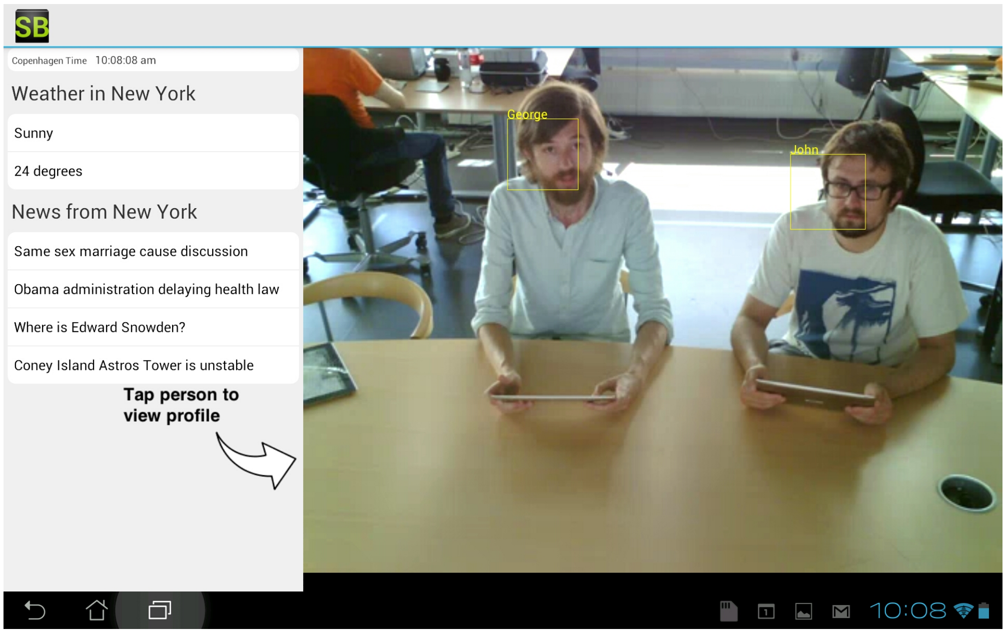

SB
Copenhagen Time 10:08:08 am
Weather in New York
Sunny
24 degrees
News from New York
Same sex marriage cause discussion
Obama administration delaying health law
Where is Edward Snowden?
Coney Island Astros Tower is unstable
Tap person to view profile
George
John
10:08

FIGURE 3.3: SideBar uses face-detection to track people in a videomeeting and provide interactive content overlay.

Users can tap an interactive rectangle on the tablet videofeed to navigate to a personal profile for the corresponding person. The personal profiles in SideBar contain personal and professional information about the person such as name, position and personal interests, access to a direct one-to-one chat as well as information about the location of the person, and information about the team as a whole. With SideBar, people can seek out information about each other during the meeting and engage in one-to-one—so called sidebar—conversations. Figure 3.3 shows the SideBar tablet user interface in a meeting situation. The videofeed from the conference camera is shown on the tablet and yellow rectangular interactive areas are superimposed onto the faces of people in the video along with the names of the people. Tapping these rectangles will bring up the profile page for that person. On the side of the videofeed, the time of day, the weather and recent news about the remote location is shown.

3.2.2 Evaluation

SideBar was evaluated in a small scenario-based usability study [16] with the purpose of assessing the perceived usefulness of the different features of SideBar as well as the ease of use. 7 participants (mean age 32, all male) were recruited for the evaluation which was conducted in two sessions. Each participant was given a role such as user interface designer or developer and was asked to play out a scenario in which a mobile application was to be produced and they had to agree on the division of work. The details of the scenario and the procedure can be found in [P2]. After completing the scenario, participants were asked to rate the usefulness of different features of SideBar on a Lickert scale and state to what extend they agreed with the statements *The system is easy to use* and *The use of tablets distracts the video meeting.* The session concluded with an open ended interview in which participants could elaborate on their answers and provide additional comments and suggestions on the system. Figure 3.4 shows a snapshot from the an evaluation session.

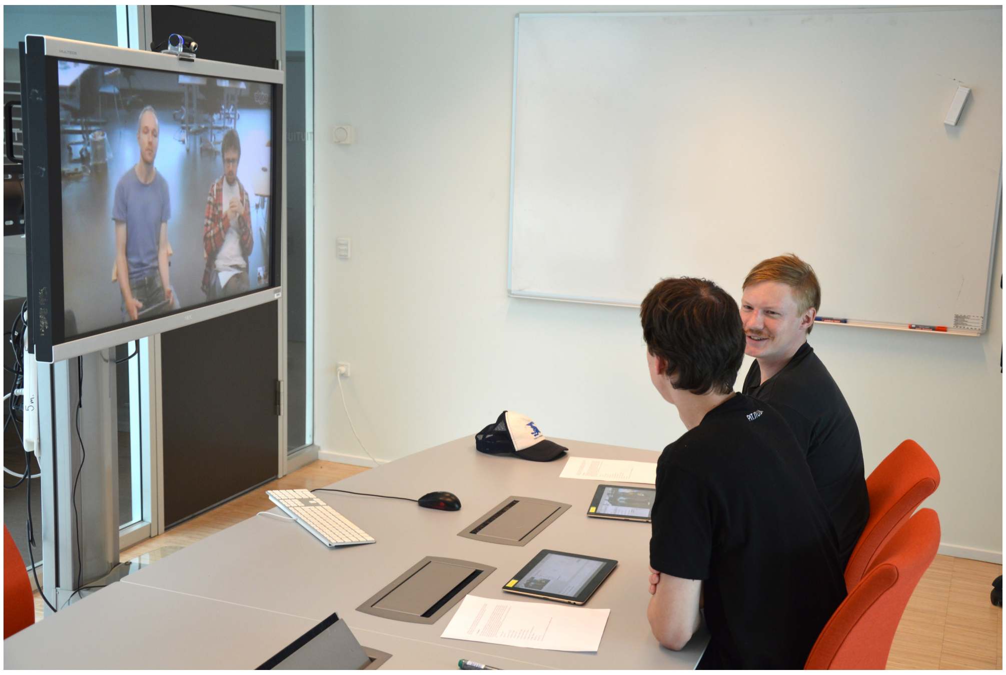


FIGURE 3.4: A snapshot from the SideBar evaluation.

Table 3.5 shows the results of the Lickert scale questionnaire reported as the minimal score, the first quartile, the median, the third quartile, the maximum score and the interquartile range. In general, the participants found SideBar useful and reported that SideBar was *"really helpful"* and *"the right way to go"*. The interactive videofeed and the personal profiles scored especially high in the evaluation. In the semi-structured interview that followed participants stated that *"[I] felt surprised how useful integrating information and video is"* and that the combined video-content interface *"added more depth to the video meeting"* and even requested more information be readily available on the video instead of in the profile pages of SideBar. The participants also found the system easy to use which was also visible from observing the evaluation scenario as it played out as all participants confidently navigated to the profiles of each other seeking out information using the interactive video.

Feature	*Min*	*Q1*	$\tilde{x}$	*Q3*	*Max*	*iqr*
Interactive video	4	4	4	5	5	1
Personal profiles	3	3.25	4	5	5	1.75
Location page	1	1.5	3	3	4	1.5
Communication backchannel	4	4	4	5	5	1
Team Page	1	1.5	4	4	5	2.5
Statement	*Min*	*Q1*	$\tilde{x}$	*Q3*	*Max*	*iqr*
The system is *easy to use*	3	4	4	4	5	0
The use of tablets *distracts* the video meeting	1	1.25	4	4	4	2.25

FIGURE 3.5: Questionnaire result on a 5-point Likert scale. For each feature, the table shows the reported minimum score (*Min*), the first quartile (*Q1*), the median ($\tilde{x}$), the third quartile (*Q3*), the maximum (*Max*) and the interquartile range (*iqr*).

3.2.3 Summary

We designed and implemented SideBar to explore how to support more social engagement in videoconferencing—these details can be found in [P2]. With respect to the concept of collaborative windows, designing, implementing and evaluating SideBar provided us with two main insights that helped inform the concept. *First*, SideBar demonstrates how—from a technical perpective—video and interactive content can be combined by superimposition of the two. *Second*, the evaluation—while limited in size—showed that participants found the interactive video especially useful and during the scenarios we observed how all participants navigated using the video-content user interface without problems. The work on SideBar showed us how to combine video and content which lead us pursue this concept and helped us develop the initial concept of collaborative windows which was the initial basis for the design of the dBoard as explained in the following section.

3.3 dBoard

The Scrum board is an important artifact in the Scrum development process. The traditional Scrum board is a large physical wall—such as a whiteboard—divided into columns representing states. Tasks to be implemented are represented on small post-it notes (or similar small objects) and placed in the first column representing the 'New' state. As tasks are being implemented they are moved from the 'New' through 'In Progress' to 'Done'. The Scrum board thus reflects the current state of implementation and provides awareness about work to developers which is one of the most sought information types of software developers [41]. The board has been subject to research and it has been identified as an important tool in reducing the amount of articulation work during a standup meetings [57] and as an object around which many other scrum processes revolve [49]. Previous research have problematized the digitization of tasks boards [65, 73] but in order to provide a Scrum board to a distributed Scrum team, either physical boards should be set up and manually synchronized across sites or digital ones be used. Recent research have shown how distributed teams have adopted digital solutions [8, 26] to their needs but there is still a need for solutions that facilitate communication as well as task board for distributed teams [26].

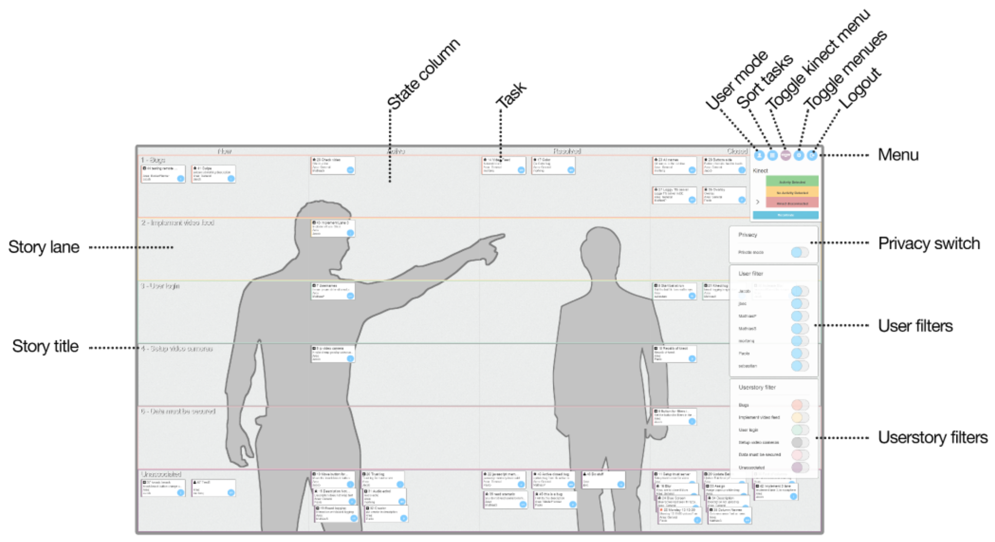


FIGURE 3.6: The dBoard is a combined video-communication tool and scrum board.

3.3.1 System Description

We designed the dBoard to provide distributed Scrum teams with a task board that supported both task management and communication. Based on our ethnographic workplace study of a distributed Scrum team (Section 3.1) and previous literature on Scrum boards (e.g. [26, 49, 57]), we designed the dBoard with the following requirements. The dBoard should:

- Provide active support during scrum meetings
- Function as an information radiator
- Give awareness about the presence of remote co-workers
- Seamlessly transition between information and awareness radiator and meeting support tool
- Integrate with existing software engineering tools
- Be easily movable and deployable
- Provide mechanisms for ensuring the privacy feeling of the users.

3.3.1.1 Scrum Board

The dBoard is a combined digital Scrum board and video-communication tool. A synchronized digital Scrum board is superimposed on top of an always-on full screen video-stream providing a shared communication and coordination surface for a distributed Scrum team. The Scrum board is laid out with columns representing states and rows representing user stories. Tasks are represented as small post-it notes that are placed in the row-column intersection corresponding with their state and the user story they belong to. The tasks show their title, a snippet of the description and the name of the developer assigned to them. The Scrum board is touch enabled and users can move tasks around using touch or re-assign tasks to each other from a drop-down menu. Performing a tap-and-hold gesture on a task brings up a sidepanel with detailed information about the task's description, estimate and change-history. The dBoard also has menues to filter out tasks based on user or user story and provides features for sorting the board by automatically placing tasks in their right row-column intersection. Figure 3.6 shows an illustration of the dBoard user interface with the user interface components explained.

3.3.1.2 Window Metaphor

We designed the video-communication features of dBoard to function as a VideoWindow [21]. Using a 'window' metaphor in our design, the dBoard provides an always-on full screen video channel between two connected sites. The intention here is that the dBoard should support ad-hoc meetings at the board and awareness of the presence and activities of remote team members not just planned meetings after which video is switched off. The window metaphor is also visible in the user interface where the video takes up the entire screen as if is was a window.

3.3.2 Proxemic Interaction

The dBoard is designed to function both as a passive information radiator, as a tool supporting the daily standup meeting with a Scrum board and as a facilitator for ad-hoc or unplanned meetings at the board and provides features for transitioning between these modes. Using proxemic approach applied to the video [60], the dBoard uses a Microsoft Kinect sensor mounted below the screen to automatically adjust its audio and video based on the proximity of people at the board. The dBoard operates in three different modes depending on the proximity of people in both ends. In the first mode where people are present in both ends, the video is clear and the audio turned on. In the second mode where no people are present on either side, the video is blurred on both sides and the audio is turned of. In the last third mode where people are present in only one site, the audio is kept off but video is blurred less than in the first mode so that it is clear that people are at the board. Requesting the attention of remote co-workers from the board is as simple as performing a ‘knocking’ gesture on the board which will play a ‘knock-knock’ sound in the other end.

3.3.3 Backend Integration

Lastly, we integrated the information on the dBoard (tasks, user stories, bugs, users) with existing task-tracking systems. This is a two-way integration; changes made on the dBoard (such as updating the state of a task) are updated in the task-tracking software and vice versa. Currently, the dBoard integrates with Microsoft Team Foundation Server[1], but using NooSphere [36] as an in between layer between task-tracking system and the dBoard, it is possible to add other task tracking systems without changing the dBoard software.

To evaluate the dBoard we conducted two studies: a scenario-based user evaluation with Scrum practitioners and a field deployment in a distributed Scrum team as described in the following.

3.3.4 User Study

To evaluate the dBoard and get feedback on the design, we conducted a scenario-based user evaluation [16]. 7 participants from 3 different companies that all use Scrum participated in the evaluation. Participants conducted the evaluation with their colleagues resulting in three evaluation sessions. In each of these sessions, two researchers played out the roles of confederates to create a distributed team consisting of the evaluation participants on one side and the two researchers on the other. Two scenarios were conducted involving common tasks in a Scrum team. The first task involved working distributed on separate tasks and updating the Scrum board when done, then contacting the remote team. The second task was a standup meeting in which participants took turns updating tasks. Again, we refer to the paper for details on the methodology [P3]. After the scenarios, the participants were asked to fill out a Technology Acceptance Model (TAM) [18] questionnaire to which we added four questions about the usefulness of specific dBoard features. At the end of each evaluation session, a small open-ended interview was conducted in which the participants were asked about their experiences with the dBoard and had the opportunity to express any comments or feedback that they might have.

[1]`https://msdn.microsoft.com/en-us/vstudio/ff637362.aspx`

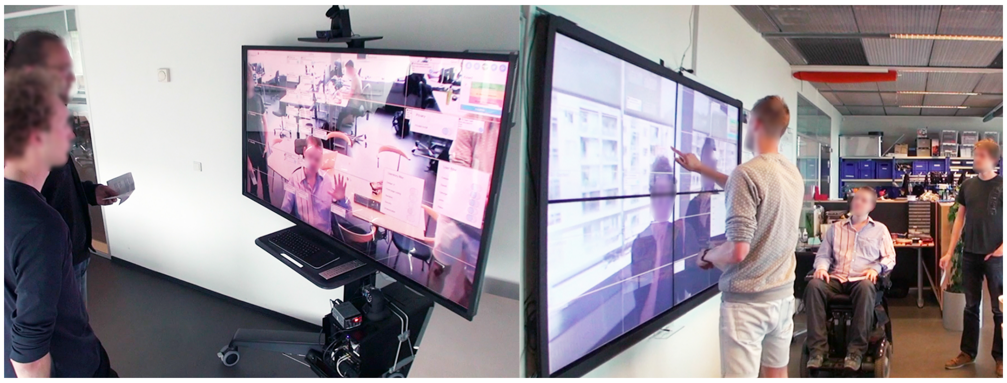


FIGURE 3.7: Evaluation participants performing a standup meeting scenario during the evaluation of the dBoard.

The results from the questionnaire is shown in Figure 3.7. In general, participants were positive towards the usefulness of the dBoard. This was especially prevalent in the evaluation of the different features which all scored high whereas the TAM-specific scores were more moderate yet still positive. In particular, the video was well received. As one participant expressed *"The entire video is really cool—really helpful."*, while another noted that *"You are actually interacting with your colleagues that you do not normally see"*. The Scrum board as well received positive scores and all participants appreciated the interactive Scrum board. As one participant noted: *"One of the things I like of a Scrum board like this is that you have not the digital but the post-its like interactions, and you can walk up there, take a task up here, and put it over here. I know you can do it on your computer—it might be easier—but I just like that feeling, I like to be able to do that"*. Indeed, we designed the dBoard with inspiration from physical Scrum board.

In general, the participants expressed that they found the dBoard useful and ease to use. The combined video and Scrum board in particular was well received and participants agreed that this was a promising direction for a collaborative Scrum board. This was also observed during the evaluation scenarios where all participants interacted with the board with ease.

3.3.5 Field Deployment

To understand how the dBoard supported daily standup meetings in a real world, we subjected the dBoard to a field deployment [66]. The purpose of the deployment was to understand how the dBoard supported the Scrum processes of a distributed Scrum team. We collaborated with a small Danish software company that in 2013 engaged in a collaboration with a medium-sized German software company. This created a distributed team in which the Danish developers and German developers form one coherent Scrum team responsible for the development of one of the German companies products.

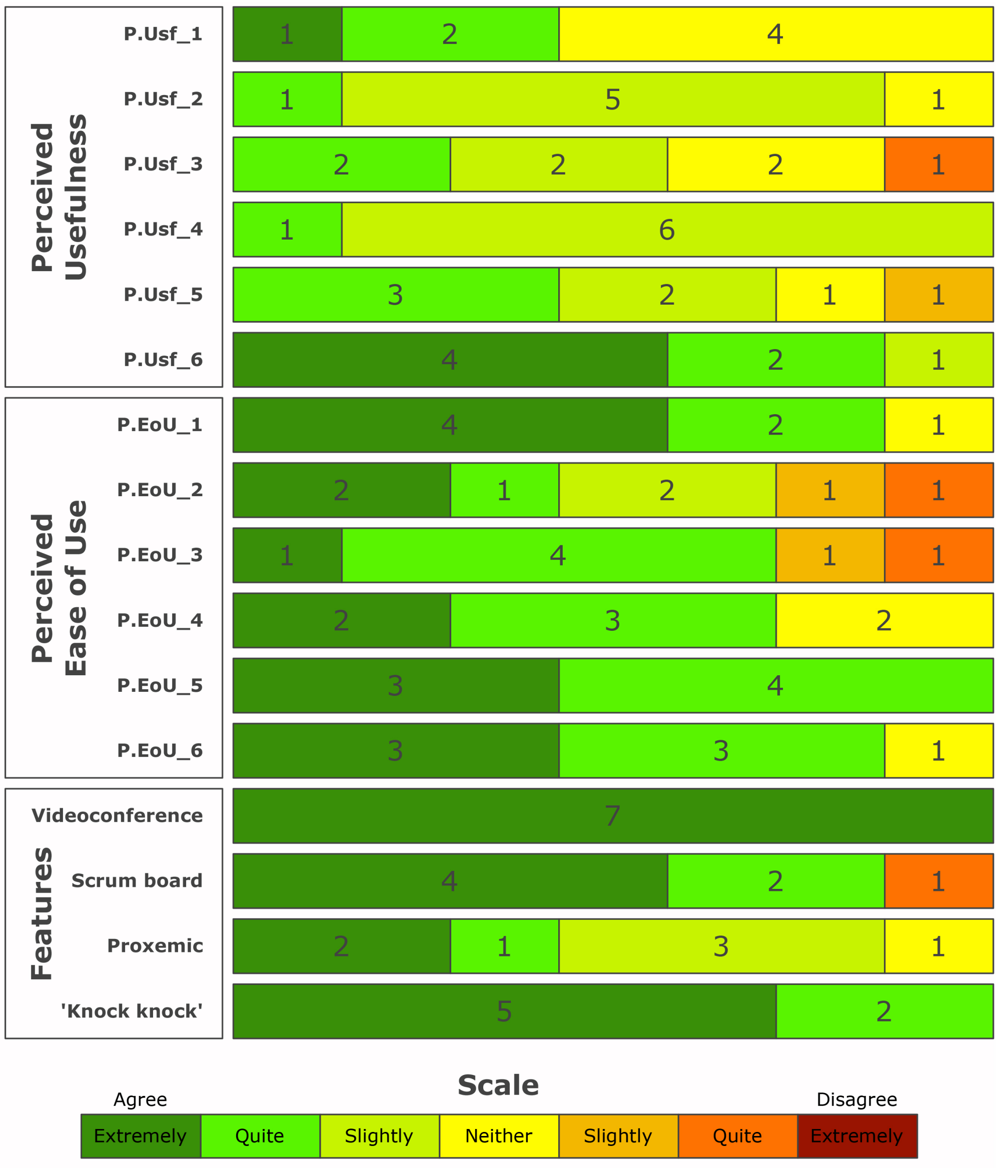


FIGURE 3.8: The results of the user evaluation of dBoard

Following the collaboration agreement with the German company, the Danish company had opted to change to using Scrum as their software development framework.

3.3.5.1 Pre-deployment study

To first gain an understanding of how team conducted these meetings, we observed the Scrum meeting practices of the team for a period of four and a half months. Meanwhile we configured the dBoard to integrate with their backend systems. During the time of observations, the team conducted 27 standup meetings. Figure 3.10 shows a picture of the setup used for the meetings. The Scrum masters desk was raised to accomodate the standup nature of the meetings and a Team Foundation Server Scrum board plugin was launched on the computer. When it was time for the meeting, the Scrum master called out the meeting and the team gathered around his desk. They then took turns explaining their tasks. From our observations it became clear that these meetings constituted an important setting for coordination and knowledge sharing between the developers. Taking turns explaining ongoing and future work, the meetings provided a setting within which the team became aware of each others work which often was follow by discussions and other knowledge sharing activities relating to implementation of specific tasks.

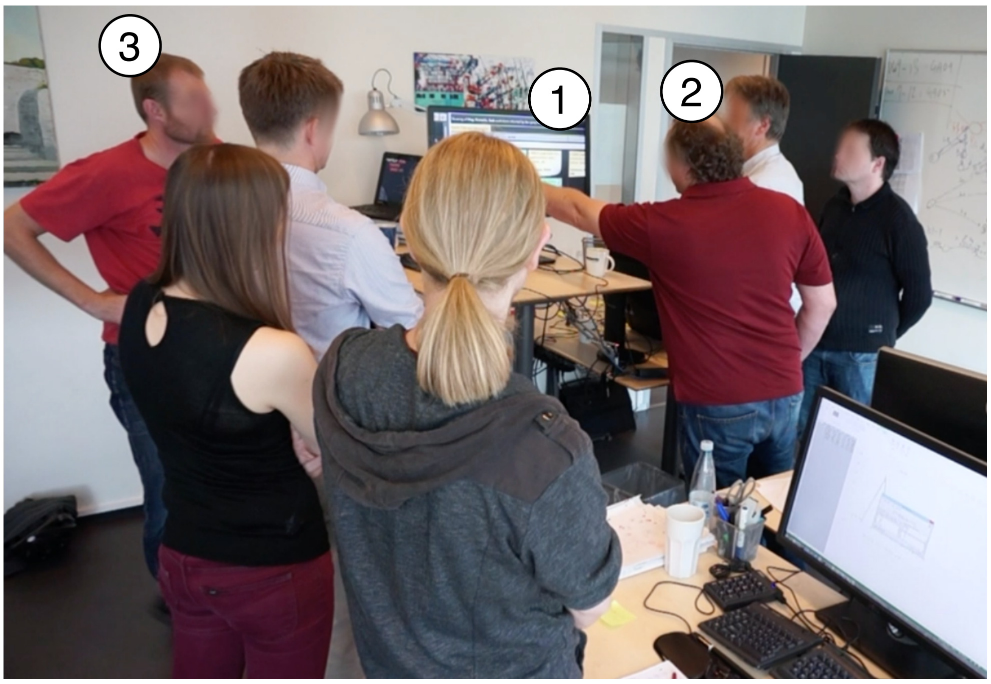


Figure 3.9: With little space around the Scrum board (1) the screen had to be turned to face the person speaking (2) leaving others with no direct view to the board (3).

We also observed several problems which we grouped into three distinct categories. First, we observed how the fact that the German team members were not included in the meeting caused problems. In several occasions the Danish developers had questions about the completion of tasks that they could not get an answer to as the German team members were not included in the meeting. In such situations, they would have to contact the German team by phone after the meeting. Second, we observed how the physical location where the meeting took place did not provide adequate space for all team members to be included in the meetings. Figure 3.9 shows a typical situation from a meeting. The limited space did not provide enough space for all people to see the Scrum board due to the amount of desks in te room so the Scrum master had to turn his screen to whomever was speaking if that person needed to see the tasks on the scrum board. And third, the software and hardware used did not support all the activities of the meeting. The small screen used to present the Scrum board was only visible to the people standing nearest and in order to see all tasks and user stories, much panning and zoomning in the program had to be done during the meetings.

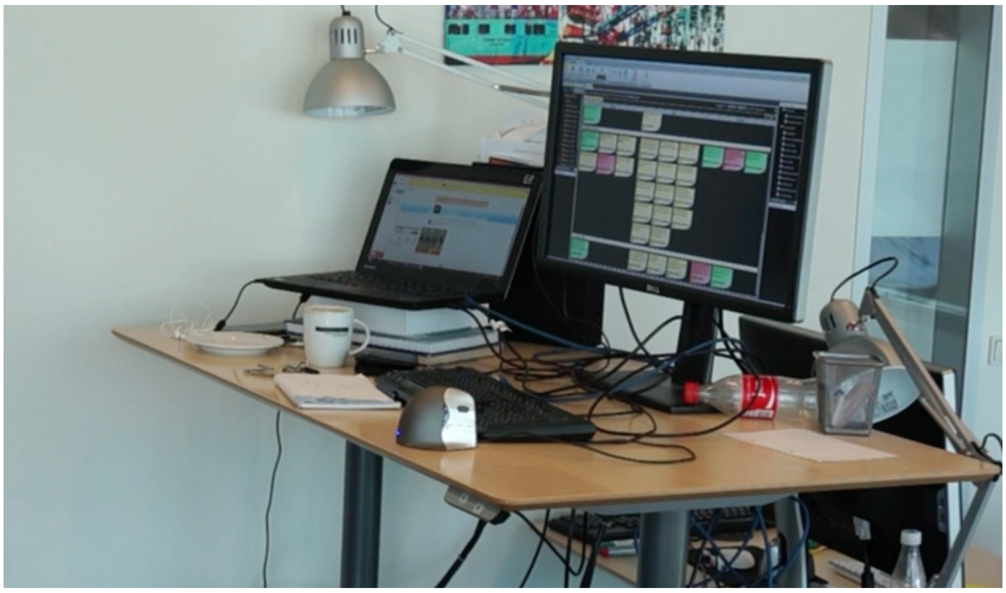


FIGURE 3.10: The desk around which the collocated Scrum meetings took place. The Scrum board used is visible on the monitor.

3.3.5.2 Deployment

The pre-study showed us that there was a need for a Scrum board that both supported distributed Scrum meetings and provided enough space around it and screen real estate to provide support for the inclusion of all team members during the meeting. Following the pre-study of the Scrum meeting practices, we therefore deployed the dBoard system as it is designed exactly to support such issues. Two dBoards were deployed at the offices of the distributed Scrum team. With the dBoards deployed, we observed six standup meetings between the people in Denmark and Germany.

From these observations it became clear, that the combined videoconferencing and Scrum board provided the team with a tool for effectively carrying out these meetings. The meetings were conducted in a similar way as the collocated ones; each developer took turns in explaining ongoing and upcoming tasks and much knowledge sharing was conducted however, now, the German team were included in the meetings. During the meetings, the video enabled the team to communicate with their distant colleagues and the Scrum board overlay provided them with a shared tool to coordinate their work. The problems that were observed during the pre-study were all adressed by the dBoard system which allowed all team members to carry out their meeting with the support of a Scrum board and in general the users interacted with the board and with each other with ease.

Interestingly, we observed how the teams did not interact much with the tasks on the board. As a requirement of the dBoard was to have it integrated with existing task tracking systems, we had configured the dBoard to integrate with the Microsoft Team Foundation Server system that the companies were using to manage their development. This integration we believe is one of the major advantages of digitizing the Scrum board as it allows task being changed on the board to be correctly updated the in the backend systems which are used to track the status of the work and later reflect on how work was carried out. What we observed however, was that the team interacted very little with the dBoard during standup meetings as it in many cases was in the correct state when the meeting was initiated. During the day, each developer opened and closed tasks they were working on and the board thus was mostly in the correct configuration when the meetings were started.

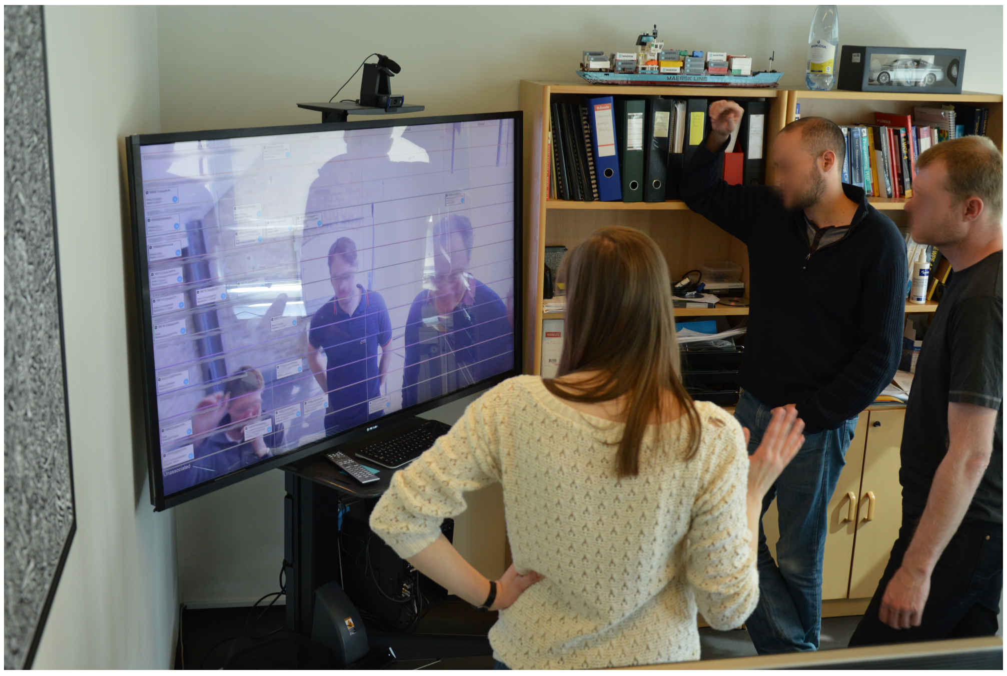


Figure 3.11: The dBoard in use during a distributed Scrum meeting

As described in detail in paper [P4], the field deployment of the dBoard also showed us some unpredicted usage. The user study of the dBoard showed a general high perceived usefulness of all features of the system but when we deployed it, we observed how it was only used as tool for planned standup meetings. The German side of the team decided to place their dBoard in a large meeting room. This decision was taken as the German team was located in a large office environment with other teams and the German team leader did not want to disturb the other teams with a video-channel in the office. The board has been designed to be placed among developers so the decision to place it in the meeting room removed its effects as a passive information radiator and as a window into the Danish office. With the dBoard in the meeting room, it served solely as a tool supporting the pre-planned standup meetings. Furthermore, we also observed a low meeting frequency during the deployment of the dBoard. While this partly was caused by vacations and upcoming relseases of the product which , we also saw how Scrum meeting were only planned if both teams could participate. This was an interesting observation as it demonstrates how the organizational arrangements—in this case the specific implementation of Scrum—also has an impact on a tool like dBoard.

3.3.6 Summary

The dBoard is a combined video-communication and Scrum board tool designed for distributed Scrum teams. We conducted two evaluations of the dBoard: a user study and a field evaluation. Both evaluations showed that the combination of video and Scrum board was highly appreciated and supported the daily Scrum meetings of a distributed Scrum team. The user study also showed that the features of dBoard were positively appreciated and that users considered the dBoard easy to use. On the other hand, the field deployment also revealed problems related to the dBoard. In particular we observed how—despite a design process where the companies were highly involved—the usage of the dBoard was limited to a few pre-planned meetings due to the placement of the dBoard. With the placement of the dBoard in the meeting room in the German office building, it lost its effects of serving as a passive information radiator and an ad-hoc meeting tool. Arguably, we would have seen other outcomes had the teams decided to place the dBoards in other location however, the field deployment demonstrates the physical and organizational effects on the adoption of a technology like the dBoard.

3.4 In Conclusion

This thesis asks the question of how we might design technologies supporting closely coupled distributed work such as that of distributed agile software development. Following a triangulation approach, this chapter provided technical and empirical perspectives on the question. In total, three overall studies were conducted: an ethnographic workplace study of globally distributed Scrum and the design, implementation and evaluation of two video-based collaboration tools. The ethnographic work-place encompassed two weeks of observations and interviews of a distributed Scrum team working from Denmark and India. The results from the study showed that despite the successful collaboration between the distributed actors, several challenges related to the technical setup of the collaborative technologies arose. In specific, the lack of integration between video and task-tracking systems and the lack of stable easy to setup meeting technologies caused delays in the daily collaboration.

The design of the two video-based collaboration tools SideBar and dBoard showed how the integration of video and content can be achieved through superimposition of the two. These combined user interfaces provided support for social engagement and distributed Scrum meetings in SideBar and dBoard respectively. The evaluations of the two showed a high degree of acceptance and ease of use of this way of designing video-based communication systems. The user study of the dBoard also showed how an always on video-connection and the use of proxemic interaction in video-communication to mediate between different modes of operation were perceived well by users. Finally, the field deployment of the dBoard showed how in a real world setting, a combined video-communication and Scrum board tool provided support for Scrum meetings but also showed how the implementation of Scrum and the physical location of the dBoard affected its usage.

Chapter 4

Collaborative Windows

Based on the workplace study of distributed Scrum and the work of designing, implementing and evaluating SideBar and dBoard, this dissertation argues, that future system designers might benefit from adopting the concept of *collaborative windows* as a user interface concept describing a particular type of systems well suited for supporting distributed collaboration. This chapter unfolds this concept and presents a design space analysis that can be used to design such collaborative windows.

4.1 Definition

A collaborative window is a user interface concept that describes a particular type of video-mediated communication tool. Collaborative windows are characterized by three properties that—combined—distinguish them from other video-based collaborative systems. *First*, collaborative windows apply a 'window' metaphor to their video-communication [21]. This entails that such systems are designed to work not only as tools for planned meetings but as windows into distant locations from which awareness information can be gathered and ad-hoc meetings started. *Second*, collaborative windows overlay content on top of their video to provide a tight integration between the video-communication feature and task specific tools. This content is interactive and collaborative in the sense that it is designed for interaction and for supporting collaboration. *Third*, collaborative are context aware as they adapt their behaviour based on their surroundings. In the following, these properties are explained in detail. Figure 4.1 shows the dBoard visualized as a collaborative window—a virtual window in the wall combined with a distributed virtual Scrum board provides a view into a distant location and allows for distributed Scrum meetings.

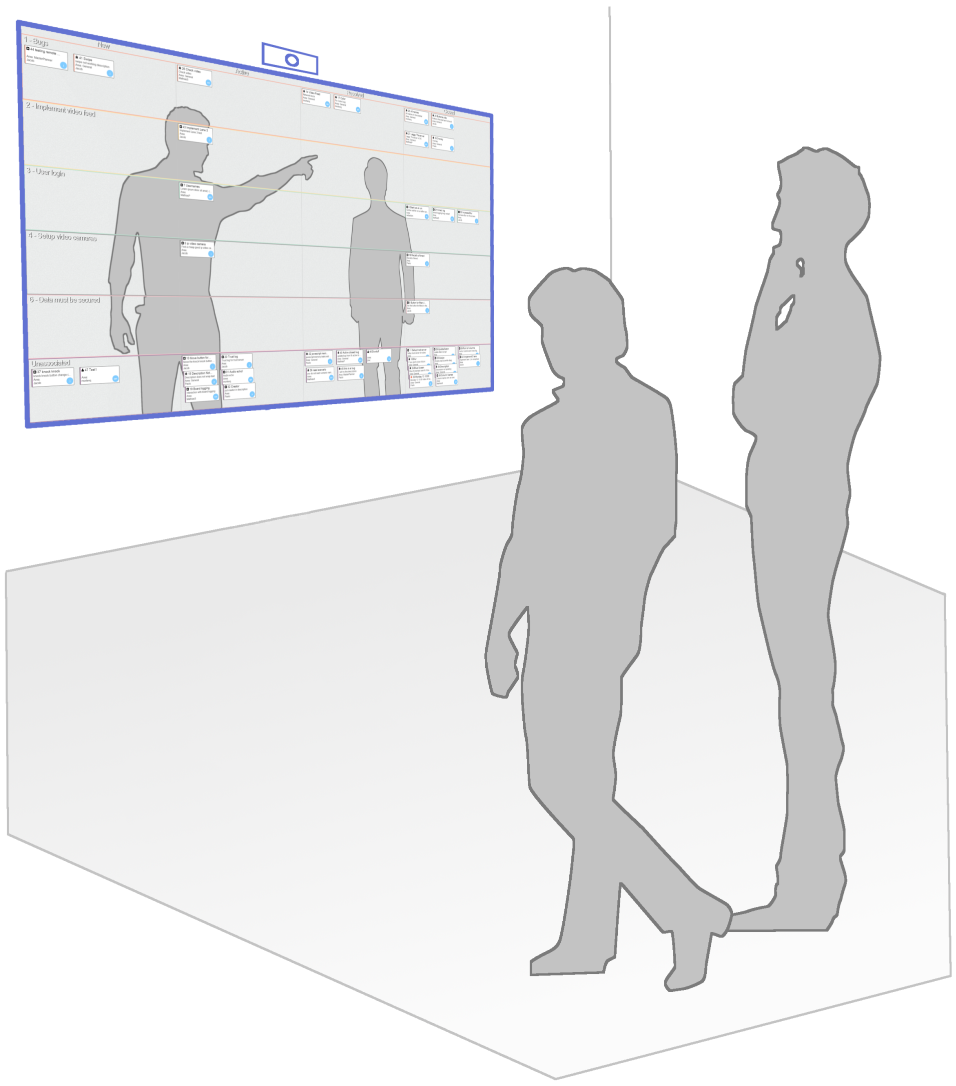


FIGURE 4.1: The dBoard conceptualized as a collaborative window.

4.1.1 Window Metaphor

Collaborative windows employ a 'window' metaphor to their video streams. As in VideoWindows [21], collaborative windows are designed as a way of seing through a window and into a distant office. This is opposed to traditional video conferencing systems that usually are designed to be started for planned meetings and stopped afterwards while being located in dedicated meeting rooms. With a window metaphor, collaborative windows support not only planned distributed meetings but allow users to have ad-hoc meetings when passing by the window or simply gather awareness about the presence of remote co-workers [6, 19].

The window property of collaborative windows should be implemented on a hardware and on a software level. On a hardware level this entails that a collaborative window should be implemented on a large wall-mounted screen—possibly embedded into the wall—to mimic a window into another place and that the camera-parallax problem is solved to allow correct transmission of gaze and gestures. On a software level, designing for the window metaphor entails that the video should be full-screen and always on. The idea is create the technological setup that—as much as possible—mimics a 'real' window and the feeling of looking into a distant location.

4.1.2 Content Overlay

Collaborative windows integrate video and content by superimposing content of top of video. The purpose of this is to provide a shared collaborative surface between distributed sites and to avoid having different tools for handling video-communication and other tasks. Content overlay seeks to solve the problems that videoconferencing technologies are detached from the other tools needed in videoconferencing. The hypothesis behind this property is, that it is possible combine video and task-specific tools by superimposing interactive content on top of the video task rather than keeping content and the video in separate applications.

One important aspect of the content overlay in collaborative windows is, that the content is interactive. This property affords collaboration on the window and distinguishes this property from previous work on content on video which doesn't enforce such interactivity [43, 51]. The dBoard for example, overlays an interactive synchronized Scrum board on top of the video that allows people to work on a shared Scrum board. The content–user stories and tasks—is interactive in the sense that users can move tasks on the board using touch and the content is synchronized across two boards such that movement on one board is reflect in real time on the other. As a result, the dBoard supports collaborative work on a Scrum board which is integrated with the video used for communication.

4.1.3 Context Awareness

With an alway-on window providing a view into another location, collaborative windows applications should implement context awareness to adapt to the environment in which they are embedded. Context awareness is the property of systems that can change their behaviour through sensing of their surrounding [62]. Context awareness was proposed in the wake of the ubiquitous computing paradigm. Ubiquitous computing as coined by Weiser was the vision describing a future where computers moved into the world and became invisible [72]. In such a world, context awareness is important as computers should be able to change their behaviour to accomodate the many different contexts in which they could be used when moved from the desktop and out into the 'real' world.

Collaborative windows are an example of a ubiquitous computing technology. Serving as an interactive window built into the environment, collaborative windows weave into the environment much like Weiser's vision [72]. Because of this factor, collaborative windows implement context awareness to make them aware of their surroundings and usage situations.

The dBoard implements context awareness to meidate its role of meeting supporting tool and passive information radiator. Using a proxemics approach [27] applied to the video stream [60] the dBoard turns on and off audio and blurs its video based on the proximity of people. If no people are in front of two connected dBoards, audio is turned off and video is blurred to ensure privacy and enhance the visibility of the Scrum board overlay serving as a passive information radiator. If people approach the dBoards, the video is de-blurred and the audio turned on to allow for communication. In SideBar, context awareness is implemented by being aware of who is in a meeting through means of face tracking.

4.2 Design Space

Based on the work of designing, implementing and evaluating the technologies presented in Chapter 3 and formulating the concept of collaborative windows , this dissertation presents a design space analysis for the design of systems building on top of the concept of collaborative windows. The design space dimensions arose from the choices that were made during the design and implementation of SideBar and dBoard and from the user studies and field deployment. The design space can be used to guide the design of future collaborative window systems or to reason about existing. The design space consist of six dimensions each associated with continuous or discrete values. As seen in Figure 4.2, these dimensions are Content Synchronization, Information Density, Interaction Awareness, Video-Content Connectivity, Privacy and Video-Content Inversion which are explained in detail below:

4.2.1 Content Synchronization

Content Synchronization refers to the degree to which the content overlay of collaborative windows are synchronized between connected windows. The dimension can take on a continuous value ranging from non synchronized to fully synchronized. On non synchronized collaborative windows, the content overlay on connected collaborative windows are not synchronized and collaborating actors work on each their interface. On

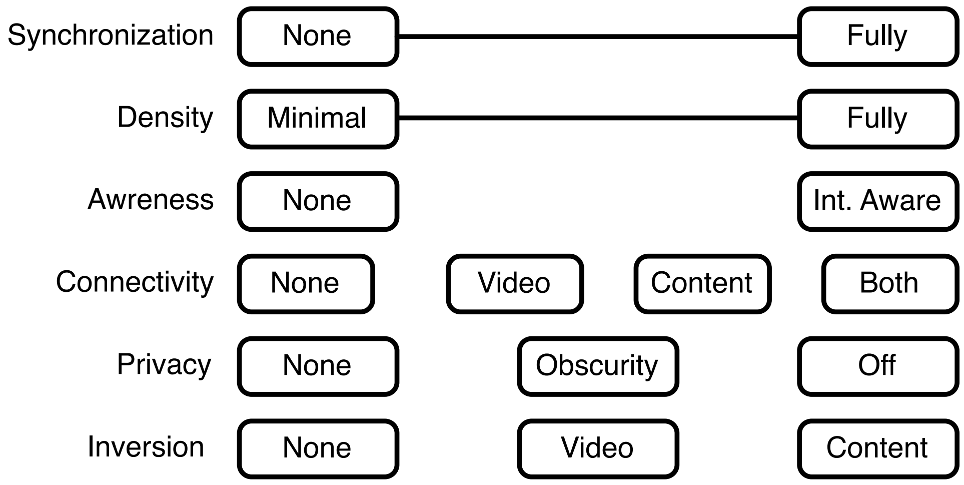


FIGURE 4.2: The 6 dimensions of the collaborative window design space and possible values. The first two dimensions (Synchroniation and Density) can take on a continuos value, the remaining can take on a discrete value.

fully synchronized collaborative windows, the content is synchronized in real time between connected windows. As a continuos dimension, degrees of synchronization can be achieved

The dBoard is an example of highly syncrhonized collaborative window. All tasks, user stories and bugs are synchronized between two dBoards and movement of tasks are updated in real time. Only the placement of the filtering menues are not synchronized. This synchronization provides users of the dBoard with feeling of working on a shared distributed Scrum board.

4.2.2 Information Density

Information density refers to how much content is presented on top of the video. It is a continuous dimensions with values that range from minimal—in which little information is presented—to fully covered where the entire video is covered with content. This dimension is important to consider in the design of collaborative windows as the amount of information placed on top of the video affects how much of the video is visible. In the case of minimal content, usually, it isn't necessary to consider how the video is affected, but as the amount of information grows, such considerations become more important. If most or all of the screen is covered with content, it may become necessary to opacify the content or employ other strategies for making sure that communication can happen undisturbed.

SideBar and dBoard both implement a low degree of information density. As seen in Figure 4.3 the high resolution screen of the dBoard leaves space for many tasks to be

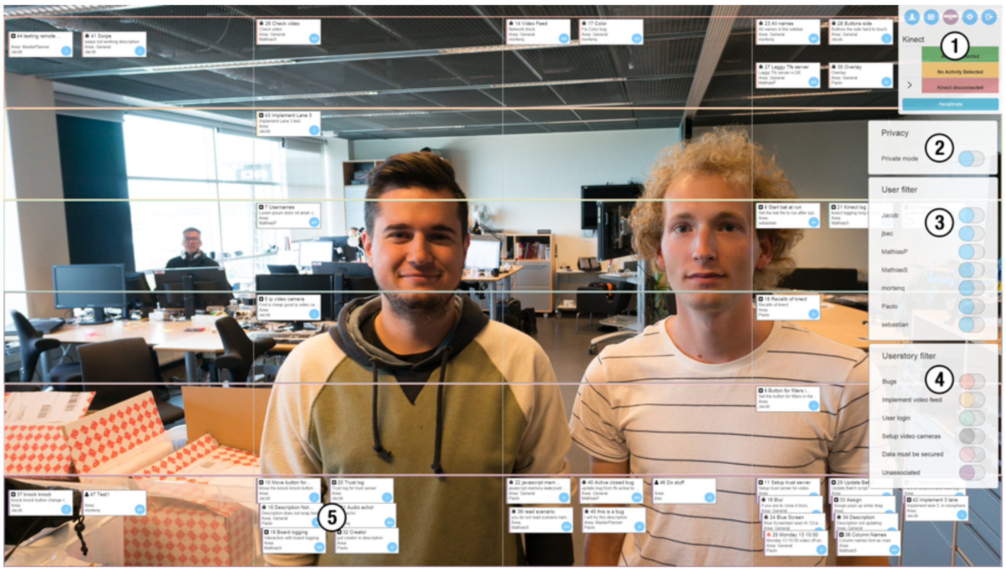


FIGURE 4.3: Tasks are overlayed the video on the dBoard but with a low degree of information density.

placed while still keeping the majority of the video visible. During the user evaluation of the dBoard however, several participants mentioned that they were concerned about how the video would perform if the board was fully covered with user stories and tasks. Indeed, in such cases, measures should be taken to ensure that the video remains visible to the meeting participants.

4.2.3 Interaction Awareness

Interaction awareness refers to awareness of interaction with content on the window. This dimension becomes important to consider if the collaborative window implements a high degree of content synchronization and if camera parallax distors gestures towards the content. In such systems, awareness of what is happing on another connected collaborative window becomes important, and interaction awareness should be considered. Interaction awareness can be implemented through touch pointers or visualizations of arms and hands [23].

The content overlay of dBoard is highly synchronized and the parallax between camera and screen does not allow the video to capture arms or hands of people interacting with the board. Therefore, the dBoard implements interaction awareness by highlighting tasks in red when they are moved and by showing touch pointers on the screen as shown in Figure 4.4.

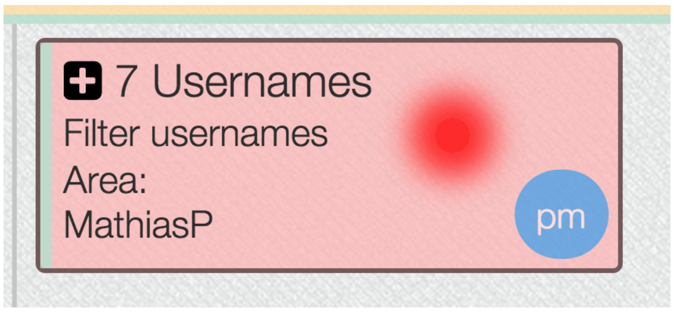


FIGURE 4.4: Interaction awareness on the dBoard is implemented using task-highlighting and touch pointers.

4.2.4 Video-Content Connectivity

The Video-Content Connectivity dimension refers to the extend to which the video and the overlaid content are connected. This dimension can undertake four discrete values; None, Video, Content, Both. In the None situtation, video and content are disconnected and the behaviour of one does not affect the other. In the Video situation, the video is connected to the content overlay but not vice-versa. In this situation, changes to the content can affect the video. The Content situation refers to the situation the other way around. In this situation, changes in the video changes the content overlay. In the last situation, there is a two-way connection between video and content and changes in either affects the other.

While dBoard does not implement video-content connectivity—the video and the Scrum board do not affect each other—SideBar uses a 'Content' approach. Using face tracking, the SideBar tablet application overlays interactive rectangles on top of the video. As people in the video-feed move, the rectangles move and as such there is a connection between the content and the video. The video however, does not change with changes to the content.

4.2.5 Privacy

The Privacy dimension describes how the collaborative window deals with privacy issues that arise from having an always-on video connection in the shared workspace. While the application seeks to mimic collocation by using a window metaphor which implies that a remote site is always visible it still might feel intrusive to the users and therefore this dimension should be considered in the design of a collaborative window. The privacy dimension applies to the video-feed between remote sites and is not to be confused with privacy of data.

We have identified three discrete values for this dimension; None, Obscurification and Off. The None value refers to systems that do not take into account the privacy of users. In this mode, the video communication channel is always-on and no privacy measures are taken. The Obscurification value refers to privacy by obscurity [64][1]. In this mode, the privacy of users is protected by means of obscuring the video at various times. This can be achieved by blurring the video which has been shown as effective means of balancing awareness and privacy in video [10]. In the off mode, privacy is achieved by completely turning off the video at certain times.

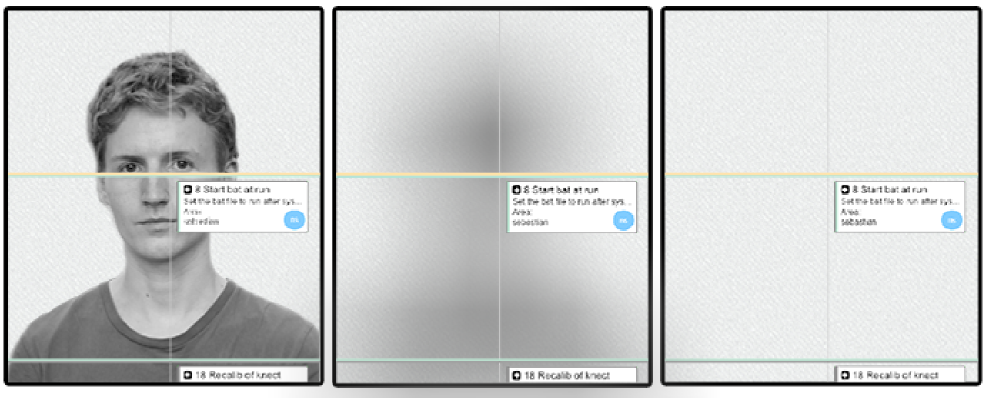


Figure 4.5: The dBoard uses proximity sensing to blur the video.

We implemented privacy by obscurity in dBoard. Taking a proximity approach by using a Kinect sensor, the dBoard senses the presence of people and blurs or de-blurs the video accordingly. Furthermore, we also implemented a 'privacy' button that when switched on turns the privacy dimension of dBoard to 'Off'.

4.2.6 User Interface Inversion

When designing collaborative windows it is also important to take into account the horizontal alignment of video and content overlay. As content is overlaid the video, users might want to gesture towards the user interface which isn't possible unless content or video is flipped. The user interface inversion dimension refers to wether the video or the content of a collaborative window are inverted or 'flipped' to allow gestures towards the user interface to be conveyed correctly. This dimensions can take on three discrete values; None, Content or Video.

In the None situation, neither video nor content is flipped and gestures such as pointing towards the screen come across mirrored. In the Content situation, the content is flipped

[1]Sellinger and Hartzog use the concept of obscurification when speaking about privacy of big data from e.g. social media sites and search engines. Here, we use the concept in relation with video as the concept describes the act of transforming data that could be used to identify people.

in one end. In this situation it is possible to keep the alignment of text and images using a relaxed WYSIWIS approach in which selected areas of the user interface are flipped [45]. The relaxed WYSIWIS approach can be used to flip for example images and text leaving their position on the window unchanged. In this way the spatial orientation of the image or text element is correct (i.e. on the left-most side on one window and on the right-most site of the other) but the text or images are equally readable from both perspectives. The last situation is Video in which the video is flipped. This situation might allow for pointing towards the screen correctly but in turn renders the perspective on the video mirrored.

In essence, whichever method is chosen has consequences for such systems. Flipping content to implement strict or relaxed WYSIWIS interfaces is only possible if the content can be mirrored on one side. The Scrum board for example has meaning in in its horizontal orientation—tasks move from left to right as they progress from 'New' to 'Done'. Other applications however might flip the content without problems. On the other hand, the video could be flipped to facilitate pointing towards user interface elements on the screen. This method however is not without consequences either as, the window the does not provide the 'correct' view into the distant location.

The dBoard does not invert video or content and thus falls into the category none. While this did not seem to have an impact on the usefulness of the system, it was mentioned in one of the evaluation scenarios.

4.3 SideBar and dBoard

Using this design space, we can reason about SideBar and dBoard as collaborative windows. While these system weren't designed using the collaborative window metaphor–rather the concept was formulated based on the systems—both systems implement some of the requirements and design dimensions of a collaborative window. These figures help to illustrate how systems might be depicted in the collaborative windows design space.

Figure 4.6 shows SideBar positioned in the collaborative windows design space. SideBar does not implement content synchronization, has a low information density, does not use interaction awareness, implements the content value of video-content connectivity, does not implement a privacy feature and has no inversion of video or content.

Figure 4.7 shows dBoard as positioned in the collaborative windows design space. The dBoard implements a high degree of content synchronization, low information density,

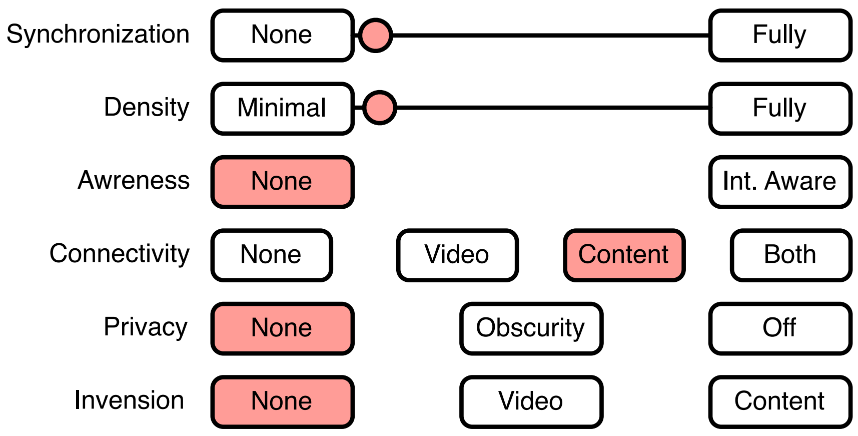


FIGURE 4.6: SideBar positioned in the collaborative windows design space.

interaction awareness through touch pointers and task highlights, no video-content connectivity, obscurity and off modes to perserve privacy (a user can select the option) and no inversion of video or content.

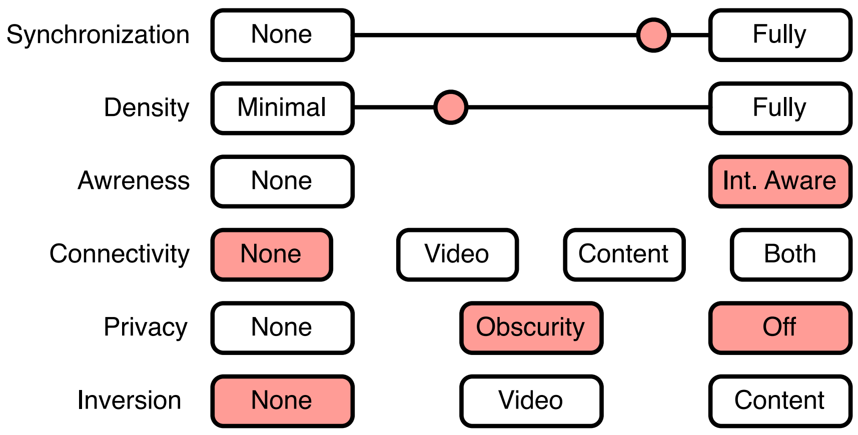


FIGURE 4.7: dBoard positioned in the collaborative windows design space.

4.4 Summary

This chapter presented the notion of *collaborative windows* as a user interface concept that describes a particular type of collaborative systems designed to support closely coupled distributed collaboration. A collaborative window is a video-based communication tool that implements three main properties. *First*, a collaborative window is built around a window metaphor which entails that the video is alway on and that the window—in software as well as hardware—seeks to give users a sense of looking through and collaborating on a window and into a distant location. *Second*, a collaborative window presents interactive content on the video using a content-on-video approach in which the content is superimposed onto the video. And *third*, collaborative windows are context aware so they can adapt their behaviour to the context in which they are situated and used. Furthermore, this chapter presented a design space consisting of six dimensions that should be considered when designing a collaborative window. These dimensions are: Content Synchronization, Information Density, Interaction Awareness, Video-Content Connectivity, Privacy and User Interface Inversion.

Chapter 5

Discussion

The previous chapter introduced the notion of *collaborative windows* as a user interface concept describing a particular type of collaborative systems. The concept was derived from an ethnographic field study of distributed Scrum that revealed problems related to the technological setup and the design, implementation and evaluation of two video-based communication systems: SideBar and dBoard. Combining the findings from these different studies lead us to suggest the concept as a novel user interface concept particularly well suited for technologies supporting closely coupled distributed work. This chapter discusses the concept of collaborative windows in terms of the limitations of the studies presented and the generalizability of the concept to other domains.

5.1 Window Metaphor

The window metaphor of a collaborative window seeks to address the problems of lack of awareness and casual encounters. This property however also enforces contraints on the technology that sets limits on what such a system can achieve. A consequence of the window metaphor property is that connecting more than two sites becomes problematic. Indeed, it is perfectly possible from a technology point of view to design systems that can connect several sites. Such an idea could be realized by dedicating different parts of the user interface to different video-streams from all the connected sites. Doing so however, is a violation of the window metaphor property that—if followed strictly—dictates that the window should be designed to look like a physical window which does not allow multiple video-streams on the window. Connecting several sites could then be realized by setting up different collaborative windows—one for each site—side-by-side. Such a setup though also requires more space.

The systems presented in this dissertation also suffer the problem of camera parallax. As much previous research has focussed on solving these issues (e.g. [68, 71, 74]) we did not implement features to account for gaze and gestures however, if we compare a collaborative window to a physical window, eye-contact, gaze and gestures should be transmitted correctly creating the illusion of standing on each side of a window. Instead, in this dissection we focussed in bringing together video and interactive content on window-like and context-aware collaborative applications. Future explorations into the domain of collaborative windows would benefit from taking into account gaze, eye-contact and gestures. A combination of e.g. and dBoard and ConnectBoard [68] would constitute an intersting system that could elaborate on the findings presented in this dissertation and move forward the research on collaborative windows.

5.2 Limitations

The concept of collaborative windows as proposed in Chapter 4 was coined based on empirical and technological research on tool support for distributed Scrum. Through an ethnographic workplace study of distributed Scrum we were able to point out how the lack of a stable meeting supporting technology and the problems of setting up multiple tools caused problems leading to delays in Scrum meetings. Through the design, implementation and evaluation of the two tools SideBar and dBoard we showed how such problems could be solved. The results from these studies combined lead us to suggest the notion of collaborative windows as a user interface concept that, we argue, is especially useful in designing technologies for closely coupled distributed work. The tools presented in Chapter 3 however, were not built on the concept of collaborative windows and the evaluations conducted thus do not evaluate the concept as a whole. We acknowledge that despite all the empirical material gathered and presented in this dissertation more research on collaborative windows should be conducted in order to evaluate systems that implement all the properties of such systems. Along these lines, it is also important to point out that a limited number of participants were included in all the studies. Nonetheless, this dissertation presented empirical and technical results that point towards the usefulness of the concept of collaborative windows.

5.3 Other Domains

The work presented here has been carried out within the domain of globally distributed Scrum, and the results presented should be read and understood within this domain. We do however believe, that it is possible to take the concept of collaborative windows into other domains. While SideBar was designed and evaluated in a software development setting, the specific purpose of the system—supporting social engagement in videoconferencing—is not tied to the domain per se. dBoard is more closely related to software development as one of the main features is the Scrum board—an important artifact in the Scrum development process. Task boards however, are also found in domains outside software development. Kanban boards for example, are widely used in other domains such as engineering and while such boards are not identical to Scrum boards, the overall intentions are the same: tasks to be completed are placed on the board and transition through various stages resulting in a clear visual depiction of the progress of work. As such, the technologies presented here could be appropriated to other domains with much modification.

A particular feature for the concept of collaborative windows is that the concept has been designed based on the assumption of closely coupled work. Indeed, previous research have suggested strategies of reducing collaboration [13] to handle the complexities of distributed work and strategies such as segregation [24] and avoidance [33] have described as coordination mechanisms in collaborative work. When bringing the concept of collaborative windows into other domains it is important to consider the *kind* of work and how the collaboration is handled. As we saw in the field deployment of the dBoard, the adoption of the technology was highly dependant on the strategies that were used for handling the collaboration. While the team used Scrum, they did not have daily meetings. As a consequence, the dBoard was used less frequently than intended. Collaborative windows have been designed for closely coupled work and how such systems perform if other strategies are used remains unclear.

Chapter 6

Conclusion

This chapter provides a conclusion to Part I of this dissertation. Chapter 2 described the context and background of the dissertation by explaining the usage of video-communication in distributed Scrum and what challenges still remain. Chapter 3 presented the research methods and the three studies that were conducted in exploring how to provide technology support for distributed Scrum with a focus on how they helped give an answer to the research question. In particular, the chapter presented the ethnographic workplace study that provided an empirical perspective on the technological problems of distributed Scrum and the two collaboration tools SideBar and dBoard that were used to explore the technological perspectives. Chapter 4 introduced the notion of collaborative windows as a way of designing systems for distributed work and and provided a design space analysis of such systems and finally, Chapter 5 discussed the concept, its limitation and its possible applicability into other domains.

6.1 Answer to Research Questions

This dissertation asked the question *How do we design technologies that support closely coupled collaborative work such as that of global Scrum?*. To provide an answer, three main studies were undertaken. Through a workplace study of distributed Scrum between Denmark and India, we found that while Scrum was used effectively to keep collaboration close and handle the dependencies in global work, issues were encountered related to the technical setup used to handle the collaboration. In particular, the lack of a stable meeting tool and the disconnection between video-conferencing equipment and task tracking system caused delays and problems in the meetings.

From a technology perspective two video-based collaboration tools were designed. SideBar was a first exploration into the domain of collaborative video and was a video-communication system designed to support social engagement in videomeetings. Afterwards, we designed and implemented the dBoard—a combined video-communication and Scrum board which utilized a content-on-video approach to superimpose a syncrhonized, digital Scrum board top of a full screen, always-on video. These tools were evaluated in user studies and the dBoard subjected to a field deployment which showed that users found the tools both useful and easy to use.

Based on the results from these studies, this dissertation provided an answer to the aforementioned research question by proposing the notion of collaborative windows—a user interface concept that can be used to design collaboration tools especially targeted closely coupled distributed work. A collaborative window is a user interface concept describing a particular type of video-mediated system that implements three overall properties; the video-communication features are designed using a window metaphor, interactive content is superimposed onto the video and context awareness is used to adapt behaviour based on the environment. The dissertation also presented a design space consisting of six dimensions that can be used to design future collaborative window systems.

6.2 Future Work

The research presented in this dissertation demonstrates early work on the concept of collaborative windows and we identify several areas in which more research should be conducted. First and foremost, a collaborative window that strictly implements the features of a window metaphor, content-on-video and context-awareness should be implemented. Such a system would require much engineering effort however, could provide valuable contributions to research on teleconferencing and collaborative systems. Second, a long term field deployment of a collaborative window should be conducted to asses the long term effects of a window embedded in the workplace. Furthermore, the notion of collaborative windows should be taken into other domains in which closely coupled distributed work is carried out.

Another intersting line of research would be to explore the potential of the Video-Content Connectivity dimension. In particular, a connection between video and content as a way of interacting with a collaborative window could be realized through adapting the user interface based on the video. SideBar demonstrated an example of how video and content can be connected and it would be interesting to take these findings and implement them in a collaborative windows system.

Bibliography

[1] Abras, C., Maloney-Krichmar, D., and Preece, J. User-centered design. *Bainbridge, W. Encyclopedia of Human-Computer Interaction. Thousand Oaks: Sage Publications 37*, 4 (2004), 445–456.

[2] Bannerman, P., Hossain, E., and Jeffery, R. Scrum practice mitigation of global software development coordination challenges: A distinctive advantage? In *System Science (HICSS), 2012 45th Hawaii International Conference on* (Jan 2012), 5309–5318.

[3] Beck, K., Beedle, M., van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., Kern, J., Marick, B., Martin, R. C., Mellor, S., Schwaber, K., Sutherland, J., and Thomas, D. Manifesto for agile software development. `http://agilemanifesto.org`, 2001.

[4] Bjørn, P., and Christensen, L. R. Relation work: Creating socio-technical connections in global engineering. In *ECSCW 2011: Proceedings of the 12th European Conference on Computer Supported Cooperative Work, 24-28 September 2011, Aarhus Denmark*, S. Bødker, N. O. Bouvin, V. Wulf, L. Ciolfi, and W. Lutters, Eds. Springer London, 2011, 133–152.

[5] Bjørn, P., Esbensen, M., Jensen, R. E., and Matthiesen, S. Does distance still matter? revisiting the cscw fundamentals on distributed collaboration. *ACM Trans. Comput.-Hum. Interact. 21*, 5 (Nov. 2014), 27:1–27:26.

[6] Bly, S. A., Harrison, S. R., and Irwin, S. Media spaces: Bringing people together in a video, audio, and computing environment. *Commun. ACM 36*, 1 (Jan. 1993), 28–46.

[7] Boden, A., Nett, B., and Wulf, V. Trust and social capital: Revisiting an offshoring failure story of a small german software company. In *ECSCW 2009*, I. Wagner, H. TellioÄlu, E. Balka, C. Simone, and L. Ciolfi, Eds. Springer London, 2009, 123–142.

[8] Boden, A., Rosswog, F., Stevens, G., and Wulf, V. Articulation spaces: Bridging the gap between formal and informal coordination. In *Proceedings of the 17th ACM Conference on Computer Supported Cooperative Work & Social Computing*, CSCW '14, ACM (New York, NY, USA, 2014), 1120–1130.

[9] Bos, N., Shami, N. S., Olson, J. S., Cheshin, A., and Nan, N. In-group/out-group effects in distributed teams: An experimental simulation. In *Proceedings of the 2004 ACM Conference on Computer Supported Cooperative Work*, CSCW '04, ACM (New York, NY, USA, 2004), 429–436.

[10] Boyle, M., Edwards, C., and Greenberg, S. The effects of filtered video on awareness and privacy. In *Proceedings of the 2000 ACM Conference on Computer Supported Cooperative Work*, CSCW '00, ACM (New York, NY, USA, 2000), 1–10.

[11] Brubaker, J. R., Venolia, G., and Tang, J. C. Focusing on shared experiences: Moving beyond the camera in video communication. In *Proc. DIS 2012*, ACM (June 2012).

[12] Brush, A. B., Meyers, B. R., Scott, J., and Venolia, G. Exploring awareness needs and information display preferences between coworkers. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '09, ACM (New York, NY, USA, 2009), 2091–2094.

[13] Carmel, E., and Agarwal, R. Tactical approaches for alleviating distance in global software development. *Software, IEEE 18*, 2 (Mar 2001), 22–29.

[14] Christensen, L. R., Jensen, R. E., and Bjørn, P. Creating relation work: Characteristics for local and gloval collaboration. COOP '14, Springer (2014).

[15] Conchúir, E. O., , P. J., Olsson, H. H., and Fitzgerald, B. Global software development: Where are the benefits? *Commun. ACM 52*, 8 (Aug. 2009), 127–131.

[16] Convertino, G., Neale, D. C., Hobby, L., Carroll, J. M., and Rosson, M. B. A laboratory method for studying activity awareness. In *Proceedings of the Third Nordic Conference on Human-computer Interaction*, NordiCHI '04, ACM (New York, NY, USA, 2004), 313–322.

[17] Daft, R. L., and Lengel, R. H. Information richness: A new approach to managerial behaviour and organizational design. *Research in Organizational Behaviour 6* (1984), 191–233.

[18] Davis, F. D. Perceived usefulness, perceived ease of use, and user acceptance of information technology. *MIS Q. 13*, 3 (1989), 319–340.

[19] Dourish, P., and Bly, S. Portholes: supporting awareness in a distributed work group. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '92, ACM (New York, NY, USA, 1992), 541–547.

[20] Esbensen, M., and Bjørn, P. Routine and standardization in global software development. In *Proceedings of the 18th International Conference on Supporting Group Work*, GROUP '14, ACM (New York, NY, USA, 2014), 12–23.

[21] Fish, R. S., Kraut, R. E., and Chalfonte, B. L. The videowindow system in informal communications. ACM Press (1990), 1–11.

[22] Fulmer, C. A., and Gelfand, M. J. At What Level (and in Whom) We Trust: Trust Across Multiple Organizational Levels. *Journal of Management* (2012).

[23] Genest, A., Gutwin, C., Tang, A., Kalyn, M., and Ivkovic, Z. Kinectarms: a toolkit for capturing and displaying arm embodiments in distributed tabletop groupware. In *CSCW 2013* (2013).

[24] Gerson, E. M. Reach, bracket, and the limits of rationalized coordination: Some challenges for cscw. In *Resources, Co-Evolution and Artifacts*, Computer Supported Cooperative Work. Springer London, 2008, 193–220.

[25] Gomaa, H., and Scott, D. B. Prototyping as a tool in the specification of user requirements. In *Proceedings of the 5th International Conference on Software Engineering*, ICSE '81, IEEE Press (Piscataway, NJ, USA, 1981), 333–342.

[26] Gossage, S., Brown, J. M., and Biddle, R. Understanding digital cardwall usage. Technical report tr-15-01, Carleton University - School of Computer Science, June 2015.

[27] Greenberg, S., Marquardt, N., Ballendat, T., Diaz-Marino, R., and Wang, M. Proxemic interactions: The new ubicomp? *interactions 18*, 1 (Jan. 2011), 42–50.

[28] Herbsleb, J. Global software engineering: The future of socio-technical coordination. In *Future of Software Engineering, 2007. FOSE '07* (May 2007), 188–198.

[29] Herbsleb, J., and Moitra, D. Global software development. *Software, IEEE 18*, 2 (mar/apr 2001), 16 –20.

[30] Herbsleb, J. D., Mockus, A., Finholt, T. A., and Grinter, R. E. Distance, dependencies, and delay in a global collaboration. In *Proceedings of the 2000 ACM conference on Computer supported cooperative work*, CSCW '00, ACM (New York, NY, USA, 2000), 319–328.

[31] Higuchi, K., Chen, Y., Chou, P. A., Zhang, Z., and Liu, Z. Immerseboard: Immersive telepresence experience using a digital whiteboard. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, CHI '15, ACM (New York, NY, USA, 2015), 2383–2392.

[32] Hollan, J., and Stornetta, S. Beyond being there. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '92, ACM (New York, NY, USA, 1992), 119–125.

[33] Holten Møller, N., and Dourish, P. Coordination by avoidance: Bringing things together and keeping them apart across hospital departments. In *Proceedings of the 16th ACM International Conference on Supporting Group Work*, GROUP '10, ACM (New York, NY, USA, 2010), 65–74.

[34] Hossain, E., Babar, M., and young Paik, H. Using scrum in global software development: A systematic literature review. In *Global Software Engineering, 2009. ICGSE 2009. Fourth IEEE International Conference on* (july 2009), 175 –184.

[35] Hossain, E., Bannerman, P. L., and Jeffery, R. Towards an understanding of tailoring scrum in global software development: a multi-case study. In *Proceedings of the 2011 International Conference on Software and Systems Process*, ICSSP '11, ACM (New York, NY, USA, 2011), 110–119.

[36] Houben, S., Nielsen, S., Esbensen, M., and Bardram, J. E. Noosphere: An activity-centric infrastructure for distributed interaction. In *Proceedings of the 12th International Conference on Mobile and Ubiquitous Multimedia*, MUM '13, ACM (New York, NY, USA, 2013), 13:1–13:10.

[37] Isaacs, E., Whittaker, S., Frohlich, D., and O'Conaill, B. Informal communication re-examined: New functions for video in supporting opportunistic encounters. *Video-mediated communication 997* (1997), 459–485.

[38] Isaacs, E. A., and Tang, J. C. What video can and can't do for collaboration: A case study. In *Proceedings of the First ACM International Conference on Multimedia*, MULTIMEDIA '93, ACM (New York, NY, USA, 1993), 199–206.

[39] Jensen, R. E. Why closely coupled work matters in global software development. In *Proceedings of the 18th International Conference on Supporting Group Work*, ACM (2014), 24–34.

[40] Kiesler, S., and Cummings, J. N. What do we know about proximity and distance in work groups? a legacy of research, 2002.

[41] Ko, A. J., DeLine, R., and Venolia, G. Information needs in collocated software development teams. In *Proceedings of the 29th international conference on Software Engineering*, IEEE Computer Society (2007), 344–353.

[42] Kuechler, M., and Kunz, A. Holoport - a device for simultaneous video and data conferencing featuring gaze awareness. In *Virtual Reality Conference, 2006* (March 2006), 81–88.

[43] Kunz, A., Dehlin, S., Piazza, T., Fjeld, M., and Olofsson, T. Collaborative whiteboard: Towards remote collaboration and interaction in construction design. In *Proc. 27th International Conference on Applications of IT in the ABC Industry Accelerating BIM Research Workshop* (2010), 132–140.

[44] Lanubile, F. Collaboration in distributed software development. In *Software Engineering*, A. De Lucia and F. Ferrucci, Eds., vol. 5413 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2009, 174–193.

[45] Li, J., Greenberg, S., Sharlin, E., and Jorge, J. Interactive two-sided transparent displays: Designing for collaboration. In *Proceedings of the 2014 Conference on Designing Interactive Systems*, DIS '14, ACM (New York, NY, USA, 2014), 395–404.

[46] Luff, P., and al., e. *Workplace Studies: Recovering Work Practices and Informing System Design*. Cambridge University Press, 2000.

[47] Mackay, W. E., and Fayard, A.-L. Hci, natural science and design: a framework for triangulation across disciplines. In *Proceedings of the 2nd conference on Designing interactive systems: processes, practices, methods, and techniques*, DIS '97, ACM (New York, NY, USA, 1997), 223–234.

[48] Matthiesen, S., and Bjørn, P. Why replacing legacy systems is so hard in global software development: An information infrastructure perspective. In *Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work and Social Computing*, CSCW '15, ACM (New York, NY, USA, 2015), 876–890.

[49] McNely, B. J., Gestwicki, P., Burke, A., and Gelms, B. Articulating everyday actions: an activity theoretical approach to scrum. In *Proceedings of the 30th ACM international conference on Design of communication*, SIGDOC '12, ACM (New York, NY, USA, 2012), 95–104.

[50] Nardi, B. A. Beyond bandwidth: Dimensions of connection in interpersonal communication. *J. Comput.-Supp. Coop. Work 14* (2005), 91–130.

[51] Nescher, T., and Kunz, A. An interactive whiteboard for immersive telecollaboration. *Vis. Comput. 27*, 4 (Apr. 2011), 311–320.

[52] Niinimaki, T., Piri, A., Lassenius, C., and Paasivaara, M. Reflecting the choice and usage of communication tools in gsd projects with media synchronicity theory. In *Global Software Engineering (ICGSE), 2010 5th IEEE International Conference on* (Aug 2010), 3–12.

[53] Noll, J., Beecham, S., and Richardson, I. Global software development and collaboration: Barriers and solutions. *ACM Inroads 1*, 3 (Sept. 2011), 66–78.

[54] O'hara, K., Kjeldskov, J., and Paay, J. Blended interaction spaces for distributed team collaboration. *ACM Trans. Comput.-Hum. Interact. 18*, 1 (May 2011), 3:1–3:28.

[55] Okada, K.-I., Maeda, F., Ichikawaa, Y., and Matsushita, Y. Multiparty videoconferencing at virtual social distance: Majic design. In *Proceedings of the 1994 ACM conference on Computer supported cooperative work*, CSCW '94, ACM (New York, NY, USA, 1994), 385–393.

[56] Paasivaara, M., Durasiewicz, S., and Lassenius, C. Distributed agile development: Using scrum in a large project. In *Global Software Engineering, 2008. ICGSE 2008. IEEE International Conference on* (aug. 2008), 87 –95.

[57] Pries-Heje, L., and Pries-Heje, J. Why scrum works: A case study from an agile distributed project in denmark and india. In *Agile Conference (AGILE), 2011* (aug. 2011), 20 –28.

[58] Prikladnicki, R., Boden, A., Avram, G., Souza, C., and Wulf, V. Data collection in global software engineering research: learning from past experience. *Empirical Software Engineering* (2013), 1–35.

[59] Ramesh, B., Cao, L., Mohan, K., and Xu, P. Can distributed software development be agile? *Commun. ACM 49*, 10 (oct 2006), 41–46.

[60] Roussel, N., Evans, H., and Hansen, H. Mirrorspace: Using proximity as an interface to video-mediated communication. In *Pervasive Computing*, A. Ferscha and F. Mattern, Eds., vol. 3001 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2004, 345–350.

[61] Roussel, N., and Gueddana, S. Beyond "beyond being there": Towards multiscale communication systems. In *Proceedings of the 15th International Conference on Multimedia*, MULTIMEDIA '07, ACM (New York, NY, USA, 2007), 238–246.

[62] Schilit, B., Adams, N., and Want, R. Context-aware computing applications. In *Proceedings of the 1994 First Workshop on Mobile Computing Systems and Applications*, WMCSA '94, IEEE Computer Society (Washington, DC, USA, 1994), 85–90.

[63] Schwaber, K., and Sutherland, J. The scrum guide - the definitive guide to scrum: The rule of the game. `http://scrum.org`, 2011.

[64] Selinger, E., and Hartzog, W. Obscurity and privacy.

[65] Sharp, H., Robinson, H., Segal, J., and Furniss, D. The role of story cards and the wall in xp teams: a distributed cognition perspective. In *Agile Conference, 2006* (July 2006), 75–86.

[66] Siek, K. A., Hayes, G. R., Newman, M. W., and johntang. *Field Deployments: Knowing from Using in Context*. Springer, June 2014.

[67] Sutherland, J., Viktorov, A., Blount, J., and Puntikov, N. Distributed scrum: Agile project management with outsourced development teams. In *System Sciences, 2007. HICSS 2007. 40th Annual Hawaii International Conference on* (Jan 2007), 274a–274a.

[68] Tan, K.-H., Robinson, I., Samadani, R., Lee, B., Gelb, D., Vorbau, A., Culbertson, B., and Apostolopoulos, J. Connectboard: A remote collaboration system that supports gaze-aware interaction and sharing. In *Multimedia Signal Processing, 2009. MMSP '09. IEEE International Workshop on* (Oct 2009), 1–6.

[69] Tang, J. C., Zhao, C., Cao, X., and Inkpen, K. Your time zone or mine?: A study of globally time zone-shifted collaboration. In *Proceedings of the ACM 2011 Conference on Computer Supported Cooperative Work*, CSCW '11, ACM (New York, NY, USA, 2011), 235–244.

[70] Tøth, T. Trust in client-vendor relations: An empirical study of collaboration across national and organizational boundaries. In *Proceedings of the 5th ACM International Conference on Collaboration Across Boundaries: Culture, Distance and Technology*, CABS '14, ACM (New York, NY, USA, 2014), 5–14.

[71] Vertegaal, R., Weevers, I., Sohn, C., and Cheung, C. Gaze-2: conveying eye contact in group video conferencing using eye-controlled camera direction. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '03, ACM (New York, NY, USA, 2003), 521–528.

[72] Weiser, M. The computer for the 21st century. *SIGMOBILE Mob. Comput. Commun. Rev. 3*, 3 (July 1999), 3–11.

[73] Whittaker, S., and Schwarz, H. Meetings of the board: The impact of scheduling medium on long term group coordination in software development. *Computer Supported Cooperative Work (CSCW) 8* (1999), 175–205.

[74] Zillner, J., Rhemann, C., Izadi, S., and Haller, M. 3d-board: A whole-body remote collaborative whiteboard. In *Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology*, UIST '14, ACM (New York, NY, USA, 2014), 471–479.

Part II

Papers

Routine and Standardization

Title of Paper

Routine and Standardization in Global Software Development

Authors:

Morten Esbensen, *Pernille Bjørn*

Published:

*In Proceedings of the 18th International Conference on Supporting Group Work **(GROUP)**. ACM, 2014. p. 12-23.*

Abstract:

We present an ethnographic field study of a distributed software development team following the Scrum methodology. During a two-week period, we observed from both sites the collaboration between a Danish software company off-shoring part of their development to an Indian solution provider. Collaboration by its very definition is based on the notion of dependency in work between multiple people. Articulation work is the extra work required to handle these dependencies. In a globally distributed team, managing these dependencies is exacerbated due to the distances of time, space, and culture. To broaden our understanding of dependencies in a global context and how they influence work practices, we made them the focus of our analysis. The main contributions of this paper are (i) an empirical account of the dependencies that are part of the collaborative work in a global software development team, (ii) a discussion of the interlinked properties of dependencies, and (iii) an explanation of how the practices of standardization and routine are developed and used to manage these dependencies.

Routine and Standardization in Global Software Development

Morten Esbensen[*]
mortenq@itu.dk

Pernille Bjørn[*†]
pbra@itu.dk

[*]IT University of Copenhagen
Rued Langgaards Vej 7
2300 Copenhagen S, Denmark

[†]UCIrvine & Intel Center for Social Computing (ISTC)
Bren Hall, 6th Floor
Irvine, California, USA

ABSTRACT

We present an ethnographic field study of a distributed software development team following the Scrum methodology. During a two-week period, we observed from both sites the collaboration between a Danish software company off-shoring part of their development to an Indian solution provider. Collaboration by its very definition is based on the notion of dependency in work between multiple people. Articulation work is the extra work required to handle these dependencies. In a globally distributed team, managing these dependencies is exacerbated due to the distances of time, space, and culture. To broaden our understanding of dependencies in a global context and how they influence work practices, we made them the focus of our analysis. The main contributions of this paper are *(i)* an empirical account of the dependencies that are part of the collaborative work in a global software development team, *(ii)* a discussion of the interlinked properties of dependencies, and *(iii)* an explanation of how the practices of standardization and routine are developed and used to manage these dependencies.

Categories and Subject Descriptors

K.4.3 [**Organizational Impacts**]: Computer-supported collaborative work

Keywords

Global software development; ethnographic study; dependencies; standardization; routine

1. INTRODUCTION

Work in software development projects is increasingly being carried out by distributed teams. Distributed or global collaboration allows companies to take advantage of a global workforce, execute work at different hours of the day, and keep work close to different target markets. However, this way of working also introduces a number of problems related to the distances of time, space, and culture by increasing the "reach" [16]. With collaboration spanning multiple countries and time zones, one key challenge concerns the difficulties in handling dependencies as they emerge and become pertinent in the collaboration across geographical sites.

Collaboration by its very definition is based upon the notion of dependency in work between multiple people. People collaborate when *"they are mutually dependent in their work and therefore are required to cooperate in order to get the work done"* [28]. Collaborative actors are interdependent in their work, which requires them to articulate their distributed yet individual activities. Articulation work is the extra work required to handle the dependencies that arise in collaborative work [32], and has in prior research been identified as an important aspect of global software development [4]. However, this extra work tends to be neglected [25]. Strategies such as coordination or knowledge practices have been suggested to solve such issues [3, 29]; however, our interest in this paper is to examine how the articulation work involved in handling dependencies is related to practices of routine and standardization.

Initially we wanted to reveal the basic set of dependencies as they unfold in geographical distributed practices; however, examining our empirical observations it was clear that certain dependencies such as location, people, collaborative activities, and artifacts created particular conditions for how the work and collaboration could unfold in practice. More interestingly, however, it became clear that the key mechanisms software developers enacted when handling the challenges that emerge due to the dependencies in practice concerned standardization and routine. Therefore, the research question investigated in this paper concerns how the mechanisms of routine and standardization were enacted in practice by software developers when overcoming the dependencies that constituted global software development practices.

In this paper we zoom in and explore ethnographically which dependencies are constitutive of the globally distributed work practices within software development. In addition, we investigate how dispersed software developers handle and deal with dependencies in their work and, in particular, the role of standardization and routine. As an empirical case we chose to study global Scrum practices as they emerge between two geographical sites in Denmark and India. What makes this an excellent case to study dependencies is that the project was divided equally between developers at both sites and entailed many diverse types of dependencies, which the participants had to deal with on a daily basis. We had

GROUP'14, November 9–12, 2014, Sanibel Island, Florida, USA.
Copyright is held by the owner/author(s). Publication rights licensed to ACM.
ACM 978-1-4503-3043-5/14/11 ...$15.00.
http://dx.doi.org/10.1145/2660398.2660413.

the unique opportunity to follow the work practices between the two sites in detail over a period of 14 days, where one of us observed the work from the site in Denmark while the other observed the work from the Indian location, allowing us access to a global view of the collaborative setup [27]. By exploring these work practices we are able to create a rich understanding of the work as it emerged from the data, while zooming in on the strategies for how the geographical distributed partners succeeded in reducing the effort involved in articulation work through strategies of standardization and routine practices.

The main contributions of this paper are as follows: *(i)* an empirical account of the dependencies that are part of the collaborative work in a global software development team, *(ii)* a discussion of the interlinked properties of dependencies, and *(iii)* an explanation of how the practices of standardization and routine are developed and used to manage these dependencies. Interestingly, we found that standardizing the technology needed for the work seemed much harder than standardizing the processes of work.

The paper is structured as follows. First, we present related work on global software development, dependencies, and standardization and routine. Then we present our empirical case, data material, and analysis method. This is followed by a results section presenting our empirical data structured into subsections, each focusing on particular dependencies that constitute the global software development practices. Then we discuss the empirical findings by enfolding previous literature on standardization and routine.

2. DEPENDENCIES IN GLOBAL SOFTWARE DEVELOPMENT

The term *dependencies* refers to coupling among entities or relations, where change in one area might produce certain changes (sometimes unanticipated) in other areas. When a dependency arises in collaborative work, a certain reliance is created, where individual work becomes interlinked with the work of others, and therefore articulation work is required [32]. Thus, collaborative work by its very definition is based on the notion of dependencies, and articulation work is the work involved in handling these dependencies. Previous research on dependencies in distributed software collaboration projects tends to focus on the software dependencies that affect programming activities [e.g. 7, 11, 19] or documents [8]. Software dependencies occur when one part of a program depends on another, and since most larger IT systems are created by multiple people working together, software dependencies are of crucial importance to understanding the often tight coupling between people.

Although software and document dependencies are important, our interest is to zoom out from the concrete software code and documents, and instead focus on the organizational practices of coordination, which are critical to handling the articulation work in global software development. Dependency in work is part of the definition of collaboration, and much research has thus been conducted in terms of figuring out how practitioners involved in collaboration accommodate the challenges of articulation work [5, 25], for example, in terms coordination [16, 18], knowledge management [3, 22], commitment and transparency [31], or awareness [10]. In this paper, we are particularly interested in coordination as a strategy to handle articulation work. Malone and Crowston define the very process of coordination as *"the act of managing the interdependencies between activities"* [24, p. 90] and suggest that we may further investigate the processes of managing dependencies by "characterizing different kinds of dependencies and identifying the coordination processes that can be used to manage them" [24, p. 91]. In order for us to unravel the coordination activities in global software work, we will need to look at the dependencies as they manifest themselves in everyday work and what strategies are used for managing these.

Coordination is a strategy to handle articulation work [e.g 16, 21, 28] and in earlier studies it has been investigated in different ways: as coordination mechanisms where a protocol for work is embedded within an artifact [29], as coordination by avoidance [21], or as the difference between segregation and standardization [16]. Segregation or avoidance are both strategies where the aim is to keep work separated and reduce connections within the work, thus reducing the complexity of coordination. As a strategy this has been applied in global software development, where practices of decomposing and recomposing have been identified [18], or in terms of minimizing interaction across sites [20]. In contrast, the strategy of standardization is quite different; instead of trying to divide the work into smaller pieces, thus reducing the relations, standardization seeks to keep the relations within the work but instead create the same (or at least similar) conditions for the collaboration across people. In global software development one of the challenges is to create standardization of practices across multiple heterogeneous sites, since the basic conditions for collaboration are dependent on the geographical site where the people are physically located. The world is not flat [35].

Standardization has often been conceptualized as top-down imposed processes where practitioners are forced or disciplined into working in certain ways guided by the categorization schemes embedded within the technology [1, 33]. However, recent studies on standardizations in practice have shown that standardizations are not stabilized and predetermined entities but instead malleable and negotiated [6], and thus standardization from a practice perspective [15] comprises incomplete and co-constructed practices, where people enact and make the standards work for them in practice [13].

While standardization is a difficult strategy to employ in global work, one of the interesting aspects we found in our data was that this was the preferred strategy of the practitioners, and thus we wanted to investigate why that is and how they manage to deploy the standardization strategy as a general approach to reduce the complexity of articulation work. So our interest is to see how standardization is achieved as a collective and emergent accomplishment [26] where the various dependencies, which affect the collaborative engagement, are coming into place.

3. METHOD

To investigate the enactment of routine and standardization in global software development practice, we examined the empirical observations from our workplace study [23], focusing on the collaborative practices that arose in a global software development team distributed across Denmark and India. What makes this approach particularly applicable for such a study is that we were able to experience the practices as they unfolded rather than forcing a particular theoretical conceptualization on our data. For two weeks we observed the work practices simultaneously from two geographical locations, one observer in Denmark and one in India, collecting data material about their work practices. We also interviewed 12 employees in total.

After the empirical observation, all notes were aligned and the interviews were transcribed. Since the study was designed with one researcher being in India while another was in Denmark, the first part of the analysis was zooming in on the observed practices from each location. To analyze the data, each researcher first went through their own notes, conducted within-case analysis [12], and annotated the dependencies that began to emerge from the data material. We used a grounded-theory-inspired approach [17] where candidates for dependencies emerged from the data rather than applying predefined theoretical categories from the literature. Following the within-case analysis we began to compare the data across the cases by conducting cross-case analysis [12]. This work was done through a shared document within which we identified important actors, activities, and technologies and the relations and associations between them. This approach allowed us to take a broad perspective on the data material, from which we believe we were able to identify many of the dependencies that existed in the work during the observation period.

During our data analysis, we created a dependency graph in which we tried to visualize the connections between observed dependencies. However, we quickly saw, as this graph grew in size, that the dependencies we had observed were all somehow linked in a much larger structure encompassing the whole observation period. Figure 1 shows a small excerpt of our dependency graph.

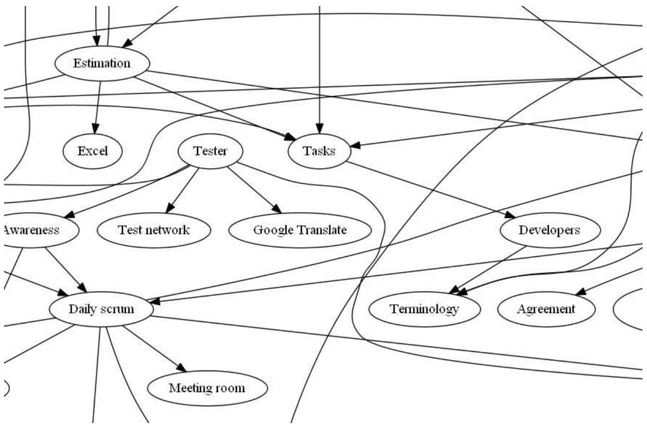


Figure 1: An excerpt of the dependency graph generated from our data material.

Despite our efforts to learn about individual or small clusters of dependencies through a dependency graph visualization, we saw how a myriad of dependencies were linked in much larger structures, and while we could pick and examine one, its meaning and effects were tightly linked to other such dependencies.

While the initial focus concerned the dependencies, as the analysis proceeded and empirical write-ups and thick descriptions were made, it became clear that the important finding of the data was not simply in the descriptions of the dependencies. Instead it became clear that the main interesting findings, which emerged in different ways and places from the data, concerned how mechanisms of standardization and routine were enacted. Therefore, in an iterative process of redefining focus and stabilizing the empirical narrative, two main contributions appeared: 1) an empirical account of the dependencies that constitute the global software development practices, 2) the role and practices by which standardization and routine are enacted to handle the complexities of the global work.

3.1 The Case

The organizational setup was an outsourcing setup between a Danish software vendor (DKSoft) working out of Denmark and L&T Infotech, a large Indian outsource specialist working out of India. While two people from L&T Infotech had been temporarily relocated to work in Denmark, most of the collaboration was mediated through technology. The IT system being developed was a financial software system for the Scandinavian market, and it had been worked on for just over a year when we visited. The global work was organized by following the Scrum methodology from the beginning of the project [30]. The decision to apply the Scrum methodology was chosen by the Danish company, as the project was large and targeted the whole Scandinavian market. Given this complexity, the Danish project management estimated that it would be impossible to define requirements up front and opted for an iterative development approach. Additionally, they had prior positive experiences following the Scrum methodology on other projects. The Indian outsourcing company had trained their staff in the execution of Scrum precisely to be able to work with Scandinavian vendors.

The Scrum team we followed was distributed across both Denmark and India. In Denmark, the Scrum master, product owner, and test lead were sitting along two Indian developers that had been temporarily relocated to Denmark. In India, the team consisted of two front-end developers, two back-end developers, and one tester. All but one of the Indian Scrum team members had previously been to Denmark to work for about half a year at the Danish site. The team followed the Scrum process with daily meetings, estimations meetings, sprint planning, demonstration meetings, and sprint retrospective. All meetings with the exception of the demonstration meeting were conducted using videoconferencing with participants from both locations.

The empirical data were collected in the period between Aug 12 2013 and Aug 23 2013 , where one researcher visited L&T Infotech in Mumbai, while another researcher visited DKSoft in Ballerup. In total, 36 hours were spent on observations, where 15 were in DKSoft and 21 in L&T Infotech. Observations included meetings, individuals working, the environment in general, as well as the workspace arrangements, etc. Twelve interviews were conducted with team members in both Denmark and India. Each interview on average lasted about 30 minutes and was later transcribed. In total, we interviewed five team members in India (four developers and one tester) and seven team members in Denmark (the project manager, product owner, test manager, Scrum master, a colleague in charge of the infrastructure, and two temporarily onshore Indian developers).

All the data material was collected and shared across the authors and discussed both during data collection (using Skype) as well as after through the iterative writing process, creating diverse, thick descriptions and cut-down narratives based on the data material.

4. DEPENDENCIES IN A GLOBAL SPRINT

While presenting our empirical data, we will be focusing on the diverse sets of dependencies that we observed during the two-week period of close collaboration between the two sites. We ended up structuring our main findings within four different perspectives, which became evident as our data analysis progressed: the locations, the people, the collaborative activities, and the artifacts. What makes each of these perspectives interesting for our analysis is that they each form a dependency set that is constitutive for the global software development practice. We label these perspectives as "dependencies" as they each, in different ways, interlink the practices while providing certain conditions for how the collaboration could unfold. Dependency concerns a coupling or relation between multiple entities, where change in one entity produces ripple effects in other entities. Coupling and reliance do not appear only in between people; certain other entities, such as the location where one works, create particular conditions for the collaboration, and as such a dependency in the work. It does matter whether the developer who is normally located in Mumbai is temporarily moved to work in Denmark. It creates certain conditions for how the work can be executed in practice. Following this perspective, we identified four different types of dependencies that shaped the work in the empirical case. Furthermore, we experienced instances where exceptions occurred, and finally, we observed how dependencies existed as interlinked entities. Below we zoom in on each of these. For each subsection, we bring the empirical account across the geographical sites we investigated.

4.1 Locations Dependencies

When we first visited the L&T Infotech offices we encountered a large, modern office building, which provided a quite different experience than the outside traffic, noise, and smells of the Mumbai city and its large highways running nearby. The building that housed the L&T Infotech offices was located within a large IT park with several other large, brand-new buildings. In the seven-story building, L&T Infotech occupied two floors. Several departments, some open and some closed off by frosted glass walls, were located side-by-side. They were only accessible after passing through a front gate guarded by three security guards. In all the departments, signs hanging from the ceiling indicated which client they were working for.

The department we were visiting was located at the centre of the floor, closed off by two access-card-operated doors. The security was quite high since DKSoft, the company they work with, develops financial software. The department hosted around 35 people, all working on projects for DKSoft. The people were divided into approximately six teams, supervised by two project managers and one delivery manager. The team we observed consisted of five software developers. Due to the time difference between Denmark and India (3.5 hours), the team usually met in between 10:30 and 11:00 AM to have more overlapping working hours with the Danish team. The department had been decorated to match the Danish company's visual identity; paper flags with the name of the company were scattered throughout the workspace, and the values and mission statement of the Danish company had been printed on large posters that were placed at the entrance to the department. These statements, however, had been modified from the original company's values and missions to fit with the Indian company. On the wall, a Danish and an Indian clock hung next to each other, but both had stopped working.

Around the same time in Denmark, we also entered the DKSoft office building, located in a suburb outside of Copenhagen together with several other IT companies. In front of the parking space of the office were three flagpoles with flags displaying DKSoft's logo. The office sign showed the same logo of DKSoft as well as two smaller logos of subsidiary companies. The three-story office building seemingly hosted many employees – not unexpected for one of the biggest IT solution providers in the banking sector. On the way to the office landscape where the team we observed worked, turnstiles prevented non-employees from passing. In order to enter, each day a temporary pass was handed to the observer at DKSoft. Additionally, any visitor had to be accompanied by one of the local employees while going through the turnstiles. This could not be one of the temporary onshore Indian developers.

The seven members of the project team located in Denmark were seated at two neighbouring islands of desks with a walkway in between, separated only by low dividers. Within the same open office space, about six more of these "desk islands" were located. Two temporarily relocated Indian developers were sitting on one side, while the Danish developers sat together with the product owner and project manager at the other island. The product owner later explained that this setup grew organically as the Indian developers were usually there for shorter periods of time. In contrast to the offices at L&T Infotech, there was no decoration revealing that part of the development team was working at L&T Infotech, except for an Indian gift from one of the Indian developers who temporarily worked onshore. Some stickers on a small cabinet revealed which project the team was working on.

We view the locations and the structure of the locations as forms of dependencies, since each geographical site produces certain conditions for how the work can be executed. For example, when the office in India is located in the heavy traffic in Mumbai, it produces particular conditions for the time it takes to move around in the city, and what time the developers can be at work in the morning and leave in the evening. Travelling in heavy traffic in a large metropolis in India is not an easy task, which means that developers either follow a schedule of company transportation or come in late and thus also stay late to accommodate the traffic. The software developer is dependent on the traffic in terms of making it to work. Turning to the Danish location, it is clear that the traffic is not an issue in the same way. The differences in travelling in traffic at the two different locations thus place particular conditions not only on the people having to make the travel, but also on the collaboration as a whole, as meetings and communication have to be planned around the times at which people can be present in the office. At the Danish offices, however, the office layout creates certain conditions for the work. Being temporarily collocated in the same office, it is possible for the visiting developers to engage with the DKSoft developers. However, a certain distance between the regular and the temporary developers still exists by having them dispersed across different desk islands.

Location dependencies include diverse sets of aspects such as security and office design, and our point here is that it does matter where people's physical bodies are located in terms of what makes the conditions for the global work. Working remotely does not limit the impact of where we are and how this shapes the kind of collaborative engagement we can commit to.

4.2 People Dependencies

The team working on the project we observed was dispersed across two locations, seven members at DKSoft in Denmark and five members at L&T Infotech in India. In Denmark, the seven members took on the role of Scrum master, product owner (one primary and one assistant), team lead, test manager, and developer (one lead and one general). All were permanently located at DKSoft, except the two Indian developers who were temporarily relocated there for six months. In India, the members took on the role of user interface (UI) developer (two members), backend developer (two members), and one tester. Although the main organizational activities were taken care of in Denmark by the Scrum master and product owner, development on the project occurred from both sites as a collaborative process. This team thus provided an excellent case to study dependencies in closely coupled work across geographical distances.

The whole team relied heavily on the knowledge about the client and the system they were developing, which was organized by the product owner in the team. The product owner represents the client stakeholder – in this case a large Scandinavian financial company – and holds regular meetings with this client. He is thus the close link between the development team and the client. The product owner was responsible for the product backlog, and so the development process depended greatly on his decisions.

The developers also depended on the product owner for the development process. With the product owner being the client stakeholder, he was the person in the team who knew most about these business processes. The developers at L&T Infotech thus needed him for explanation of these different processes.

During the course of the project, DKSoft intended to have all developers from L&T Infotech come to Denmark for a 6-month period. This was handled by having two developers at a time in Denmark. During our observations, these two developers were the lead developer and a UI developer. The motivation for this was clear; if the L&T Infotech developers had been to DKSoft, they would get to know much more about the team in Denmark and how work was organized. When returning to India to work, these developers would not only have a better understanding of how people worked in Denmark, but they would also have developed more personal relationships with the Danish team – giving a more coherent feeling of "team spirit" despite the fact that the team was based in two different companies.

While the relocation of the lead developer from L&T Infotech was temporary, the Scrum master told us that he preferred to keep the lead developer in Denmark as long as possible because they feared that another person could not do the job as well. When asked what they would do when he left for India again, the Scrum master answered, *"We might have to tie him to the table . . . "*. With team members from both sites relying heavily on his role as boundary spanner and keeping the collaboration working, they were very dependent on his presence at the Danish site. The L&T Infotech developers depended on him for their daily development work, while the Danish team relied on him as a boundary spanner between the Indian developers and the Danish team. His role in the project thus had many sides: He was the most experienced programmer from India and thus the natural contact person for questions regarding implementation, he was responsible for hand-picking team members from the Indian outsource company to work on the project, and he maintained continuous communication with the developers in the team.

People dependencies arise in the work practices where different people have diverse sets of knowledge, both in terms of knowledge about the content, process, and the business, as well as knowledge about technologies and the system under construction. What is particularly interesting in this perspective is that part of what makes the global software development team function is dependent upon the practice by which team members are hand-picked across distance, and the special competences of boundary spanners to be able to solve such a task. Also interesting is that people dependencies are not simply one-to-one relations, but instead a multiplicity of relations across people, their roles, and work tasks. Understanding the perspective of people in terms of people dependencies thus points to interlinked practices across diverse subgroups spanning geography, roles, and responsibilities.

4.3 Collaborative Dependencies

While the two preceding sections focus on the location and people dependencies, this subsection zooms in on the collaborative dependencies of the software developers. These dependencies emerged and became observable particularly during the diverse types of meetings (collocated as well as across sites) the team engaged in.

The team we observed used Scrum as their development methodology. Scrum puts forward a number of meetings and processes that should be followed throughout a development process. We observed that some of these processes and meetings were followed in the collaboration and had eventually become routines. These routines also helped integrate new team members into the project. One of the developers from L&T Infotech who had just worked with the team for two sprints explained it as follows:

"We have a daily Scrum meeting every day and we do estimation and we know that in the QC [software requirements management tool] looking at the tasks we know the remaining estimated hours and we have to finish within that time frame. So everything is written within the system, and if you follow the system, you can finish the tasks on time." (Interview, developer, India, 21/08/2013)

During our observations, we observed 10 daily Scrum meetings, two estimation meetings for the estimation of tasks for an upcoming sprint, and the demonstration meeting towards the end of the sprint. These meetings were particularly interesting as they rendered the dependencies between the different sites quite visible.

The daily Scrum meetings were executed as short 15-minute meetings and were attended by the developers and testers from DKSoft and L&T Infotech. In most cases, the product owner also took part in the meeting. In the meetings, each participant took turns explaining the progress of work. Following the same routine, the Danish and Indian team went to their respective videoconferencing rooms at the same time each day – 9:45 AM Danish time / 1:15 PM India time. The same developers at L&T Infotech always prepared the video equipment and made the call to the Danish site.

The meetings proceeded with each present meeting participant taking turns explaining what she had been working on, what she was going to work on, and mentioning any problems that she had experienced. The turn order was not defined and was settled by whoever took initiative to start. The Danish Scrum mater was responsible for deducting hours from the tasks that had been worked on to reflect their new status. At one meeting, for example, an L&T Infotech developer explained that he had been working on the document handling user story and that the Scrum master could deduct 5 hours from that task. The Scrum master then reduced the number of remaining hours on that task from 7 to 2. With the screen shared to the Indian site, this process was visible for the whole team.

The Scrum meetings were usually finished within the 15 allocated minutes. In a few instances, the meetings went over time due to either starting late (as a consequence of change of meeting rooms at the Danish site) or due to discussions about certain issues. In these cases, another team at the Indian site would knock on the door of the meeting room and the meeting was quickly wrapped up. However, in Denmark the room booking system did not allow booking rooms for 15 minutes, so they were always booked for 30 minutes, but only used for 15.

While the daily Scrum meetings were conducted with both the Danish and the Indian parts of the team present, the demonstration meeting at the end of the sprint was deliberately held with the Danish team only. During the demonstration meeting, the developers in the Scrum team demonstrated the features they had implemented in the sprint, and the product owner would then accept or reject the feature. Such a setup, however, would require a different technological setup, as the test environment where the product being developed was running was located on a Danish internal network. Giving access to the Indian team for them to demonstrate what they had implemented would require a virtual private network (VPN) connection from India to Denmark. Instead, the developers at the Danish site handled demonstrating all the features, even those implemented in India. Due to the daily meetings and the lead developer keeping daily contact with the Indian team, the developers in Denmark were quite aware of the features implemented in India and therefore capable of demonstrating them.

One particularly interesting meeting was held each afternoon. By the end of each working day in India, around 3:30 PM Danish time / 6:45 PM India time, the lead developer located in Denmark contacted the development team in India over instant messaging to get feedback on their progress and resolve any issues they might have when meeting for work the next day. These meetings were not a part of the traditional Scrum methodology, however, and they resembled that of the daily meeting in content. The developers at L&T Infotech were given the opportunity to explain their work and discuss any potential issues. One of the Indian developers explained this process as the following:

"We don't have a Scrum board because our team lead is not sitting here... But then, our team lead communicates with us every now and then in the communicator. So we update the daily status before we go – that we have done this, we have done that – even before the daily Scrum meeting." (Interview, developer, India, 21/08/2013)

These meetings were routinely handled and had thus become a natural part of the development process. These procedures had all become routines, and they helped the Danish team gain awareness of the work in India and made sure that the people at L&T Infotech did not have unresolved questions before they went home.

Collaborative dependencies arise in the everyday work when different people are interdependent in the actual work. These dependencies become particularly visible during meetings where distributed actors are brought together. However, they also arise in the arrangement and structure of the work – in this case the iterations of the project in sprints.

4.4 Artifactual Dependencies

We have now discussed the work environment, the team, the task, and the collaborative activities. However, an important part of the closely coupled work also concerns the diverse set of artifacts that the participants use in making their collaboration work. Interestingly, the absence of particular artifacts also places certain constraints on the developers. We will look into how the participants managed to organize their work in particular ways to accommodate limitations of technology.

The team relied heavily on the technical infrastructure for their work. Most common software engineering tools were used to develop the product: integrated development environments for coding, source control management for managing code, and test environments for testing. For project management, an integrated task-tracking tool was used. SAP was used for noting working hours and the team communicated using email, telephone, instant messaging, and videoconferencing.

In order to let daily Scrum meetings take place in a global context, dedicated videoconferencing rooms were used, equipped so that setting up bidirectional communication was as effortless as possible. The conference rooms used on both sides had a meeting table with a central microphone and surrounding chairs. Mounted to the wall were two screens, one to display the video feed of the team on the other side, as well as a smaller video feed of themselves, and another to display a shared screen. The Scrum master was the only one to bring along a laptop, as it was his screen that was shared with the Indian team.

Besides the video conferencing equipment, the Application Lifecycle Management (ALM) tool by HP usually open on the Scrum master's laptop was the central artifact used during global Scrum meetings. It was used to provide an overview of and to update the state of user stories, for example, to reduce the amount of expected hours to work on certain tasks. In order to see the screen shared by the Scrum master in India, one of the Indian developers had to log into the VPN prior to the meeting to set up the screen sharing. Only the Scrum master made changes to the ALM visible on the shared screen as the other developers updated the team on their progress.

The back-end developers used business process modeling notation (BPMN) – a modeling notation used for specifying business processes – to develop the core infrastructure of the product. On an artifact level, the project used BPMN to model their back-end. BPMN is used to model business processes using a graph-like notation. The back-end developers in India modelled the business workflows of the application as graphs, and these graphs, in turn, were used to generate code for the back end.

Arguably, any modelling or programming language is a form of standardization, as it equips programmers with a structured way to construct software and a common way to talk about such implementations. However, we argue, that BPMN adds to this standardization. With the abstraction towards actual business processes that BPMN adds, we observed how it equipped the back-end developers with a language that could be used to talk with the domain knowledge experts. We thus claim that the use of BPMN provided a form of standardization highly suitable for the development of such business-logical applications.

The project followed an agile approach in which the development process was split into four-week iterations during which a fixed workload of features were selected for implementation. Within Scrum these iterations are called "sprints." The sprint we observed lasted 4 weeks and comprised 10 user stories describing certain features that needed to be implemented. Examples of user stories from the project include integrating scanning documents into the business process, providing download links to digital contracts, and implementing "fast track" processes for certain merchants. Depending on the assessed complexity of the user stories, more or less can be taken up into a sprint. The success of a sprint depends on the developers actually finishing them.

Artifactual dependencies concern situations where technology is involved, such as videoconferencing, task-tracking software, communication software, and networks. The division of work into user stories also constitutes an artifactual dependency as these were the objects dictating work in the team.

4.5 Exception Handling

The four previous sections concerning sets of location dependencies, people dependencies, collaborative dependencies, and artifact dependencies all describe situations where work goes as expected. However, there are several situations where exceptions occur, and it is critical to investigate what happens in these situations in terms of dependencies and how these are organized.

Despite the routine manner in which the daily Scrum meetings were executed, we observed different situations were exceptions occurred. The room for the daily meeting was booked for the Indian team for use from 1:15 PM to 1:30 PM each day. The Danes switched rooms according to availability. They used a room booking system to book rooms; however, in several instances, problems with this booking caused a delay in the daily meeting as the Danes had to find an available room. In addition, sometimes some of the Danes showed up on time for the meeting while others were late, as they did not check themselves beforehand which room had been booked for the meeting. Whether people showed up late or not, the meeting always started once the videoconferencing was set up. We observed this on several occasions where the Indian team was on time, as was the Danish Scrum master at the other end, but the rest of the Danish team was still missing. The meeting would be initiated and the rest of the Danish team would show up after a few minutes.

The technology used to facilitate the daily meetings also caused exceptions. Take the following observation, for example, from the preparation of a daily Scrum meeting where a virtual desktop connection needed to be made from a computer in the Indian videoconferencing room to a Danish computer.

"U1 is sick today, so U2 heads for the meeting room to start the video equipment and the projector. He's clearly not as experienced doing that so it is taking longer than when U1 is doing it. He also doesn't have a login to the VPC so he has B1 establish a virtual private network (VPN) connection after which B2 logs in with her account on the VPC." (Observation notes, India, 20/08/2013)

Usually, the same developer from L&T Infotech (U1) would prepare the videoconferencing equipment and make the call to the Danish site. When he was on holiday one day, another person (U2) had to perform this work. Setting up the video and the shared workspace took significantly longer and also forced the person to seek help with login to a virtual PC.

Interestingly, company rules and regulation regarding connectivity and firewalls created particular technical constraints for the development team, forcing them to organize their work in particular ways. As the testing environment was located in Denmark – and the team at L&T Infotech did not have access to this environment over the network – they were forced to run a similar setup on their own computers. This setup consisted of a virtual machine running Linux and the web-server software that the product was developed for. The setup was so heavy on computer power that the team – for this project – had been given new, powerful computers, as opposed to other teams in the company working on other projects. Despite this investment, the developers at L&T Infotech had to reboot this test setup on a daily basis. This process took between 10 and 15 minutes and caused work to halt for that duration. The developers would usually go for a coffee if this happened.

Another example of exception handling became clear when the Scrum master explained the team's testing procedure. Interestingly, the testing and fixing of bugs in the product was done one sprint behind. The Scrum master explained it as follows:

"Yeah, of course optimally we would like to have everything well tested before the end of the sprint so that at least very few defects are found in the next sprint. But we haven't really, that's our goal, but we haven't really succeeded because the implementation just is rarely finished before very close before the end of the sprint. So then there is not enough time to test." (Interview, Scrum master, Denmark, 23/08/2013)

Despite their intention to deliver a working and well-tested version of the product after each sprint, as the implementation was not finished until the very end of the sprint, the testing was left for the next sprint. This process caused some problems, as errors that emerged were to be resolved quickly because the product of the sprint, when it was introduced, had already been marked as done. In one specific case, the two developers at L&T Infotech stayed until midnight to fix an error that was introduced in a previous sprint.

4.6 Sets of Dependencies

We have now presented four sets of dependencies that created certain conditions for how the global software development collaboration was executed and organized. In addition, we looked into the practices by which the developers handled exceptions and unforeseen challenges. The four sets of dependencies were location dependencies, people dependencies, collaborative dependencies, and artifactual dependencies. While this overall categorization of dependencies helps us understand what organizational dependencies in global software development comprise, one important empirical observation remains, namely the interlinked nature of dependencies.

Analyzing our data iteratively, and creating diverse sets of rich descriptions, we realized that even though we were able to label specific dependencies, several categorizations could be applied to a dependency. Therefore, we decided to think in terms of "sets of dependencies" rather than dependencies as singular causal relations. Furthermore, we saw how one set of dependencies (e.g. people dependencies) was tightly linked to others (e.g. collaborative dependencies). This leads us to suggest that dependencies in global software development have an interlinked structure, where it is not easy to pick apart and set boundaries for what is part of and what is outside of the particular dependency, for example, what is technical and what is social. We might, for example, talk about the development process of a particular artifact, such as a user story, and how it has multiple dependencies embedded in the very task: coupling the IT system under development, the technical architecture, the work across developers, and other user stories. But at the same time each user story also has close connections with other artifacts – for example, the range of documents that makes the development project, such as requirement specification or test documents [8]. To handle all these dependencies, a range of tools such as development environments, test setups, and communication tools are being included in the collaborative activities. However, what is more surprising is how participants' hardware and technical infrastructure, which is related to the developer's location dependencies as well as the artifactual dependencies, place particular conditions for how the participants can manage and handle their work, for example, as we saw in the need for powerful computers and the technical setup for demonstration meetings, which placed the participants in a situation where developers located in India could not fully participate. How the technical infrastructure and the hardware devices impact the conditions for collaboration in this particular case was due to the location dependencies in the diverse structure across the different geographical sites. The technical infrastructure locally thus created particular conditions for how the participants could manage to handle and organize their work according to the set of location and artifactual dependencies they need to accommodate.

Similarly, we may find that one particular set of dependencies gives rise other sets of dependencies. As such, the structure across the sets of dependencies resembles that of the documentscape [8]. The documentscape refers to the dynamic interlinked ensemble of project documents in global software development practices between developers located in various locations. It stipulates that while we might zoom in on one single document, the meaning of the document is embedded within the location of the document within the documentscape. We may also look at a single dependency; however, its meaning is embedded within the network of dependencies across all sets of dependencies. While we may look at a dependency and label it according to different sets of dependencies, each dependency plays a role in a larger set of interlinked dependencies where the boundaries are not easily defined.

Category	**Features**
Location Dependencies	External environment; office design; office security; time zone
People Dependencies	Roles; responsibilities; knowledge
Collaborative Dependencies	Meetings; turn taking; discussions
Artifactual Dependencies	Business process modelling notation; VPN; video conferencing setups; test environment; user stories

Table 1: Categorization of dependencies

Despite the interlinked nature of dependencies, we may still talk in terms of sets of dependencies. As our analytical cut-downs in the data material demonstrate, we found four important sets of dependencies: location dependencies, people dependencies, collaborative dependencies, and artifactual dependencies. Location dependencies entail the multiplicity of dependencies that are related to the geographical site of the software developers. This includes the external society within which the location is placed, as well as the inner office environment design and the conditions this creates for the collaboration. People dependencies entail the roles, responsibility, and specific knowledge the collaborative actors have and use in the work together. This includes knowledge about competences, knowledge about technologies, as well as knowledge about the business in which the system is being created. Collaborative dependencies in particular become observable in situations where multiple people engage in a common practice – in most cases in meetings, but also outside of meetings. The meetings can take many different forms and shapes and be organized differently. Some meetings involve participants from both locations (e.g. daily stand-up meetings), while others are location specific (e.g. demonstration meeting). Artifactual dependencies are pertinent at all times, since technology is what makes it possible for the dispersed developers to collaborate and interact with the same coding environment, which serves as the main technical infrastructure supporting the work. However, as we have also shown, the artifactual dependencies also comprise other important artifacts such as the user stories, the BPMN notation, the VPN connection, and the video equipment, which all took part in linking and organizing the work and connections between the developers. It is important to notice that in practice these sets of dependencies are not singular entities, but instead function as multiplicities that connect and interrelate across all types of dependencies. Table 1 summarizes the four sets of dependencies and their characteristics.

5. STANDARDIZATION & ROUTINE

Having conceptualized the four main categories of dependencies that emerged in our data, the next question concerns how the developers managed to collaborate despite the diverse set of dependencies and how these each brought about an increasingly complex work arrangement. Looking across the categories of dependencies, we found a repeating pattern in the work regarding the participants' ability to continuously create and apply standardization as a mechanism to reduce complexity. There were standardizations in terms of meeting structure and time for conducting meetings. There were standardizations in terms of the team at L&T Infotech arriving late in the morning to accommodate the time difference between the sites. There were standardizations in terms of people's roles and knowledge, and to some extent there were standardizations in terms of artifacts such as user stories or the use of the BPMN notation form.

What is interesting about how the strategy of standardization was applied by the practitioners in the empirical case was that it did not arrive as a top-down forced structure on the work. Instead, it emerged over time as the practitioners made the methodology of Scrum into a practice fitting their work. It was not that the developers did not follow the methodology; our point here is that they had aligned the methodology with their practices in the global work. In this way, the standardization of when, where, and how to meet supported the developers in organizing their work, taking into account the different sets of dependencies, but at the same time the standardization had a transformative effect on the practice [13], as it was dynamically developed. Engaging with the standardized process of Scrum was not a process by which the developers simply followed Scrum in a scripted manner. Instead, they adjusted and recreated the process, making it fit their practices while still taking into account the conditions for collaboration created by the location and artifactual dependencies. So when the Indian developers could not participate in the demonstration meeting due to the lack of a standardized technical setup across sites – the artifactual dependencies not letting them participate – they adjusted the work accordingly and found other ways to demonstrate the requirements developed at L&T Infotech. In this way, the standardization was a transformative practice where processes and technologies were being accommodated, and thus standardization was incomplete initially. What is also fascinating with this example is that while we expected that standardization across the technical platforms would be easy to create (e.g. having access to the same tools and applications etc.), this turned out to be the most challenging area, where local artifactual dependencies place certain constraining conditions on the collaboration.

We observed how the daily meetings were executed in a routine manner, giving rise to frequent interaction across sites. However, whereas the team at L&T Infotech always had the same meeting room booked for their meetings, the team at DKSoft had a different meeting room with videoconferencing equipment booked on a day-to-day basis. This caused several situations where people from DKSoft arrived a bit late since they first arrived at the wrong meeting room prior to finding out the designated meeting room for that day – in other words, the complexities of relation work increased [2, 9]. This lack of standardization in terms of rooms for daily meetings thus created extra challenges – not only for the developers at the Danish location, but also for the developers at the location in Mumbai, since they had to wait for the videoconferencing to be started from the remote location. In contrast, the room in Mumbai never changed, and as such the location dependencies did create different types of conditions for the work, both locally and in the distributed situation. However, not only did the location constraints affect the daily meetings, we also observed an exception-handling situation where the developer who was normally in charge of setting up the VPN connection was not present. Here it turned out that the developer who wanted to set up the equipment instead did not have a VPN login, and extra work was required to locate a person with a VPN login before the daily meeting could take place. The complexities of managing the dependencies – in this case the amount of relation work needed to establish a connection between the two sites – increased.

What also becomes obvious in the example above is the role of routine [14]. For each activity the developers engage in, both locally and in between the two sites, the more frequent the activity, the more aligned and transformed their common practice becomes. Routine is the practice by which the developers turn standardization strategies into concrete activities that support their practice. It is through the enactment of routine behavior in daily interaction that the developers are able to reduce the effort required for articulation work and to spend more time on the actual project they are developing. We observed routines as repeating patterns of coordination employed to handle the complexities of the diverse set of dependencies. Routines, as opposed to strategies, exist only through the actual enactment and the different activities [14]. Referring to the concepts of plans and situated actions [34], standardization constitutes the plan whereas the routine is embedded within the situated action. In our case it was clear that the routinized behaviors were mostly connected to the various types of meeting activities in the distributed team, as in the example where the developers explain why they have an additional meeting at the Danish location to ensure that the team at the Indian site is kept up to date. They explained how this meeting was a replacement of the Scrum board, and how this practice supported their work.

Other artifactual dependencies gave rise to exception handling. The lack of standardization regarding access to the test network required the developers at L&T Infotech to be equipped with much more powerful computers than the rest of the employees. The restrictions caused by firewalls and the generally slow bandwidth out from India forced the developers to have a local version of the testing environment running, which was only possible on these more powerful computers. While it was not possible to standardize the technical connectivity across the geographical sites due to firewall, bandwidth, and privacy issues, the developers created other routine behaviors in terms of testing practice, making it possible for them to collaborate despite the technical constraints.

Standardization and routine did not solve all issues of dependencies. Our data clearly demonstrated that a repeating pattern across sprints was the testing that was pushed back to the next sprint, often causing the developers at L&T Infotech to work overtime. Here it is important to notice that working late at the Mumbai location means evening and late evening, and since this occurred quite often it clearly impacted the work at L&T Infotech. The reason for the delay was that the developers were implementing source code until the very last minute, not leaving time for the testers to reach their goal within the pre-allocated time of the sprint. The developers depend on the tester to report defects. However, the tester cannot begin until the developers have delivered the source code. The product owner in the end depends on the developers to fix the defects detected by the tester. All of this is organized through the bug report artifact. This made the activity of testing a closely coupled activity requiring closely coupled coordination. No routine or standardization was able to solve this activity, which meant that major defects in the software sometimes occurred in subsequent sprints, forcing the team to work even later, solving tasks from the last sprint while neglecting their tasks for the current sprint.

In summary, in many cases the practitioners applied standardization as a strategy to handle the complexity evolving from heterogeneous locations, people, and artifacts, and in most cases it worked well. However, there were also situations where standardization did not work – and surprisingly this was largely in the technical infrastructure and hardware across the sites. This lack of standardization did increase the complexity of including participants equally in, for example, the demonstration meeting, and also in terms of having enough machine power to execute the work. It was clear that standardization made it possible for the participants to reduce their effort required to organize their collaboration, making it possible to focus on the task ahead. However, at the same time the extra standardization also required work – articulation work – and as such the standardization strategy was not “free”. Standardization is a known strategy to handle coordination [16]; however, what was interesting in our study was how the standardization process made it possible for the participants to handle and deal with many complexities that emerged from the diverse set of dependencies. On the other hand, issues of solving the closely coupled task of testing remained. Similar to how specific artifacts reduce the complexities of articulation work [29], knowing the standardization practices in a routine manner made it possible for the participants to reduce time spent on articulation work and instead made them focus on the content of the development project.

6. CONCLUSION

In the very definition of collaboration exists the notion of dependencies in work. However, dependencies as pertinent in collaborative work might take different forms and shapes. In this paper, we reported on an ethnographic study of global software work executed across two sites – Denmark and India – zooming in on what makes the pertinent dependencies in such work. We observed how various types of dependencies constitutive of the collaborative practice across the developers included location dependencies, people dependencies, collaborative dependencies, and artifactual dependencies. In addition, we saw how the strategies of standardization form a repetitive pattern in how the dependencies were managed across sites, and in particular how routine in the work was crucial for making the collaboration function. Based on our observations, this paper puts forward three main contributions: *(i)* Global software development is based on a multiplicity of dependencies that fall into four broad categories: location, people, collaboration, and artifactual. Using these categories, we can start to examine more carefully how dependencies constitute global work, and in particular point to situations where dependencies are challenging for the collaboration. *(ii)* Dependencies in global software development are not singular entities, but instead exist as one multiplicity of interlinked practices, which makes it difficult to clearly define and distinguish the borders between the dependencies. *(iii)* Standardization and routine can be used as specific coordination mechanisms to handle dependencies, thus reducing the complexity of the associated articulation work. However, surprisingly, our data suggest that it is more difficult to standardize the artifactual dependencies (e.g. hardware and software practices) across geographical sites, compared to standardizing the organizational practices such as user stories, roles and responsibility, or daily meetings.

7. ACKNOWLEDGEMENTS

This research was conducted in collaboration with L&T Infotech, both in terms of funding as well as in terms of access to the empirical sites. In particular, we would like to mention the efforts and work of Sarbajit Deb (Vice President, L&T Infotech, the Nordic Region) and Muthuramalingam P. (Deputy Head, Banking & Financial Services, L&T Infotech, Denmark) in making this research possible. Without such strong connections between industry and academia, we would never have been able to conduct such ethnographic research.

Furthermore, we would like to thank the employees at DKSoft for welcoming us into their company, working with us, and allowing us to follow them during their work. Lastly, we thank Steven Jeuris for his participation and help in the study.

This research has been funded by the Danish Agency for Science, Technology and Innovation under the project “Next Generation Technology for Global Software Development”, #10-092313.

References

[1] P. Bjørn and E. Balka. Health care categories have politics too: Unpacking the managerial agendas of electronic triage systems. In *ECSCW 2007*, pages 371–390. Springer London, 2007.

[2] P. Bjørn and L. R. Christensen. Relation work: Creating socio-technical connections in global engineering. In *ECSCW 2011: Proceedings of the 12th European Conference on Computer Supported Cooperative Work, 24-28 September 2011, Aarhus Denmark*, pages 133–152. Springer London, 2011.

[3] A. Boden, G. Avram, L. Bannon, and V. Wulf. Knowledge management in distributed software development teams - does culture matter? In *Global Software Engineering, 2009. ICGSE 2009. Fourth IEEE International Conference on*, pages 18–27, July 2009.

[4] A. Boden, B. Nett, and V. Wulf. Articulation work in small-scale offshore software development projects. In *Proceedings of the 2008 International Workshop on Cooperative and Human Aspects of Software Engineering*, CHASE ’08, pages 21–24, New York, NY, USA, 2008. ACM.

[5] A. Boden, F. Rosswog, G. Stevens, and V. Wulf. Articulation spaces: Bridging the gap between formal and informal coordination. In *Proceedings of the 17th*

ACM Conference on Computer Supported Cooperative Work &#38; Social Computing, CSCW '14, pages 1120–1130, New York, NY, USA, 2014. ACM.

[6] G. C. Bowker and S. L. Star. *Sorting things out : classification and its consequences*. MIT Press, Cambridge, Mass., 1999.

[7] M. Cataldo, M. Bass, J. Herbsleb, and L. Bass. On coordination mechanisms in global software development. In *Global Software Engineering, 2007. ICGSE 2007. Second IEEE International Conference on*, pages 71–80, Aug 2007.

[8] L. R. Christensen and P. Bjørn. Documentscape: Intertextuality, sequentiality & autonomy at work. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '14, New York, NY, USA, 2014. ACM.

[9] L. R. Christensen, R. E. Jensen, and P. Bjørn. Creating relation work: Characteristics for local and gloval collaboration. COOP '14. Springer, 2014.

[10] C. de Souza and D. Redmiles. The awareness network, to whom should i display my actions? and, whose actions should i monitor? *Software Engineering, IEEE Transactions on*, 37(3):325–340, May 2011.

[11] C. R. de Souza, S. Quirk, E. Trainer, and D. F. Redmiles. Supporting collaborative software development through the visualization of socio-technical dependencies. In *Proceedings of the 2007 international ACM conference on Supporting group work*, pages 147–156. ACM, 2007.

[12] K. M. Eisenhardt. Building theories from case study research. *The Academy of Management Review*, 14(4):pp. 532–550, 1989.

[13] G. Ellingsen, E. Monteiro, and G. Munkvold. Standardization of work: Co-constructed practice. *The Information Society*, 23(5):309–326, Oct. 2007.

[14] M. S. Feldman and W. J. Orlikowski. Theorizing practice and practicing theory. *Organization Science*, 22(5):1240–1253, 2011.

[15] M. S. Feldman and A. Rafaeli. Organizational routines as sources of connections and understandings. *Journal of Management Studies*, 39(3):309–331, 2002.

[16] E. Gerson. Reach, bracket, and the limits of rationalized coordination: Some challenges for cscw. In *Resources, Co-Evolution and Artifacts*, Computer Supported Cooperative Work, pages 193–220. Springer London, 2008.

[17] B. G. Glaser and A. L. Strauss. *The Discovery of Grounded Theory: Strategies for Qualitative Research*. Aldine de Gruyter, New York, NY, 1967.

[18] R. Grinter. Recomposition: Coordinating a web of software dependencies. *Computer Supported Cooperative Work (CSCW)*, 12(3):297–327, 2003.

[19] J. D. Herbsleb, A. Mockus, T. A. Finholt, and R. E. Grinter. Distance, dependencies, and delay in a global collaboration. In *Proceedings of the 2000 ACM Conference on Computer Supported Cooperative Work*, CSCW '00, pages 319–328, New York, NY, USA, 2000. ACM.

[20] M. Hertzum and J. Pries-Heje. *Is Minimizing Interaction a Solution to Cultural and Maturity Inequality in Offshore Outsourcing?*, pages 77–97. TAPIR Akademisk Forlag, 2011. 2011; 4.

[21] N. Holten Møller and P. Dourish. Coordination by avoidance: Bringing things together and keeping them apart across hospital departments. In *Proceedings of the 16th ACM International Conference on Supporting Group Work*, GROUP '10, pages 65–74, New York, NY, USA, 2010. ACM.

[22] R. E. Jensen and P. Bjørn. Divergence and convergence in global software development: Cultural complexities as social worlds. In *From Research to Practice in the Design of Cooperative Systems: Results and Open Challenges*, pages 123–136. Springer London, 2012.

[23] P. Luff and e. al. *Workplace Studies: Recovering Work Practices and Informing System Design*. Cambridge University Press, 2000.

[24] T. W. Malone and K. Crowston. The interdisciplinary study of coordination. *ACM Comput. Surv.*, 26(1):87–119, Mar. 1994.

[25] S. Matthiessen, P. Bjørn, and L. M. Petersen. "figuring out how to code with the hands of others": Recognizing cultural blind spots in global software development. In *Proceedings The 17th ACM Conference on Computer Supported Cooperative Work*, CSCW '14, New York, NY, USA, 2014. ACM.

[26] T. Meum, E. Monteiro, and G. Ellingsen. The pendulum of standardization. In *ECSCW 2011: Proceedings of the 12th European Conference on Computer Supported Cooperative Work, 24-28 September 2011, Aarhus Denmark*, pages 101–120. Springer London, 2011.

[27] R. Prikladnicki, A. Boden, G. Avram, C. Souza, and V. Wulf. Data collection in global software engineering research: learning from past experience. *Empirical Software Engineering*, pages 1–35, 2013.

[28] K. Schmidt and L. Bannon. Taking cscw seriously. *Computer Supported Cooperative Work (CSCW)*, 1(1-2):7–40, 1992.

[29] K. Schmidt and C. Simone. Coordination mechanisms: Towards a conceptual foundation of cscw systems design. *Computer Supported Cooperative Work (CSCW)*, 5(2-3):155–200, 1996.

[30] K. Schwaber and J. Sutherland. The scrum guide: The definitive guide to scrum: The rules of the game, 2011.

[31] A.-M. Søderberg, S. Krishna, and P. Bjørn. Global software development: Commitment, trust and cultural sensitivity in strategic partnerships. *Journal of International Management*, 19(4):347 – 361, 2013. Developing Offshoring Capabilities for the Contemporary Offshoring Organization.

[32] A. Strauss. The articulation of project work: An organizational process. *Sociological Quarterly*, 29(2):163–178, 1988.

[33] L. Suchman. Do categories have politics? the language/action perspective reconsidered. In *Proceedings of the Third Conference on European Conference on Computer-Supported Cooperative Work*, ECSCW'93, pages 1–14, Norwell, MA, USA, 1993. Kluwer Academic Publishers.

[34] L. A. Suchman. *Plans and Situated Actions: The Problem of Human-machine Communication*. Cambridge University Press, New York, NY, USA, 1987.

[35] G. Walsham. Icts and global working in a non-flat world. In *Information Technology in the Service Economy: Challenges and Possibilities for the 21st Century*, volume 267, pages 13–25. Springer US, 2008.

SideBar

Title of Paper

SideBar: Videoconferencing System Supporting Social Engagement

Authors:

Morten Esbensen, *Paolo Tell, Jakob E. Bardram*

Published:

In Collaborative Computing: Networking, Applications and Worksharing ***(Collaborate-Com)****, 2014 International Conference on. IEEE, 2014. p. 358-367.*

Abstract:

Companies are increasingly organizing work in globally distributed teams. A core challenge to these distributed teams is, however, to maintain social relationships due to limited opportunities and tools for social engagement. In this paper we present SIDEBAR: a videoconferencing system that enhances virtual meetings by enabling social engagement. Through image analysis of the conference video feed, SIDEBAR tracks meeting participants in real-time. A personal tablet then allows each participant to identify and track other participants, to look up information about them and their local work context, and to engage in peer-to-peer chat conversations. We describe the motivation, design and implementation of SIDEBAR and report results from a preliminary evaluation, which shows that participants found SIDEBAR useful and easy to use. The paper concludes by providing three design guidelines for collaborative technologies supporting social engagement.

SideBar: Videoconferencing System Supporting Social Engagement

Morten Esbensen, Paolo Tell, Jakob E. Bardram
Pervasive Interaction Technology Laboratory
IT University of Copenhagen
Rued Langgaardsvej 7, 2300 Copenhagen S, Denmark
Email: {mortenq,pate,bardram}@itu.dk

***Abstract*—Companies are increasingly organizing work in globally distributed teams. A core challenge to these distributed teams is, however, to maintain social relationships due to limited opportunities and tools for social engagement. In this paper we present SIDEBAR: a videoconferencing system that enhances virtual meetings by enabling social engagement. Through image analysis of the conference video feed, SIDEBAR tracks meeting participants in real-time. A personal tablet then allows each participant to identify and track other participants, to look up information about them and their local work context, and to engage in peer-to-peer chat conversations. We describe the motivation, design and implementation of SIDEBAR and report results from a preliminary evaluation, which shows that participants found SIDEBAR useful and easy to use. The paper concludes by providing three design guidelines for collaborative technologies supporting social engagement.**

I. INTRODUCTION

Companies are progressively organizing work in globally distributed teams [1]. It has been recognized that such distributed collaboration is affected by new challenges, which, according to Herbsleb [2], can be ascribed to the absence or disruption of those mechanisms that in collocated settings naturally support coordination among practitioners. Examples of these mechanisms range from spontaneous engagement in conversation to visual clues conveying awareness capable of offering insights on, for instance, availability of team members as well as their current activity. Studies reflecting on the challenges of distributed collaboration (e.g., [3], [4]) agree on the view that physical distance is a factor of paramount importance that profoundly influences how people interact [5].

Social relationships and engagement are in particular negatively affected by geographical distance [6], [7]. Given for granted in collocated settings and often overlooked and underestimated in distributed arrangements, social engagements represent powerful facilitators capable of fostering successful collaboration [8], which have to be nurtured and encouraged not only by practices but also by technologies. Important aspects of social engagement, which have been investigated and for which direct dependences have been observed are: the feeling of *group cohesion*, which has been argued to be a facilitator for the effectiveness of communication technologies [9]; the feeling of *connectedness*, for which an impact on communication has been identified [10]; the *awareness* of remote team members and of their locations, which have been linked to communication patterns [11]; and, *trust*, which has been investigated from multiple angles and perspectives as systematically reported in [12].

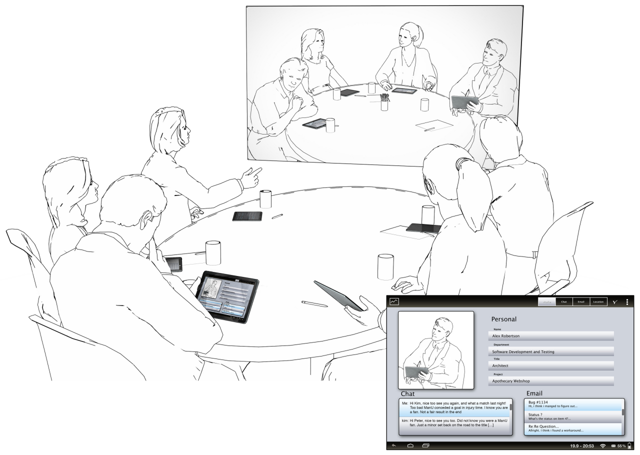


Fig. 1. SIDEBAR is a videoconferencing system supporting social engagement. This is achieved by face tracking in the video feed and by provide an interactive mirrored video feed on tablet computers, allowing users to recognize and seek information about each other, and engage in backchannel peer-to-peer chat conversations.

A core challenge in distributed collaboration is, therefore, how to support such social engagements. Collocated environments provide good conditions for social engagements to occur through shared physical spaces allowing for mutual awareness, close collaboration, shared experiences, and social encounters [5]. In distributed arrangements, however, actors have to commit to much more explicit work to cultivate such social aspects [13], especially when the social connections required to initiate these aspects of collaboration need to be established (e.g., visit to remote site and team building activities [14]).

To address the challenge of providing social engagements in distributed arrangements, we have designed SIDEBAR: a videoconferencing system with special support for social engagements. SIDEBAR incorporates face tracking into a videoconferencing system to provide an interactive mirrored video feed on tablet computers that allows users to recognize and seek information about each other and engage in backchannel peer-to-peer conversations.

In this paper, we provide an overview of videoconferencing solutions and then describe the SIDEBAR system. The more significant design decisions are discussed before detailing the architecture of the system. A preliminary evaluation of the system is also presented, which aimed at assessing the perceived usefulness and ease of use of SIDEBAR, as well as whether or not the system was perceived as a distraction. Finally, before concluding, we propose a set of guidelines for designing collaborative technologies with support for social engagement.

II. BACKGROUND AND RELATED WORK

Social connections are core to collaboration [8] but challenged in distributed arrangements. For example, Herbsleb and Mockus [7] have shown that the size of social networks in distributed arrangements is smaller as compared to collocated ones, and that the ability to recognize remote team members is more difficult for distributed team members. Similarly, it has been found that people tend to form groups with collocated colleagues rather than distant ones [15]. Social factors, however, are important in distributed arrangements in which the communication is affected by the feeling of connectedness [10] and of group cohesion [9]. Social connections and engagement, thus, are of much importance when designing tools supporting distributed collaboration.

Videoconferencing meetings are used extensively in distributed collaboration and are, in terms of media richness theory, the closest we come to collocated meetings and collaboration [16]. However, despite the richness of the video medium, a meeting supported by technologies still does not compare to a collocated face-to-face one [17]. Therefore, a significant body of research has focused on bringing the feeling of 'sitting together' to the video meeting by improving the videoconferencing technology to include information intrinsic to the collocated meeting such as eye-contact [18] and non-verbal cues [19]. Such information can be valuable in distributed collaboration by increasing the social awareness between team members [20].

The setup of videoconferencing equipment has been shown to affect different aspects of participants perception of each other. Using a video setup that captures both face and upper body have been shown to have a positive result on trust and empathy in groups as opposed to only capturing the face [21]. Another study of trust in different group-to-group videoconferencing setups suggests that a combination of personal displays and individual streams of each participant contribute to a higher level of trust development [22], and that the perception of proximity in videoconferencing is linked to the zoom of the camera [23].

GAZE-2 [18] is a group video system that supports eye-contact transmission. Using several cameras and an eye-tracker per setup, the system ensures parallax-free transmission of eye contact, by choosing the camera the user is looking at. eyeView [24] is a videoconferencing system that leverages gaze direction. Using eye tracking to resize individual video windows based on looking behavior, eyeView keeps focus on the current speaker while keeping an overview of all participants though scaled down videos of them. These systems are examples of technologies, which include eye-contact [25] or 3D-experience [26] in traditional videoconferencing setups.

Other approaches to enhance videoconferencing have focused on creating a more immersive experience. MAJIC [27], for example, uses large curved, semi-transparent displays with cameras placed behind them. This setup allows for life-size video projection with eye-contact support. TeleHuman [19] uses a cylindrical 3D screen to display life-size video of a person, increasing users sense of social presence and improving the ability to asses gaze and body language cues. LiveMask [28] uses a video setup that presents a movable 3D screen of a persons face upon which live video is projected leading to a more correctly transmitted gaze direction than traditional video setup. More generally, moving and zooming the videoconference camera to suggest movement of participants have been shown to enhance social telepresence of remote actors [29].

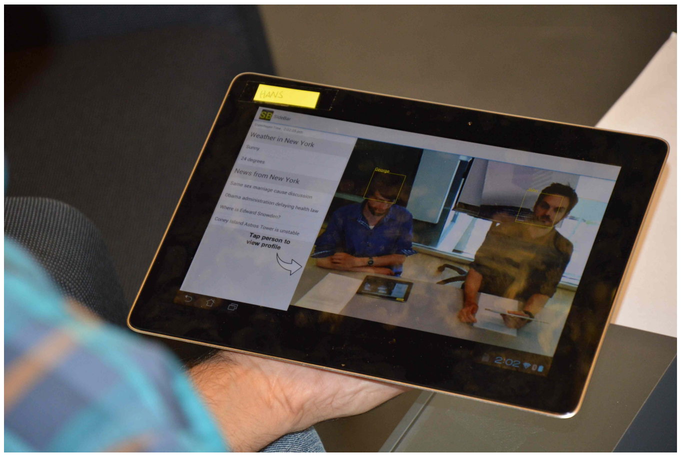


Fig. 2. SIDEBAR extends the video conferencing setup with tablet computers for all meeting participants.

In summary, prior research in videoconferencing technologies has focused on enhancing the video meeting by incorporating aspects from collocated meetings, including non-verbal cues and physical presence of remote participants information that can lead to an increase in social awareness [20]. Focus has been on improving the fidelity of the audio-video channel as such, and less on using other channels during videoconferencing to build social relationships and engagement. The latter is the objective of our research and this is approached by integrating face-tracking into the video feed of a regular videoconferencing, and use this to identify and engage with meeting participants, and support ad-hoc backchannel conversations during meetings.

III. SIDEBAR

Social aspects among people have a large impact on communication [10] and collaboration [9]. SIDEBAR is a videoconferencing system that focuses on supporting social engagement. As illustrated in Figure 2, the SIDEBAR system extends a traditional videoconferencing system with a tablet computer for each meeting participant.

The purpose of these personal tablets is to provide meeting participants with a tool to recognize remote participants and to engage in backchannel—or sidebar—task-related work, information seeking, and communication. Using face tracking techniques, SIDEBAR provides an interactive mirrored video feed (Figure 3-(1)). Meeting participants can access information about each other via personal profiles (Figure 4-(1)), can learn about the different geographical locations they are located in through location profiles (Figure 5), and engage in sidebar conversations using a communication backchannel (Figure 4-(2)).

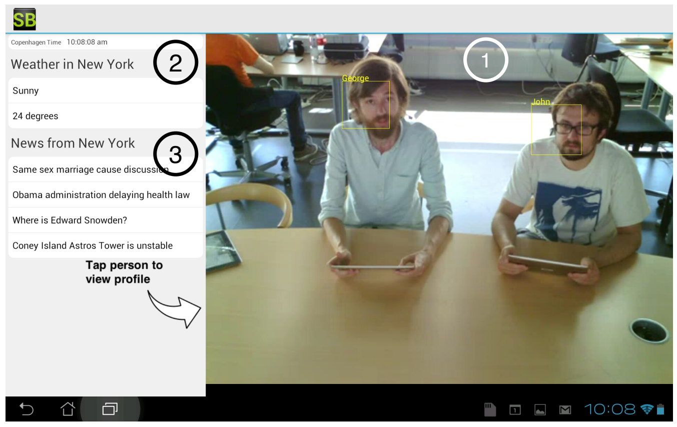
The following scenario describes how SIDEBAR is used in a video meeting.

> *Each Thursday a video-meeting between a software SME and its distant sister-company is held. The video connection is initiated and everyone logs into SideBar on their tablets. A few participants notice a person at the other site they have not seen before. They use the interactive video feed to navigate to his profile and find out that he has just been assigned to the project. Given recent re-arrangement of teams, he is now the new test manger. The project manager immediately starts the meeting leaving no time for introduction, however, the participants are able to quickly greet the new member using the communication backchannel. As the project manager explains some proposed changes, he has some questions regarding impact and estimates. Using the tablet, he can see which team member has been working on these modules and, since this person is part of the meeting, he can directly address questions to him.*

A. System Description

In the following, the main features of SIDEBAR are described.

Interactive Video Feed. The interactive video feed is the main screen on the SIDEBAR tablet computer (Figure 3-(1)). The user interface augments the video stream from the large conference display with information about each meeting participant. Using the videoconference camera, SIDEBAR tracks people in the meeting room and overlays the video image with a tracking box surrounding each face and adding a name above it. On the tablet, users can tap on the image of a remote participant to visualize his or her personal profile. This helps participants to identify and recognize remote colleagues at a glance, without the need to explicitly look up names in meeting agendas, documents, or by asking local people about who is present.



Copenhagen Time 10:08:08 am

Weather in New York

Sunny

24 degrees

News from New York

Same sex marriage cause discussion

Obama administration delaying health law

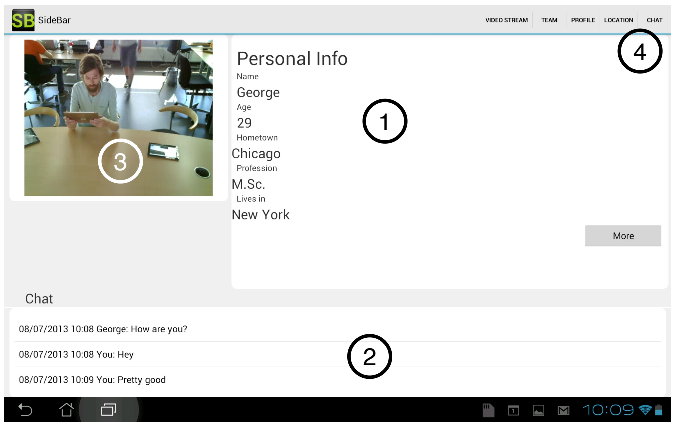
Where is Edward Snowden?

Coney Island Astros Tower is unstable

Fig. 3. Interactive video feed on the tablet computer that tracks each participant and display their name (1). Alongside, information about the weather (2) and recent news (3) of the remote site is displayed.

Association Between User and Position. Given the unreliability of face recognition techniques and their need for additional resources (i.e., precise pictures of each person the system should recognize), SIDEBAR only tracks the faces of participants, and the mapping of the person's profile with the correct tracked face is created during the login procedure. When logging in, users are asked to select their name, their geographical location, and the meeting they are participating in. After users have selected these options, a live stream of the local meeting is presented. The user is then asked to select him- or her-self in the video stream. Upon selection, the association between the actual user and a tracked face in the stream is created.

Personal Profiles and Team. Each meeting participant has a personal profile in SIDEBAR (Figure 4), which contains both personal and profession information, such as name, age, hometown, profession, education, and interests. This information provides background information and awareness between meeting participants, and can be used to acknowledge similarities between them facilitating the establishment of relationships. Similarly, a team page that provides a side-by-side overview of all members of a team is also available.



Personal Info

Name: George

Age: 29

Hometown: Chicago

Profession: M.Sc.

Lives in: New York

08/07/2013 10:08 George: How are you?

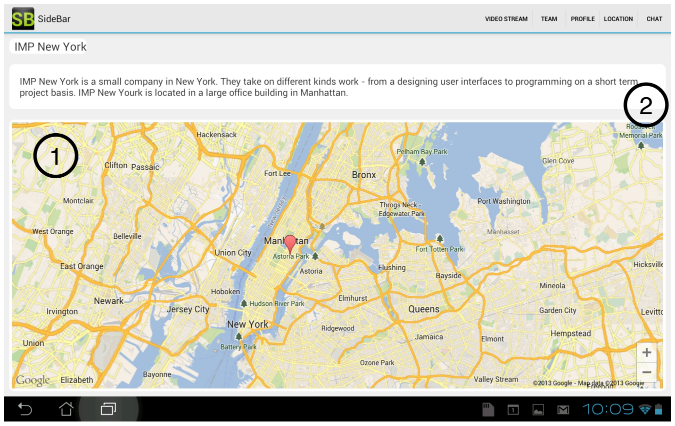
08/07/2013 10:08 You: Hey

08/07/2013 10:09 You: Pretty good

Fig. 4. The personal profile of each meeting participant contains basic personal information (1) and allows to access the communication backchannel ('chat room') with this person (2). Furthermore, the live video feed is displayed (3) and the menu for navigation in the app (4).

Communication Backchannel. One shortcoming of a standard videoconferencing setup is that it provides little opportunities for side conversations among the meeting participants [17]. As discussed above, such side conversations about both professional and more personal matters are important in building and maintaining personal connections in collocated settings. By using the tablet computer, SIDEBAR allows people to engage in backchannel conversations (Figure 4-(2)), hence, providing globally distributed actors with similar opportunities for side conversation.

Local Information. A mundane, yet often observed, obstacle to efficient communication and collaboration across distance is the lack of location information, including weather, time zone, and geography. To help build personal relations, SIDEBAR provides such information in different places. Local information on weather (Figure 3-(2)) and various relevant news feeds (Figure 3-(3)) are visualized in the main screen, and a dedicated display in SIDEBAR providing local information about the remote meeting participants and including a description of both the remote office as well as its geographical location is also provided (Figure 5).



IMP New York

IMP New York is a small company in New York. They take on different kinds work - from a designing user interfaces to programming on a short term project basis. IMP New Yourk is located in a large office building in Manhattan.

Fig. 5. The location information screen provide simple information about the geographic location of the remote user and the company that he or she is sitting at through a map (1) and a short description (2).

B. Design Methodology

The SIDEBAR system was designed in a three step process. First, we derived a number of requirements informed by the literature review presented above and field studies reported elsewhere [13], [31]. Next, an initial paper mockup of the SIDEBAR system was designed, which was refined in a user-centered design process with a software SME [32]. Finally, the system was implemented in an initial prototype.

We based our initial design on three overall requirements. First, SIDEBAR should provide users with *awareness of each other*. Second, SIDEBAR should provide users with *opportunities to connect to each other*. While technology for communication already exists—email and chat for example are extensively used in collaboration—the opportunity of interaction should exist in conjunction with the awareness of each other. Third, SIDEBAR should *integrate with existing technologies and practices*. These three overall design requirements formed the basis for adding support for social engagement in a video-conferencing system. This support was added 'on the side' by using tablet computers which allows meeting participants to maintain an awareness of each other, to provide opportunities to connect and communicate, while still being integrated with the video meeting.

We created an initial paper mockup of this design as an input to a design process with a software SME. The company makes extensive use of video meetings with its two offices located in a different parts of the world. The onshore office handles contact with customers while the offshore office handles implementation and test of software. Two workshops focusing on testing the design of the system were conducted, in addition to a series of more informal conversations and discussions of the system design. Each workshop was attended by two researchers and two employees from the company and lasted approximately two hours. The workshops were videotaped for further analysis. Figure 6 shows a snapshot from one of the design workshops.

Fig. 6. A snapshot of a design workshop with researchers and industry partners.

These design workshops gave us important insights on the feasibility of the system design and suggestions on how to improve it. First of all, the general idea of providing a SIDEBAR tablet for parallel information seeking, task-based interaction, and communication was well received. The specific UI sketches for looking up information about co-workers, for peer-to-peer communication, and for identifying meeting participants were all considered useful, and the workshop participants provided input for enhancing the UI design of these features. The design study, however, also revealed that some features were missing in the design. Among them, the lack of information—hence, awareness—about the setting and location of remote team members. Information about the local geography, weather, time zone, etc. of the remote place was also considered very useful information for making distributed colleagues more comfortable when contacting each other. Another issue that was raised during the design sessions was that the system should incorporate support for the team aspect of collaboration. The design of the paper prototypes focused on one-to-one connections between people. But considering the role of a person within the team is equally important as a general background information for meeting participants. Finally, we received suggestions for features that were not included in the final design of SIDEBAR. In particular, the participants in the workshop suggested that SIDEBAR should include information about cultural habits associated with the different locations of people. One participant for example, mentioned that small-talk with co-workers about family is common in some parts of the world whereas in other parts, this is seen as inappropriate. While such suggestions indeed are interesting, we decided not to include them in the design of SIDEBAR as such feature should require a thorough investigation of if and how traits associated with cultures can be identified and disseminated.

In summary, on the one hand the user-centered design process confirmed the overall system design of SIDEBAR , while on the other, it provided valuable input for detailed refinements as well as two new important features related to location and team awareness.

C. System Architecture & Implementation

The SIDEBAR system is composed of several interconnected sub-systems (Figure 7): *SideBar App* is the tablet application and main user interface to the system; *Relation Server* is a web server and database system handling data access to all relevant information; the *Registration* application allows for registration of new users, locations, and meetings; *VLC* [33] is used to stream video to the tablets; the *Tracking Client* handles face tracking; finally, *Skype* [34] is used as the videoconferencing system.

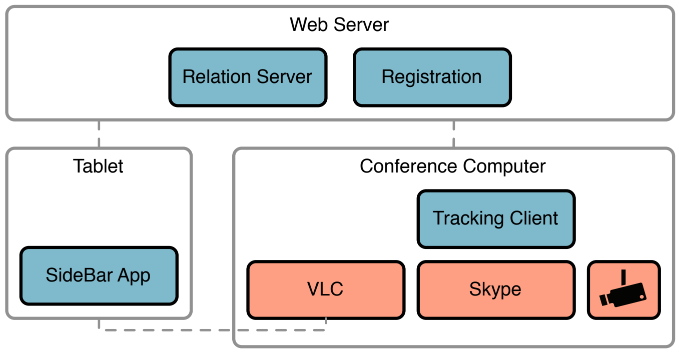


Fig. 7. Overview of the architecture of SIDEBAR. Red boxes represent existing hardware/software, while blue boxes show the novel components of SIDEBAR.

Figure 7 shows an overview of the SIDEBAR system architecture at each meeting location. The conference computer runs the Tracking Client, Skype, and VLC. The Tracking Client performs face tracking, Skype is used for the videoconference, and VLC is used to stream video to the SideBar App. A web server runs the Relation Server and Registration application. The Relation Server is a web application that handles all data access except video. Access to personal profiles, location, meeting, chat, and position information can be achieved using a REST interface of the server. New users, locations or meetings can be added to the system using the Registration application. The SideBar App runs on Android tablets and receives both video streaming over RTSP from the VLC server deployed on the conference computer as well as tracking data, personal and location data, and chat from the Relation Server running on the web server. In the following, we describe the technical details behind the SIDEBAR sub-systems.

Tracking Client. The Tracking Client is a Java application running on the video conference computer implemented via JavaCV [35]. The client determines the position of the head of each meeting participant, and post these to the Relation Server. As previously described, the Tracking Client performs face tracking and not face recognition. Faces are constantly tracked and their positions are regularly sent to the relation server. The association between face positions in the video and actual users is resolved during the login procedure. The Tracking Client constantly updates the position of people, however, it does not allow people to switch seats or leave the meeting and return without logging back in.

Video Streaming. Any videoconferencing setup can be used with SIDEBAR as long as it supports one of the following two requirements; (1) it must expose a camera that VLC and the Tracking Client can access, or (2) it must be possible to place a camera very close to the existing camera to provide a similar video stream to the SIDEBAR system. In our current implementation, Skype is used as the video conferencing technology, and since Skype, VLC, and the Tracking Client are running on the same computer, they all have access to the same camera. Video streaming to the tablets is done using VLC. VLC captures the camera feed and exposes it as an RTP stream playable by the default Android media API. The video is encoded with .h264 and streamed at a resolution of 480 x 360 and a bit rate of 500Kbps.

Relation Server. The Relation server is responsible for data handling of SIDEBAR. The server is implemented in Java using the Java Spring framework and runs in an Apache Tomcat server. The Relation Server stores all information regarding user locations and meetings as well as all position information as tracked by the tracking client and makes this information available through a REST interface to the tablet computers.

SideBar App. SIDEBAR App is the tablet application and user interface of the SIDEBAR system. SIDEBAR App is implemented in Android 4.0 and is designed to run on a 10.1 inch tablet. SIDEBAR App handles the login procedure, streaming video, backchannel communication, and implements the profiles, location and team profiles.

IV. EVALUATION

To gather early feedbacks on SIDEBAR, we performed a usability experiment. The key research questions (RQ) addressed by this evaluation were:

RQ1 *How useful are the core features of* SIDEBAR*?* Do users perceive SIDEBAR's features useful for improving videoconference meetings, or not?

RQ2 *Is* SIDEBAR *easy to use?* Do users perceive SIDEBAR as easy to use, or not?

RQ3 *Is the use of* SIDEBAR *during meetings a distraction?* Do users feel distracted when using SIDEBAR during videoconference meetings; does it improve their focus; or do they experience no difference in using it.

These research questions were approached by assessing the so-called 'perceived ease of use' and 'perceived usefulness' of SIDEBAR. Research has shown that there is a strong correlation between user acceptance of a technology and its perceived ease of use and usefulness [36]. In addition to perceived usability and usefulness, we also wanted to explore the notion of 'distraction', since introducing additional devices like the SIDEBAR tablet into a meeting may lead to distraction of the participants.

A. Method

Given the novelty of the system, we opted to expose study participants to SIDEBAR through a scenario-based approach. This method entails the design of scenarios based on realistic settings exposing participants to complex situations, which would otherwise be hard to observe [37]. Each evaluation session consisted of three phases: a briefing, a scenario, and a debriefing.

In the first phase, participants were welcomed, the SIDEBAR tablet application was demonstrated to them, and they were given the opportunity to get accustomed to the application. During this phase the experimenter was present with the participants to answer any questions or concerns regarding both the evaluation procedure as well as the SIDEBAR system.

In the second phase, participants were divided in two groups, were administered a sheet describing the scenario and their role (details below), and were asked to reach their designated position: two rooms were used to simulate two different locations. During this session, they were observed by the experimenter but interactions were kept at a minimum to avoid potential bias.

In the debriefing phase, participants were called back to the initial room and were administered a questionnaire. The questions addressed the three research questions and participants were asked to rate the questions on a 5-point Likert scale[1]: (RQ1) the usefulness of the main features of SIDEBAR (i.e., interactive video feed, personal profiles, location, communication backchannel, and team information); (RQ2) the extend to which they agreed with the statement *"The system is easy to use."*; and, (RQ3) the extend to which they agreed with the statement *"The use of tablets distracts the video meeting."*. After completing the questionnaire, participants were subject to a semi-structured interview including questions about the general experience with the system. The interview allowed participants to elaborate on their previous answers and to provide detailed comments. Interviews were recorded for further analysis.

Evaluation Scenario. The scenario addressed coordination in distributed software development. A software development scenario was chosen to promote discussion among participants by accommodating the background of the people envisioned as potential candidates for the recruitment, which happened within the SSS department[2]. The scenario focused on a kickoff meeting in which two remote teams of developers had to discuss the design and implementation of a smartphone application. Participants were asked to plan the project using their knowledge about the competences of each other, and divide the work between application development and user interface design. Furthermore, participants were asked to schedule future meetings. Rather then emphasizing the creation of personal connections, the evaluation scenario described a common software scenario, i.e. the kickoff meeting.

The purpose of the study was to investigate whether SIDEBAR was used in connection-making processes without explicitly asking participants to do so. Each participant received a slightly different version of the scenario, which described their specific roles in the scenario. Since participants knew each other, they were all equipped with artificial identities and roles. Participants were asked to enact the character described in the scenario, which required them to 'learn to know each other', which again promoted conversation and the use of the the social features in SIDEBAR .

B. Participants & Setup

We recruited a total of seven participants for the evaluation (mean age 32, all male) for two sessions (4 and 3 participants respectively). The participants were a mix of master students, PhD students, and research assistants. Two compulsory requirements were considered during the selection process related to the experience in software development and the experience in collaborating with distant people using Skype for communication. Besides these requirements, no other inclusion/exclusion criteria were applied. Two meeting rooms were used for the evaluation, each equipped with SIDEBAR. The setup comprised a computer, a large screen, a high resolution webcam with microphone, speakers, and a tablet for each participant. Figure 9 shows a picture from an evaluation session.

[1]Likert scale parameters: 1 (strongly disagree) to 5 (strongly agree).

[2]SSS department: Software and Systems Section, IT University of Copenhagen, Denmark.

Feature	*Min*	*Q1*	$\tilde{x}$	*Q3*	*Max*	*iqr*
Interactive video	4	4	4	5	5	1
Personal profiles	3	3.25	4	5	5	1.75
Location page	1	1.5	3	3	4	1.5
Communication backchannel	4	4	4	5	5	1
Team Page	1	1.5	4	4	5	2.5
Statement	*Min*	*Q1*	$\tilde{x}$	*Q3*	*Max*	*iqr*
The system is *easy to use*	3	4	4	4	5	0
The use of tablets *distracts* the video meeting	1	1.25	4	4	4	2.25

Fig. 8. Questionnaire result on a 5-point Likert scale. For each feature, the table shows the reported minimum score (*Min*), the first quartile (*Q1*), the median *($\tilde{x}$)*, the third quartile (*Q3*), the maximum (*Max*), and the inter quartile range (*iqr*).

C. Results

The results of the questionnaire are shown in Table 8. In the following we describe these results in detail.

Interactive Video Feed. The interactive video feed was one of the SIDEBAR features that scored highest (*$\tilde{x}$=4; iqr=1*). Participants liked the interactive video feed and it was found very useful for navigating. One participant even *"felt surprised how useful integrating information and video is"* and mentioned that the interactive video feed *"added more depth to the video meeting"*. During the evaluation, we observed that the participants quickly picked up on the names of each other, and used the augmented video feed to reassure themselves about the name of another person before directing a question to this person. Participants were also able to use the interactive video feed to navigate to the personal profiles. Thereby, participants quickly recognized the roles of each other and were able to make the connection between the video image and the person.

Personal Profiles. The personal profile pages were also well received by the participants (*$\tilde{x}$=4*), but with some disagreement (*iqr=1.75*). During the evaluation, the participants actively used the profiles to seek out information about each other. The data they gathered from these profiles were used to direct questions at the right person. The profiles were also used to asses how to divide the work involved in creating the application mentioned in the scenario, thus, aiding them in performing the task described in the scenario. In one session, for example, a participant noted, while referring to the remote site: *"you guys are mostly UI designers [. . .]"*. This information was then used to argue for a particular division of work.

Location. The location page was the feature that was rated lowest in the evaluation (*$\tilde{x}$=3; iqr=1.5*). When asked about

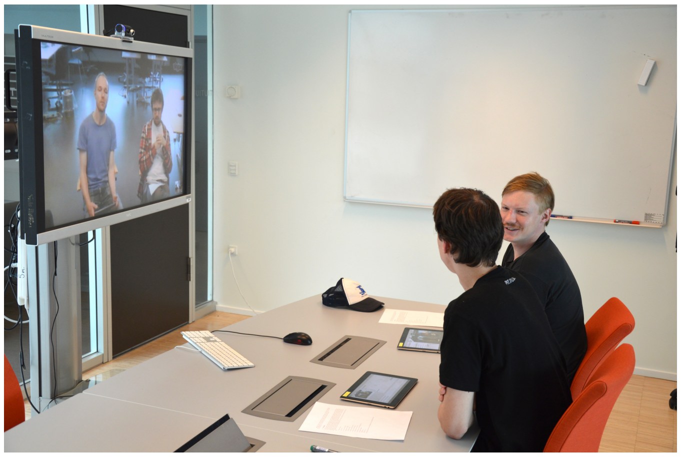


Fig. 9. Evaluation setup. A team of four persons—two at each location—is having a video meeting using SIDEBAR including the larger video display, a camera, and a tablet computer for each meeting participant.

the page after the evaluation, most participants noted that the information was not particularly useful to them. The information provided by the map was, however, used in the evaluation. For example, one participant asked, while referring to the position of the company being located near Central Park in New York: *"so your office is near Central Park?"*. Hence, getting access to local information about the remote site did spark a more informal conversation, thereby building knowledge about the remote participants.

Communication Backchannel. The communication backchannel also scored high in the questionnaire (*$\tilde{x}$=4; iqr=1*). The participants noted that it provided them with an easy way of sharing textual messages and notes during the meeting. Several participants mentioned that with traditional videoconferencing setups, one-to-one channels for chatting or sharing information are not easily accessible. The participants used the chat actively in the meeting to greet each other and towards the end of the meeting to share relevant information, such as email and web addresses.

Team Profile. There was less agreement on the usefulness of the team information (*$\tilde{x}$=4; iqr=2.5*). This was also evident from the observations, for some participants used the page actively, while others did not. In the former case, one participant found the team information useful to support the beginning of the meeting in which the teams briefly introduced themselves. In the latter case, another participant found the information in the team page to be redundant, as the same information was available on the profile pages. He noted that it just caused more navigation within the system.

Ease of use. Participant found the system easy to use (*$\tilde{x}$=4; iqr=0*), and throughout the evaluation, participants were able to navigate within the system without any problems. One remark, however, focused on the fact that the top-menu of the SIDEBAR App is associated with a person, therefore, not accessible on the display showing the video. This caused some confusion as participants were looking for the menu when interacting with the video screen.

Distraction. Participants disagreed on the question of how distracting the tablets were in the video meeting (*$\tilde{x}$=4; iqr=2.25*). On the one hand, some participants did not find that the tablets would disturb the meeting and some argued that SIDEBAR would not add more disturbances than the ones already existing. As one participant argued; *"it [*SIDEBAR*] does not disturb more than, for example, printed meeting agendas."* Another noted that smart phones and computers are already extensively used during meeting today. On the other hand, two participant noted that eye contact in the video meeting cease when people turn to the tablet. One participant said; *"you think you have eye-contact as you see the same video on the tablet—but you don't."* Lastly, a participant expressed concern that turning to the tablet felt like turning away from the meeting.

D. Limitations

This study is a preliminary evaluation and, as such, presents several limitations posing threats to the validity of its results. The main limitations, which were identified and addressed, are briefly discussed in the following.

First, being a preliminary evaluation, no statistical significance, scalable, or generalizable results were sought, as the key objective was to systematically gather and interpret empirical evidence about the perceived ease of use of the system, the perceived usefulness of its core functionalities, and the user perception of the system as a distraction. With regards to the external validity, even if participants were not practitioners and it could be argued that the sample was not fully representative of the intended population, all participants had software development experience and had experience with remote collaboration. Second, aware of the complexity of assessing collaboration technologies [38], we carefully designed this preliminary evaluation by leveraging established methods (i.e., [37]). Regarding the ecological validity, the scenario was extensively discussed to be as realistic as possible in the simplifications that were applied. This process led us to the decision of simulating a videoconferencing meeting involving three to four people per session separated into two teams physically distributed in different rooms. A hands-on training session was also included before the main scenario to mitigate bias connected to the use of a novel technology. Third, given that our purpose was to gather initial understandings on SIDEBAR, rather than utilizing an established instrument for collecting data, we preferred an ad hoc questionnaire comprising a set of very focused questions. Therefore, the questionnaire used has been designed by the authors, and it was only reviewed and discussed with colleagues knowledgeable and experienced in empirical research. This poses threats to the construct validity; however, the decision allowed us to keep the number of questions to a minimum avoiding lengthly instruments appropriate to more extensive evaluations. Forth, internal validity. Even though all participants satisfied the selection constraints, affiliation with the authors represents a bias. Nonetheless, such recruiting approach is a common practice for preliminary evaluations. Additionally, to avoid experimenter bias, observer/participants interactions were reduced to a minimum by providing participants with a scenario including predesigned roles as described Section IV-A.

Finally, it is worth to notice that no technical validation for the system or its features was conducted. The purpose of the evaluation was purely to get feedback on the design of SIDEBAR, hence, technical aspects were not considered. In a future evaluation, the stability and integrity of the system in a realistic setting should be evaluated.

V. Discussion

The evaluation of SideBar was designed to provide insights on three research questions. In this section, we discuss these insights.

[RQ1] *How useful are the core features of* SideBar*?* The evaluation showed that the participants found SideBar useful and the participants commented on SideBar as being *"really helpful"* and *"the right way to go"*. In particular, participants liked the linking of video and personal information through the interactive video feed. Furthermore, the communication backchannel was also appreciated. The evaluation, thus, shows promising results for the support of social engagements in video meetings. Nonetheless, some participants found specific features (i.e., the team and location information) less useful compared to the video stream and the communication backchannel. Interestingly, these features were derived from the design workshops we conducted. This points to the possible gap between the design and evaluation; SideBar was designed in collaboration with industry partners, but evaluated with students and researchers. These two aspects of SideBar also contained some redundant information; the team page contained some information that was also available in the profile pages, and the location information was split between the interactive video feed (news and weather) and the location screen (description and map). The information about people and location could be possibly redesigned to be available in one screen to ease navigation.

[RQ2] *Is* SideBar *easy to use?* SideBar was found easy to use and participants did not have problems navigating the application, seeking information about each other or using the communication backchannel. A few participants found it confusing that the navigation menu (Figure 4-(4)) was not accessible from the main interactive video screen but only from the profile screens. This suggests that some improvements in terms of navigating in the app should be made.

[RQ3] *Is the use of* SideBar *during meetings a distraction?* Participants responded very differently to this question. To really investigate how the introduction of a tablet-based technology like SideBar changes video meetings, a more thorough study should be made, possibly comparing meetings with SideBar to meetings without. For now, it is hard to say whether such a technology would introduce distractions that would disturb the video meetings.

The evaluation of SideBar also pointed out some areas of improvements. Two participants mentioned that the augmented video feed could display even more information than the current implementation offers, including information from the personal profile pages, and highlighting those relevant for the meeting. Also, one participant suggested that the information exposed by SideBar should be available even outside the meeting session. In the current implementation of SideBar, the personal profiles are available only after login—a procedure that requires an ongoing meeting. The suggestion is particularly interesting to consider as in line with our future plans. In fact, on the one hand, social engagements activities are not limited to the meetings per se, and allowing out-of-meeting usage of SideBar might improve their support. On the other hand, considering that features similar to the ones implemented by SideBar (e.g., personal profiles and communication back channel) are, in some cases, already provided within the ecology of tools used in distributed collaboration projects, integrating existing online social profiles or company chat applications would clearly both facilitate user adoption of the technology as well as increase the chances of finding an industrial partner for performing a field deployment of a company-specific version of SideBar.

VI. Designing for Social Engagement

Establishing more personal and non-work relations in collaborative settings is an important part of successful collaboration. This is particularly relevant in distributed and global collaboration in which supporting social engagements is more challenging. Therefore — we argue — the design of technologies for distributed collaboration should incorporate support for this social dimension. This section discusses ways of designing for social engagement in distributed collaboration technologies.

We designed and implemented SideBar based on the three design objectives that the system should: support awareness amongst meeting participants, provide them with opportunities to connect to each other, and integrate with existing technologies and practices. These design goals can be generalized to a model of how to design for social engagement in distributed collaboration, as illustrated in Figure 10. Collaborative technologies should provide support for social engagement through: *(i)* relational context awareness, *(ii)* relation building and sharing, and *(iii)* relationship maintenance. In other words, the core design approach is to design for a mutual awareness of relational context, which may trigger relational building and sharing, which again builds and maintains relationships.

A. Relational Context Awareness

Maintaining an awareness of the nature of relationships between members of an organization is core to social connection making. This happens while overhearing desktop discussions, ad-hoc queries, small exchanges during coffee breaks and in the hallway, and while setting up a meeting. Central to relational context awareness is that people build knowledge about each other and the relationships they are involved in. Design for relational context awareness is evident in architectural design of office space, which is designed with open space where people easily can see and overhear each other, and the office layout is designed so that people easily 'bump into' each other. Similarly, the use of shared artifacts like visible post-it notes, print-outs or drawings on a shared wall allows for the same kind of awareness that collocation brings.

When designing technologies for social engagement, relational context awareness is of great importance. Technologies should seek to provide actors with a sense of awareness about each other. SideBar was designed to support relational context awareness by helping people render relevant relationships visible for others, and for people to be able to monitor the relationships of colleagues. The evaluation of SideBar showed that participants appreciated the awareness and the association between video and personal profiles and the information were actively used in the scenarios.

B. Relation Building and Sharing

A common way to build relationships is the classic team building exercises, company dinners or similar social activities. These activities are all explicitly designed to bring people together within a non-work context. One a more daily basis, informal talks around a shared office space helps establish and maintain connections between people.

Collaborative technologies should seek to provide users with opportunities for building and sharing relations. SIDEBAR provides users with a communication backchannel in video meetings. This channel is accessible from the personal profiles, providing an easy link from the relational context awareness information. As one participant put it, *"…this [technology] could replace a kick off meeting"* which fits well with the intention of designing for relationship building.

C. Relationships Maintenance

Once connections have been established, they need to be maintained and remembered. Often people keep specific artifacts such as pictures, tokens, prizes, diplomas, toys, and award medals from e.g. team building activities as souvenirs and reminders of specific relationships. However, more active involvement is often required to maintain relationships and keep them alive. Social media such as Facebook and Twitter, for example, encourages users to regularly update their online profiles with current information about their doings, interests, and whereabouts etc.

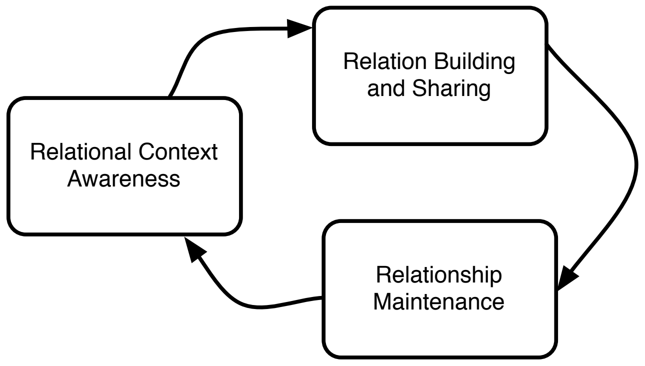


Fig. 10. Design guidelines for social engagement. The core design approach is to design for a mutual awareness of relational context, which may trigger relational building and sharing, which again builds and maintains relationships.

In summary, technology should provide relational context awareness, giving access to relationship building and sharing, which in turn promotes relationship maintenance and thereby creating new awareness. If these three processes are supported, a positive spiral of social engagement is achieved in a collaborative setting. By tapping into the design model shown in Figure 10, SIDEBAR is designed to integrate with existing videoconferencing equipment while providing continuous support for social engagement.

VII. CONCLUSION

In this paper we presented SIDEBAR: a videoconferencing system with a special focus on supporting social engagement. Through the use of face tracking, SIDEBAR offers an interactive mirrored video feed of the ongoing videoconference on tablets, which allows meeting participants to seek information about each other and engage in backchannel conversations. SIDEBAR was designed in a user-centered design process involving a software company implemented in a functional prototype. A preliminary study of SIDEBAR showed that participants found the system easy to use and appreciated both the interactions using an augmented video feed as well as the introduction of a communication backchannel. Based on the design and evaluation of SIDEBAR, we presented three guidelines for the design of technology supporting social engagement revolving around the concepts of *relational context awareness*, *relation building and sharing*, and *relationship maintenance*. In the future, we plan to integrate SIDEBAR into a larger suite of tools for distributed software engineering and evaluate the system in a larger study.

VIII. ACKNOWLEDGMENTS

We would like to thank the participants of the design and evaluation workshops. This research has been funded by the Danish Agency for Science, Technology and Innovation under the project "Next Generation Technology for Global Software Development", #10-092313.

REFERENCES

[1] P. J. Hinds and D. E. Bailey, "Out of sight, out of sync: Understanding conflict in distributed teams," *Organization Science*, vol. 14, no. 6, pp. 615–632, Nov. 2003. [Online]. Available: http://dx.doi.org/10.1287/orsc.14.6.615.24872

[2] J. Herbsleb, "Global software engineering: The future of socio-technical coordination," in *Future of Software Engineering, 2007. FOSE '07*, May 2007, pp. 188–198.

[3] J. Noll, S. Beecham, and I. Richardson, "Global software development and collaboration: Barriers and solutions," *ACM Inroads*, vol. 1, no. 3, pp. 66–78, Sep. 2011. [Online]. Available: http://doi.acm.org/10.1145/1835428.1835445

[4] S. Deshpande, I. Richardson, V. Casey, and S. Beecham, "Culture in global software development - a weakness or strength?" in *Global Software Engineering (ICGSE), 2010 5th IEEE International Conference on*, Aug 2010, pp. 67–76.

[5] G. M. Olson and J. S. Olson, "Distance matters," *Hum.-Comput. Interact.*, vol. 15, no. 2, pp. 139–178, sep 2000. [Online]. Available: http://dx.doi.org/10.1207/S15327051HCI1523_4

[6] S. Sarker and S. Sahay, "Implications of Space and Time for Distributed Work: An Interpretive Study of US-Norwegian Systems Development Teams," *Eur. J. Inf. Syst.*, 2004.

[7] J. Herbsleb and A. Mockus, "An empirical study of speed and communication in globally distributed software development," *Software Engineering, IEEE Transactions on*, 2003.

[8] J. Kotlarsky and I. Oshri, "Social ties, knowledge sharing and successful collaboration in globally distributed system development projects."

[9] S. Kiesler and J. N. Cummings, "What do we know about proximity and distance in work groups? a legacy of research," pp. 57–80, 2002.

[10] B. A. Nardi, "Beyond bandwidth: Dimensions of connection in interpersonal communication," *J. Comput.-Supp. Coop. Work*, vol. 14, pp. 91–130, 2005.

[11] O. Gotel, V. Kulkarni, M. Say, C. Scharff, and T. Sunetnanta, "Quality indicators on global software development projects: does getting to know you really matter?" *Journal of Software: Evolution and Process*, vol. 24, no. 2, pp. 169–184, 2012. [Online]. Available: http://dx.doi.org/10.1002/smr.474

[12] C. A. Fulmer and M. J. Gelfand, "At What Level (and in Whom) We Trust: Trust Across Multiple Organizational Levels," *Journal of Management*, 2012.

[13] P. Bjørn and L. R. Christensen, "Relation work: Creating socio-technical connections in global engineering," in *ECSCW 2011: Proceedings of the 12th European Conference on Computer Supported Cooperative Work, 24-28 September 2011, Aarhus Denmark*. Springer London, 2011, pp. 133–152. [Online]. Available: http://dx.doi.org/10.1007/978-0-85729-913-0_8

[14] B. Lings, B. Lundell, P. J. Agerfalk, and B. Fitzgerald, "A reference model for successful Distributed Development of Software Systems," in *Proceedings of the International Conference on Global Software Engineering*, 2007.

[15] N. Bos, N. S. Shami, J. S. Olson, A. Cheshin, and N. Nan, "In-group/out-group effects in distributed teams: An experimental simulation," in *Proceedings of the 2004 ACM Conference on Computer Supported Cooperative Work*, ser. CSCW '04. New York, NY, USA: ACM, 2004, pp. 429–436. [Online]. Available: http://doi.acm.org/10.1145/1031607.1031679

[16] R. Daft and R. Lengel, "Information Richness: A New Approach to Managerial Behaviour and Organizational Design," *Research in Organizational Behaviour*, 1984.

[17] E. A. Isaacs and J. C. Tang, "What Video Can and Can'T Do for Collaboration: A Case Study," in *Proceedings of the First ACM International Conference on Multimedia*, 1993.

[18] R. Vertegaal, I. Weevers, C. Sohn, and C. Cheung, "Gaze-2: conveying eye contact in group video conferencing using eye-controlled camera direction," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI '03. New York, NY, USA: ACM, 2003, pp. 521–528. [Online]. Available: http://doi.acm.org/10.1145/642611.642702

[19] K. Kim, J. Bolton, A. Girouard, J. Cooperstock, and R. Vertegaal, "Telehuman: effects of 3d perspective on gaze and pose estimation with a life-size cylindrical telepresence pod," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI '12. New York, NY, USA: ACM, 2012, pp. 2531–2540. [Online]. Available: http://doi.acm.org/10.1145/2207676.2208640

[20] C. Gutwin, S. Greenberg, and M. Roseman, "Workspace awareness in real-time distributed groupware: Framework, widgets, and evaluation," in *Proceedings of HCI on People and Computers XI*, ser. HCI '96. London, UK, UK: Springer-Verlag, 1996, pp. 281–298. [Online]. Available: http://dl.acm.org/citation.cfm?id=646683.702625

[21] D. T. Nguyen and J. Canny, "More than face-to-face: empathy effects of video framing," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI '09. New York, NY, USA: ACM, 2009, pp. 423–432. [Online]. Available: http://doi.acm.org/10.1145/1518701.1518770

[22] P. Slovák, P. Novák, P. Troubil, P. Holub, and E. C. Hofer, "Exploring trust in group-to-group video-conferencing," in *CHI '11 Extended Abstracts on Human Factors in Computing Systems*, ser. CHI EA '11. New York, NY, USA: ACM, 2011, pp. 1459–1464. [Online]. Available: http://doi.acm.org/10.1145/1979742.1979791

[23] D. Grayson and A. Anderson, "Perceptions of proximity in video conferencing," in *CHI '02 Extended Abstracts on Human Factors in Computing Systems*, ser. CHI EA '02. New York, NY, USA: ACM, 2002, pp. 596–597. [Online]. Available: http://doi.acm.org/10.1145/506443.506501

[24] T. Jenkin, J. McGeachie, D. Fono, and R. Vertegaal, "eyeview: focus+context views for large group video conferences," in *CHI '05 Extended Abstracts on Human Factors in Computing Systems*, ser. CHI EA '05. New York, NY, USA: ACM, 2005, pp. 1497–1500. [Online]. Available: http://doi.acm.org/10.1145/1056808.1056950

[25] C. Kuster, T. Popa, J.-C. Bazin, C. Gotsman, and M. Gross, "Gaze correction for home video conferencing," *ACM Trans. Graph.*, vol. 31, no. 6, pp. 174:1–174:6, nov 2012. [Online]. Available: http://doi.acm.org/10.1145/2366145.2366193

[26] C. Harrison and S. Hudson, "Pseudo-3d video conferencing with a generic webcam," in *Multimedia, 2008. ISM 2008. Tenth IEEE International Symposium on*, 2008, pp. 236–241.

[27] K.-I. Okada, F. Maeda, Y. Ichikawaa, and Y. Matsushita, "Multiparty videoconferencing at virtual social distance: Majic design," in *Proceedings of the 1994 ACM conference on Computer supported cooperative work*, ser. CSCW '94. New York, NY, USA: ACM, 1994, pp. 385–393. [Online]. Available: http://doi.acm.org/10.1145/192844.193054

[28] K. Misawa, Y. Ishiguro, and J. Rekimoto, "Livemask: a telepresence surrogate system with a face-shaped screen for supporting nonverbal communication," in *Proceedings of the International Working Conference on Advanced Visual Interfaces*, ser. AVI '12. New York, NY, USA: ACM, 2012, pp. 394–397. [Online]. Available: http://doi.acm.org/10.1145/2254556.2254632

[29] H. Nakanishi, K. Kato, and H. Ishiguro, "Zoom cameras and movable displays enhance social telepresence," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI '11. New York, NY, USA: ACM, 2011, pp. 63–72. [Online]. Available: http://doi.acm.org/10.1145/1978942.1978953

[30] G. Aranda, A. Vizcaí andno, R. Palacio, and A. Morá andn, "What Information Would You Like to Know about Your Co-worker? A Case Study," in *Global Software Engineering (ICGSE), 2010 5th IEEE International Conference on*, 2010.

[31] L. R. Christensen, R. E. Jensen, and P. Bjørn, "Relation work in collocated and distributed collaboration," in *Proc. COOP 2014*. Springer, 2014.

[32] D. A. Norman and S. W. Draper, "User centered system design," *New Perspectives on Human-Computer Interaction, L. Erlbaum Associates Inc., Hillsdale, NJ*, 1986.

[33] VLC, "http://www.videolan.org/." [Online]. Available: http://www.videolan.org/

[34] Skype, "http://www.videolan.org/." [Online]. Available: http://www.videolan.org/

[35] JavaCV, "https://code.google.com/p/javacv/." [Online]. Available: https://code.google.com/p/javacv/

[36] F. D. Davis, "Perceived usefulness, perceived ease of use, and user acceptance of information technology," *MIS Quarterly*, vol. 13, no. 3, pp. 319–339, September 1989.

[37] G. Convertino, D. C. Neale, L. Hobby, J. M. Carroll, and M. B. Rosson, "A laboratory method for studying activity awareness," in *Proceedings of the third Nordic conference on Human-computer interaction*, 2004.

[38] J. Whitehead, "Collaboration in software engineering: A roadmap," *Future of Software Engineering (FOSE '07)*, 2007.

dBoard 1

Title of Paper

The dBoard: a Digital Scrum Board for Distributed Software Development

Authors:

Morten Esbensen, *Paolo Tell, Jacob B. Cholewa, Mathias K. Pedersen, Jakob E. Bardram*

Conditionally Accepted:

*In Proceedings of the 10th International Conference on Interactive Tabletops and Surfaces, **(ITS '15)**, ACM, 2015*

Abstract:

In this paper we present the dBoard – a digital scrum board designed for distributed agile software development teams. The dBoard connects two locations with live video and audio which is overlaid with a synchronized touch-enabled digital scrum board. The dBoard is designed to work (i) as a passive information radiator from which it is easy to get an overview of the status of work, (ii) as a media space providing awareness about the presence of remote co-workers, and (iii) as an active meeting support tool. The dBoard senses when people are in front of it and automatically blurs the video when no people are present and de-blurs the video when people are approaching to allow for easily initiating meetings. We present the motivation, design and implementation of the dBoard and report on an initial usability study which shows that users found the dBoard both useful and easy to use. Based on this work, we suggest that superimposing collaborative applications onto live video is a useful way of designing collaborative videoconferencing applications.

The dBoard: a Digital Scrum Board for Distributed Software Development

Morten Esbensen, Paolo Tell, Jacob B. Cholewa, Mathias K. Pedersen & Jakob Bardram
IT University of Copenhagen
Rued Langgaards Vej 7, 2300 Copenhagen S
{mortenq,pate,jbec,mkin,bardram}@itu.dk

ABSTRACT
In this paper we present the dBoard – a digital scrum board designed for distributed agile software development teams. The dBoard connects two locations with live video and audio which is overlaid with a synchronized touch-enabled digital scrum board. The dBoard is designed to work (i) as a passive information radiator from which it is easy to get an overview of the status of work, (ii) as a media space providing awareness about the presence of remote co-workers, and (iii) as an active meeting support tool. The dBoard senses when people are in front of it and automatically blurs the video when no people are present and de-blurs the video when people are approaching to allow for easily initiating meetings. We present the motivation, design and implementation of the dBoard and report on an initial usability study which shows that users found the dBoard both useful and easy to use. Based on this work, we suggest that superimposing collaborative applications onto live video is a useful way of designing collaborative videoconferencing applications.

Author Keywords
dBoard; Scrum Board; Videoconferencing; Scrum; Task Board

ACM Classification Keywords
H.5.3 Group and Organizational Interfaces: Computer Supported Cooperative Work

INTRODUCTION
In todays globalized business world, distributed collaboration is becoming ever more widespread. This trend has been enabled by new and faster technologies that allow distributed co-workers to instantly connect across distances. One such strong enabling technology is videoconferencing. From the early ages of the internet where bandwidth was limited and video quality as a result suffered, the video has evolved into a ubiquitous technology with high resolution and low latency. As a high communication bandwidth channel, video is well suited for distributed collaboration where complex tasks are solved collaboratively by co-workers across different countries.

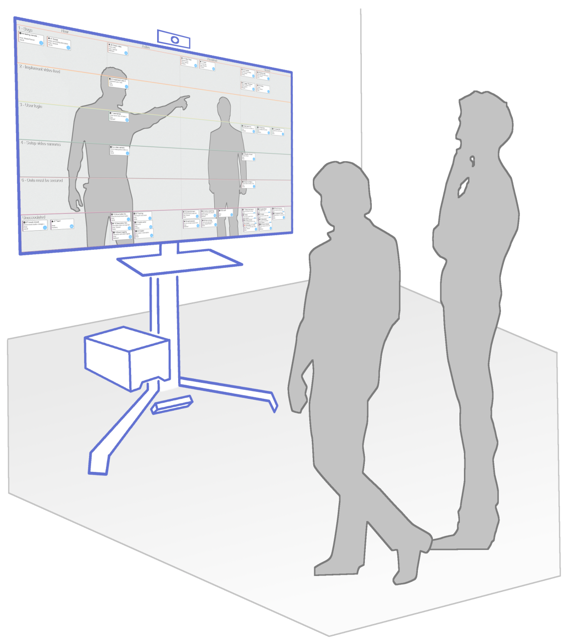


Figure 1. The dBoard is an interactive scrum board and videoconferencing tool designed for distributed teams.

However, video is not without shortcomings and studies have shown that videoconferencing is often conducted as part of a shared experience [5]. When people meet using video it is often with a specific task or purpose at hand. As a result, there is a need for integrating videoconferencing with other systems that are used in such arrangements [26]. Likewise, traditional videoconferencing systems often are only designed for planned meetings, thus, not supporting the nuanced awareness information that lies in seeing and informally interacting with co-workers in a shared office space [2].

One particular area of work in which video communication is central is distributed agile software development. Agile development like scrum puts an emphasis on close collaboration and frequent meetings and in order to achieve such arrangement in a distributed environment video is essential [19]. Often central to the scrum process is the scrum board around which many other scrum practices revolve [18]. The board serves two main functions; during the daily meetings it is used to guide the discussion, and during the everyday work

Conditionally Accepted for ITS 2015.

it is used as a passive information radiator. The scrum board thus is an example of a tool which is not used for one specific task but as a supporting artifact over many collaborative practices. While concerns have been raised with respect to replacing physical taks boards with digital counterparts [23], a recent study suggests that companies will adopt and use technological solutions as their scrum board [3] and technological readiness is emerging in the modern business world of distributed software development [1].

In this paper, we present the design, implementation and evaluation of the dBoard – a digital scrum board designed specifically for distributed agile software development teams. The dBoard is shown in figure 1; it is designed to function as an information radiator, a media-space and as a scrum meeting support tool providing support for seamless transitions between these modes. Instead of separating videoconferencing and scrum board, the dBoard integrates these two by superimposing the scrum board specific user interface elements onto a full screen video-channel. We implemented the dBoard as a web-application that runs on large multitouch-enabled screens and conducted an initial scenario-based user evaluation with scrum practitioners which succeeds in revealing that participants found the dBoard both useful and easy to use.

In the remainder of this paper we present related works relevant to the design of the dBoard. We then report on the ethnographic field studies of a distributed agile software development team that helped us inform the design of the dBoard. We describe the features and implementation of the dBoard and report on a scenario-based user evaluation with scrum practitioners. Finally, we conclude by discussing the results and by suggesting the potential for this kind of collaborative video systems as an innovative way of designing videoconferencing system for distributed collaboration.

RELATED WORK

The dBoard builds on different research on immersive teleconferencing, media spaces and scrum boards. In this section we therefore review related works in these areas.

Immersive Teleconferencing

The idea of superimposing a shared user interface on top of videoconferencing has been researched previously but largely as a way of preserving eye-contact or conveying gestures to provide more immersive teleconferencing. Early prototypes demonstrated how to preserve eye-contact and convey gestures in shared drawing applications. ClearBoard [15] for example, was designed based on a metaphor of looking through and drawing on a transparent window into another office. This line of research have since been extended to include systems that capture the image of a user through half-silvered screens to provide parallax free videoconferencing systems that convey eye contact [17, 25]. Recently, advanced prototypes have taken a step further, using 3d-sensors in conjunction with cameras to capture people and provided interfaces in which 3d realistic representations of participants are presented blended with user interface elements [13, 27]. Despite such focus on gaze and gesture, these systems demonstrate an idea in which videoconferencing and task specific user interfaces can be combined into one technology instead of keeping each tool in its own window or on separate screens.

Media Spaces

Traditional videoconferencing setups have mostly been designed for planned meetings however, in collocated environments, much information is conveyed outside meetings by being aware of the presence of nearby colleagues or during informal talks outside scheduled meetings. This fact has been recognized by system designers who have proposed videoconferencing systems capable of providing such awareness information. Portholes [8] demonstrated an early version of such system where still images were broadcasted throughout an office space to provide co-workers sitting in different offices with awareness of each other. The PARC media space [2] and VideoWindow [11] connect different offices using live video. More recently, systems such as MirrorSpaces [21] and Pêle-Mêle [12] further explored the concept of media spaces by offering always-on video systems that take into account the proximity of people to the system screen as a means of altering the video. This is especially interesting in the context of this paper as the dBoard incorporates proximity awareness to mediate between the modalities of active meeting and passive awareness tool.

Scrum Board

The scrum board has been recognized as an important artifact in the scrum development framework. The board works as a boundary object, helps reduce the amount of articulation work [20] and serves as a central artifact around which other scrum practices revolve [18]. Despite the fact that several commercial products offer digital versions of the board, less studies have been dedicated to such solutions. Sharp et. al. argue that transitioning to digital task boards can become problematic as some affordances of the traditional task board lie in the fact that it is physical [23]. Nonetheless, Boden et. al. have shown how their implementation of the concept of articulation spaces were appropriated to be used as a scrum board for a collocated software development team [3]. Finally, Rubart has proposed using an interactive tabletop-based scrum board as a tool for collocated scrum [22]. However, in general research on scrum boards is scarse and little is known about how to design scrum boards for distributed teams.

In summary, previous studies have investigated from a technological perspective how to address the problems of integrating video and task specific applications or creating media spaces for informal communication and awareness in distributed collaboration. In this paper, we draw on these works, bring them together and into the domain of distributed scrum to develop the dBoard – a combined videoconferencing system and collaborative scrum board.

DESIGN

The dBoard is designed as part of a larger project studying the work of distributed software developers[1]. The motivation of

[1]The *anonymized* project is a research project dedicated to studying global software developement.

designing the dBoard came partly from the related work we described and partly from a set of ethnographic field studies of distributed software developers, and the design was done in a user-centered iterative design process. In total, four different companies were involved in the studies and design process.

Field Studies of Distributed Software Development

Field studies of distributed software development have been undertaken in four different companies, involving sites in Denmark, Philipines and India. Different findings from these studies have been reported elsewhere [1, 9, 16] but, in this paper, we focus on how these studies informed the dBoard.

Common to the studies was that software development was set up as an agile process across distributed team members. Hence, the goal was to adopt an agile software development methodology in which software developers should work as a coherent scrum team of people, despite their physical and temporal separation. All field studies applied participant observations in which we observed daily work and meetings of the developers and conducted onsite interviews of team members.

Our observations identified a set of recurrent challenges in running an agile, distributed software development process. First, the agile methodology encourages team members to do a short and efficient daily stand-up meeting for easy and low-level coordination. Such meeting should take place in the shared workspace of the team in front of the scrum board and should not exceed more than 15 minutes. However, once the team is distributed, setting up and running daily stand-up meetings was cumbersome and often impossible. The problems often were related to the lack of proper tool support in the right location – both for video meetings and for handling tasks on a scrum board. For example, a team distributed between Denmark and India would try to meet at the same time each day (9:45 CET in Denmark and 1:15 PM in India). For this to happen, the team first had to go to dedicated video-conferencing rooms at each site. Then they had to set up the videoconference call and all the software development tools needed. The meetings were usually initiated by an Indian developer setting up all the required systems and then calling the Danish side. The team used a traditional videoconferencing setup in conjunction with the HP Application Lifecycle Manager (ALM) system to manage and update their tasks. As this application was not designed for collaborative use, the scrum master – who was located in Denmark – controlled the ALM application while the team in India used VPN and Remote Desktop Connection to connect to the scrum master's computer and project the application onto a screen next to the videoconferencing screen. As such, the amount of work needed to set up and conduct a stand-up meeting was extremely demanding, and, when the experienced developer was not around to start the meeting, we observed how the others struggled to get the video conferencing, ALM, VPN and Remote Desktop Connection up and running. Hence, these meetings were far from the ideal agile stand-up meeting taking place in the developers' workspace and only lasting 15 minutes.

A second recurring challenge we observed in the distributed teams was the lack of mutual awareness, especially outside the daily stand-up meetings. In one instance for example, two Indian developers were working on the same bug. When using a traditional scrum board in a collocated team workspace, such a situation would seldom happen, since a developer would physically go to the board and move the 'bug ticket' from one column to another, thereby signaling a status change. However, in the distributed setup there was no way for a developer to notice such a change if not explicitly communicated. To account for this lack of awareness between the developers, the lead developer located in Denmark contacted the Indian developers on a daily basis using instant messaging. This process allowed him to get an overview of the status of work, but this information then only resided with him and no mutual awareness amongst team members was maintained.

Designing the dBoard

Based on the field studies we designed the dBoard as an augmented version of the traditional physical one. We first built a prototype that integrated video and scrum board features into one single tool designed to run on a large multitouch surface. This prototype was presented to a compay that works with distributed scrum in a workshop. Based on the feedback from this workshop we formulated the following set of guidelines that was used to guide the final design and implementation of the dBoard. Namely, the dBoard should:

- Provide active support during scrum meetings
- Function as an information radiator
- Give awareness about the presence of remote co-workers
- Seamlessly transition between information and awareness radiator and meeting support tool
- Integrate with existing software engineering tools
- Be easily movable and deployable
- Provide mechanisms for ensuring the privacy feeling of the users.

SYSTEM OVERVIEW

The dBoard is a combined videoconferencing and scrum board tool that is designed to support distributed agile software developers. The dBoard is designed to work in pairs; two dBoards should be setup at the locations of two collaborating parties connecting these locations with video and audio. The main feature of the dBoard is its videoconferencing ability which functions as a window to another office. On top of this video-window, a synchronized digital touch-enabled scrum board is superimposed. Furthermore, the scrum board senses when people are in front of a board and adjusts its behavior accordingly. In this section we describe these features in detail.

Video Window

The dBoard applies a 'window' metaphor to its video conferencing features. The whole background of the dBoard user interface is a large video stream from a connected dBoard designed to give same the feeling that can be experienced when looking through a window into another office. All other user interface elements such as the scrum board and the menus are

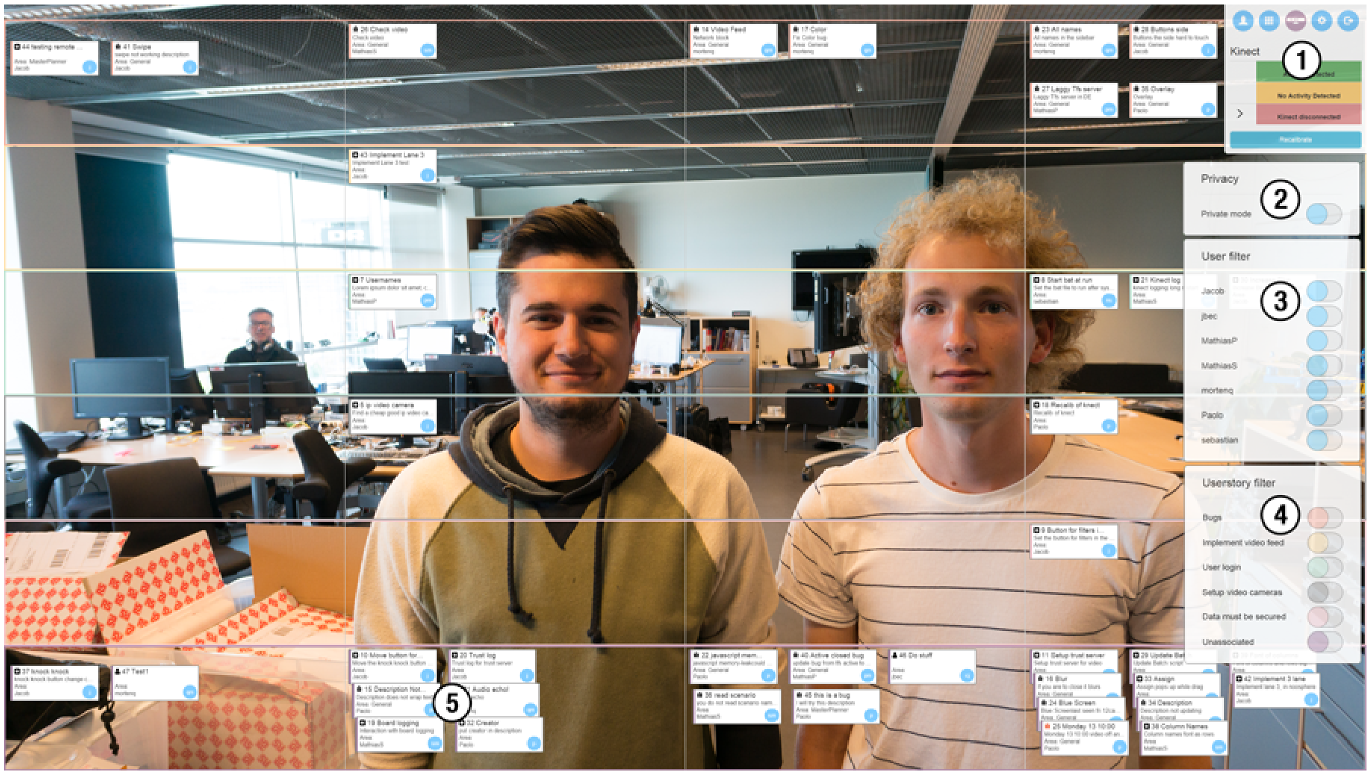


Figure 2. The user interface of the dBoard. (1) Menu, (2) Privacy menu, (3) User filters, (4) Story filters, (5) Tasks

placed on top of this video. Similar to a media space [2], the dBoard video is always on and is started automatically when the application is started. The dBoard also captures and streams audio from a connected microphone. This always-on feature makes the initiating of a meeting as simple as walking up to a board. To get the attention of people at the location of another dBoard, we implemented a 'knock knock' feature. When a user performs a knocking gesture on the board, a knocking sound is played at the other board signaling that someone requests attention.

Information Radiator

The state of the dBoard is kept synchronized across the two connected boards. All task movements on one board are immediately updated on the other in a WYSIWIS [24] fashion. As in traditional videoconferencing setups, the camera on the dBoard is mounted on top of the stand, which makes it impossible to point to specific tasks due to the parallax. To address this issue we implemented interaction awareness on the screen. When a user drags a task, the task is highlighted in red on both boards. When a user touches anywhere else on the board, a small red pointer is shown on both boards. Furthermore, as described later, the dBoard is synchronized with existing software development tracking systems such that the board is automatically updated even if tasks are updated from outside of the dBoard system.

We built a prototype of the dBoard mounted on a wheeled stand that can easily be moved anywhere. Alternatively, the screen of the dBoard can be setup using a wall display. The important aspect that is highlighted is, that the hardware of the dBoard system can be setup in many configurations as it doesn't require any specially engineered hardware or large physical space which allows the dBoard to be deployed in the place that best fits its needs or that would be chosen for placing a physical board – for example among the developers.

Task Management

The scrum board elements of the dBoard are superimposed onto the full screen video stream from the other dBoard (see Figure 2). The scrum board is organized as a traditional physical board; tasks are represented as small digital post-it notes that are arranged into columns representing their state and rows representing the user story they belong to. As some tasks do not belong to specific user stories and as some user stories do not contain any tasks, we implemented a *Unassociated* row where all such items are placed. The tasks display information about their name, their type, the user story they belong to, their description and the person assigned to it as seen in Figure 3. Tasks can be repositioned using touch, and dragging a task from one column to another changes its state. Tasks can be placed anywhere on the board and do not 'snap' into position when dropped allowing users to arrange the tasks on the board in a similar way that post-it notes can be arranged on a physical board. The progress update of tasks thus also follows those on a physical board; tasks are moved from one column to another as they are being completed. To assign a task to a developer, users can tap the task's name or user icon to access a dropdown list of people to whom the task can be assigned. The dBoard shows all tasks and user stories

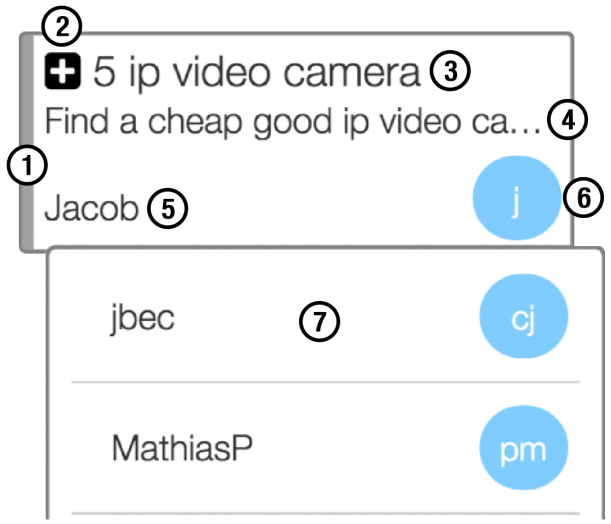


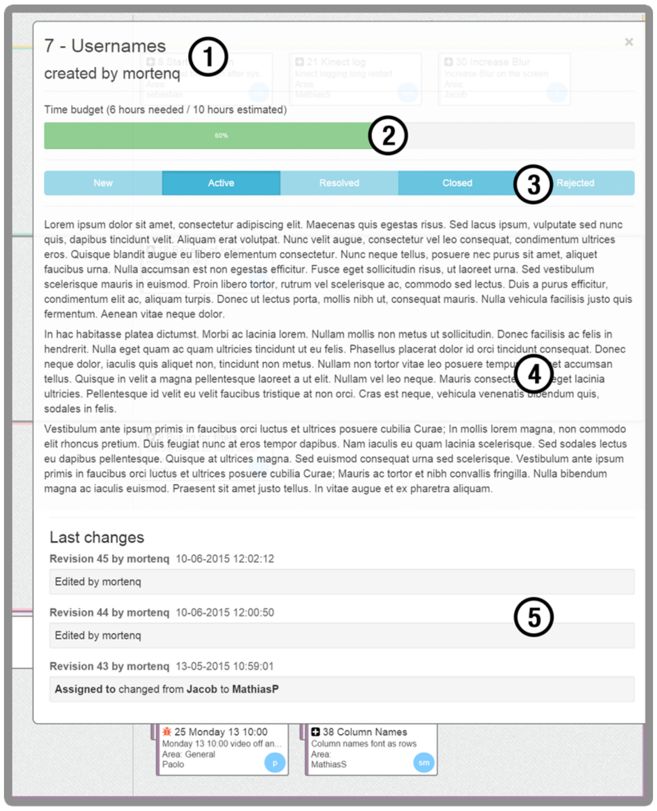
Figure 3. A closeup of a task on the dBoard. (1) Task color, (2) Task type, (3) Title, (4) Description, (5) Assigned user, (6) Assign button + User initial, (7) Assign dropdown

for one sprint at a time. When signing in to the application, users select which sprint to work on.

We also implemented a number of options to filter and sort the tasks on the board. Two filter menus can be used to highlight specific tasks based on two parameters: user and user story. When a user selects a user or user story filter, the dBoard opacifies all tasks that do not match the given filter. As tasks can be dragged around the board and positioned anywhere we also implemented two ways to sort the board. When sorting, tasks are automatically positioned in the row and column of the scrum board layout that correspond to their state and user story. To sort the entire board, a user can tap the sorting button in the menu. To sort only a specific tile (i.e. row/column intersection), the user can tap and hold that tile. As the state of the board is kept synchronized across the two connected sites, sorting in one end results in tasks on the other board being sorted as well. Lastly, we implemented a user mode to the board. By tapping the user mode button in the menu, the dBoard rearranges itself to show users as rows rather than user stories. This means that all tasks are re-positioned to be located in the row representing the user they belong to. Tapping this button again brings the board back to show user stories as rows.

Detailed Task Information

Tasks on the dBoard are deliberately designed to be small post-it notes-like elements. These notes contain the very basic information about the task however; to access detailed information users can bring up a side panel. The panel (Figure 4) slides in from the right side of the board when a use performs a tap-and-hold gesture on a task or a user story and it contains much more detailed information about the task. The side panel contains: task name, full description, state information and legal states, time estimate and change history list. From the side panel a user can change the time estimate by performing a dragging gesture on the progress bar or change the state by tapping one of the legal state buttons. The side



7 - Usernames
created by mortenq

Time budget (6 hours needed / 10 hours estimated)

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Maecenas quis egestas risus. Sed lacus ipsum, vulputate sed nunc quis, dapibus tincidunt velit. Aliquam erat volutpat. Nunc velit augue, consectetur vel leo consequat, condimentum ultrices eros. Quisque blandit augue eu libero elementum consectetur. Nunc neque tellus, posuere nec purus sit amet, aliquet faucibus urna. Nulla accumsan est non egestas efficitur. Fusce eget sollicitudin risus, ut laoreet urna. Sed vestibulum scelerisque mauris in euismod. Proin libero tortor, rutrum vel scelerisque ac, commodo sed lectus. Duis a purus efficitur, condimentum elit ac, aliquam turpis. Donec ut lectus porta, mollis nibh ut, consequat mauris. Nulla vehicula facilisis justo quis fermentum. Aenean vitae neque dolor.

In hac habitasse platea dictumst. Morbi ac lacinia lorem. Nullam mollis non metus ut sollicitudin. Donec facilisis ac felis in hendrerit. Nulla eget quam ac quam ultricies tincidunt ut eu felis. Phasellus placerat dolor id orci tincidunt consequat. Donec neque dolor, iaculis quis aliquet non, tincidunt non metus. Nullam non tortor vitae leo posuere tempu [illegible] et accumsan tellus. Quisque in velit a magna pellentesque laoreet a ut elit. Nullam vel leo neque. Mauris consecte [illegible] eget lacinia ultricies. Pellentesque id velit eu velit faucibus tristique at non orci. Cras est neque, vehicula venenatis bibendum quis, sodales in felis.

Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; In mollis lorem magna, non commodo elit rhoncus pretium. Duis feugiat nunc at eros tempor dapibus. Nam iaculis eu quam lacinia scelerisque. Sed sodales lectus eu dapibus pellentesque. Quisque at ultrices magna. Sed euismod consequat urna sed scelerisque. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Mauris ac tortor et nibh convallis fringilla. Nulla bibendum magna ac iaculis euismod. Praesent sit amet justo tellus. In vitae augue et ex pharetra aliquam.

Last changes

Revision 45 by mortenq 10-06-2015 12:02:12

Edited by mortenq

Revision 44 by mortenq 10-06-2015 12:00:50

Edited by mortenq

Revision 43 by mortenq 13-05-2015 10:59:01

Assigned to changed from **Jacob** to **MathiasP**

Figure 4. The side panel of dBoard can be used to see detailed information about tasks and user stories. (1) Title, (2) Time estimate, (3) State, (4) Rich text description, (5) Change history

panel can be hidden again by performing a swipe gesture on it.

Proxemic Interaction

To provide seamless transition between awareness tool and meeting tool, the video of the dBoard is automatically blurred and de-blurred and the audio turned on and off by sensing proximity of people. If no people are in front of the dBoards at both ends, the video is blurred and the audio is turned off. If people are present at only one of the dBoards, the video is blurred less in both ends while the audio is kept turned off. If people are present in both ends, the video is completely un-blurred and the audio is turned on. The blur is used to enhance visibility of the scrum board – when blurred, the contours in the video are washed out and the colors are faded thus bringing forward the non-blurred user interface elements of the scrum board. Furthermore, the audio is only turned on if people are present at both end to ensure that the dBoard does not become too much of a distraction if placed in a busy working environment. While blur has been shown to be able to provide the balance between awareness and privacy [4], we also implemented a privacy button that instead of blurring the video removes it altogether until people are present at both boards. Figure 5 shows a cutout of the dBoard without blur, with blur and with privacy enabled.

To provide transparency to users with respect to what the dBoard senses, the proximity button in the menu changes color based on the state of the sensing. When green, the button signals that the dBoard has sensed one or more people in front of the board, yellow signals that the sensing is running but no people are seen, while flashing red signals a disconnection to the sensing application. This information is also available when tapping the button; in this case a dropdown appears in which the colors are explained in more detail and the current state is highlighted. This dropdown also contains a calibration button that can be used to calibrate the sensing application in case of erroneous sensing.

Synchronization
All sprints, user stories, tasks, bugs and users are synchronized with a Microsoft Team Foundation Server (TFS) installation. We chose this platform as it is a widely adopted software management platform. The synchronization makes sure that all information on the dBoard is taken from a TFS instance and that all changes made either on the dBoard or on another system are kept synchronized across the dBoards and TFS. In case of inconsistency between the dBoard and TFS due to network problems, TFS is considered the master. As explained later, we implemented the dBoard to be able to integrate to other software management systems as well.

IMPLEMENTATION
The dBoard system consists of the two main parts: the *dBoard*, which is the board itself; and the *backend* which handles communication between dBoards and integration with external tools. This section presents the implementation of these parts.

dBoard
The dBoard hardware consists of a large high-resolution screen with a PQ Labs multitouch overlay. A connected high-definition camera and a quality microphone captures video and audio. All the hardware is mounted on a movable stand and all components are connected to a fan-less computer strapped to the stand. The setup ensures that setting up and starting the dBoard is as simple as moving it to the desired location and plugging in power. The Kinect v2 sensor used to sense proximity to the board is mounted between the wheels of the stand. Alternatively, the components can be dismounted from the stand the screen hanged on a wall.

On startup, the dBoard launches the dBoard frontend application in fullscreen. The dBoard frontend is an HTML5 and JavaScript application built with AngularJS and a designed

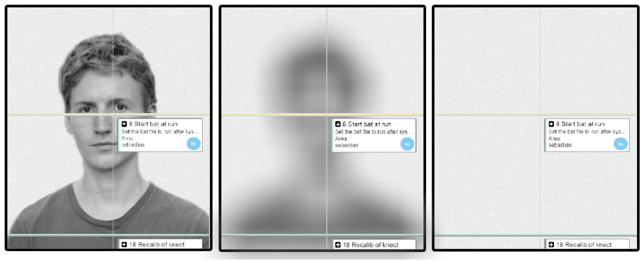


Figure 5. The dBoard in its normal mode (left), with blurred video (middle, and with privacy enabled (right).

for Google Chrome. Two connected dBoards send and receive video and audio using WebRTC, which is a peer-to-peer based JavaScript framework while all other information (such as tasks and user stories and state information) is communicated through the server using web-sockets.

Proximity Sensing
The dBoard senses proximity with a Microsoft Kinect v2 attached between the wheels of the system. The Kinect is accessed through the official .NET SDK in a C# application which analyzes skeletal information and depth frames to infer human presence. Depth frames are used in addition to skeletal tracking as it is possible for people to stand too close to the dBoard for the Kinect to pick up skeletal information. The depth frames are analyzed in two ways: large frame-by-frame pixel changes suggest movement in front of the sensor and a large summed difference between the current frame and a calibration frame suggests proximity. In case presence information is sensed for an extended period of time without any movements happening, the application assumes that a non-human artifact (such as a chair) has been placed in view of the camera and the application recalibrates automatically. A menu button also allows to manually request a recalibration of the proximity sensing application. All presence information is sent to the dBoard web application over a websocket.

Backend
As shown in Figure 6 the dBoard backend consists of three main components: (i) a Node.js web server, (ii) a Microsoft Team Foundation Server (TFS), and (iii) the activity-centered infrastructure NooSphere [14] with a plugin to communicate with the TFS instance.

The Node.js server is responsible for serving the AngularJS web application to the dBoards along with facilitating web socket connections for task and interface synchronization between the boards and the backend. Any change made on the dBoard is reflected in NooSphere which in is turn reflected on the TFS and vice versa. In case of conflict, the TFS will always take precedence. NooSphere exposes all scrum activities (i.e. sprints, user stories, tasks, bugs and users) trough a RESTful web interface. We added this extra layer on top of TFS as it allows for easily changing the task tracking software from TFS to, e.g., Jira without modifying the rest of the application. Changing or adding task management systems can be done by writing a plugin. Other tools would then in turn also be able to access the same data in NooSphere making it an infrastructure able to support other collaborative tools.

STUDY
To evaluate the concepts behind the design of the dBoard and collect user feedback, we conducted a scenario-based user evaluation [6]. The goals of this study were to (i) observe how participants would use the system for performing ad-hoc and standup meetings, (ii) elicit user input on the perceived usefulness and ease of use of the dBoard, and (iii) gather feedback on the main features provided by the dBoard.

Participants
The study was conducted by contacting local software development companies to participate to a workshop in our lab. In

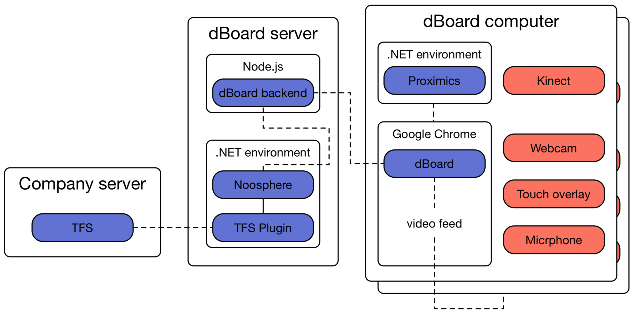


Figure 6. An overview of the architecture of dBoard with its three main components; the TFS server, the dBoard server and the dBoards

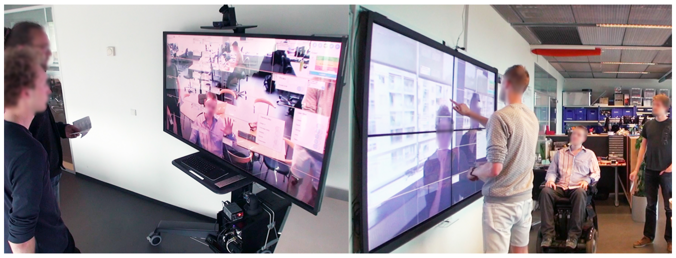


Figure 7. Snapshot of a workshop setup taken during Scenario II—the 'standup meeting'.

the workshops we would have demonstrated the dBoard and they would have had the opportunity to try it out first hand. Strong requirements for their selection were that (i) they did not participate in the design stages and (ii) their team had to be using scrum to run their projects. Three different companies participated in the study for a total of seven people (age $\mu = 30{,}5$, $\sigma = 4{,}53$). Participants reported to be experienced agile and scrum practitioners, their seniority ranges from 6 months to 10 years ($\mu = 4{,}3$; $\sigma = 3{,}98$). Among them, 2 cover scrum master roles, 2 product owner roles, and the rest varies from developers to testers.

Method

As shown in Figure 7, the setup comprised one board placed on the stand depicted in Figure 1 and one displayed on a large wall display located in our lab. This decision was taken to highlight the flexibility of the hardware form factor of the dBoard. The workshop was divided into four parts: (i) an introduction to the dBoard in which the participants were welcomed to the study, signed an informed consent form, filled out a demographic questionnaire, and were given a detailed presentation of the dBoard features; (ii) a hands-on session in which the participants were invited to experiment with the two different dBoards; (iii) a session in which we asked participants to act in two different scenarios; and, (iv) a closing group discussion in which participants, after completing a short survey, were asked to reflect on the dBoard and elaborate on their survey answers by providing additional insights and feedback.

The two scenarios were chosen from common activities in traditional distributed scrum teams. Scenario I required the participants to first perform a simple design task (i.e., select a template for a website design based on inspirations from existing websites); update the status of the task on the dBoard; and, contact the remote team (enacted by confederates [6]) to communicate that the design task was completed as well as seek information about the progress on the remote site. Scenario II simulated a standup meeting; participants were given a script describing the role of the enacted character (one of which was the scrum master dictating the pace of the meeting), the work done on the previous day, the one planned for the coming day, and whether there were some impediments or clarifications required from the remote site.

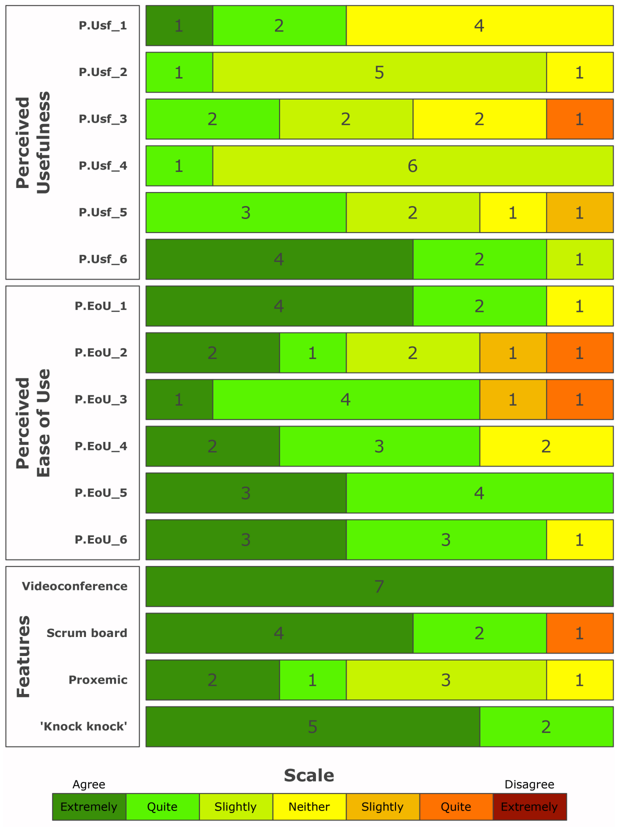


Figure 8. The results of the 7-point Likert-scale questionnaire on the perceived usefulness, perceived ease of use, and usefulness of the different features of the system. The numbers in the bars represent the amount of participants that selected a given score.

The survey was constructed based on the technology acceptance model (TAM) [7] to which four questions were added to assess the usefulness of specific dBoard features (i.e., videoconferencing, scrum board, privacy through proxemic, and 'knock knock'). In line with what has been used in [7], each question was measured with a seven-point Likert scale[2].

Results

Figure 8 depicts an overview of the results of the questionnaire on the perceived usefulness, perceived ease of use, and usefulness of the different features of the dBoard.

[2]Likert scale parameters: (1) extremely likely, (2) quite likely, (3) slightly likely, (4) neither, (5) slightly unlikely, (6) quite unlikely, (7) extremely unlikely.

Perceived Usefulness

Even though overall participants scored positively the usefulness of the dBoard, on average they remained quite neutral with answers that varied across participants. In particular, they reported to be uncertain about the ability of the dBoard to enable them to accomplish tasks more quickly (P.Usf_1: $\tilde{x}$ = 4, iqr = 2); a pattern that is repeated across the next four questions related to the ability of the system to: improve performance (P.Usf_2: $\tilde{x}$ = 3, iqr = 0); improve productivity (P.Usf_3: $\tilde{x}$ = 3, iqr = 1,5); enhance effectiveness on the job (P.Usf_4: $\tilde{x}$ = 3, iqr = 0); and, make it easy to do the job (P.Usf_5: $\tilde{x}$ = 3, iqr = 1,5). Nonetheless, on a more direct question about the usefulness of the dBoard in their job, the participants scored the dBoard very positively (P.Usf_6: $\tilde{x}$ = 1, iqr = 1). Compared to the remaining questions – especially with the group on feature usefulness – these results appear contradicting; therefore, we will provide a possible explanation in the Discussion Section by interpreting further these mixed results.

Perceived Ease of Use

All participate managed to appropriate the system with more or less confidence in a very brief period of time (P.EoU_5: $\tilde{x}$ = 2, iqr = 1); they reported the dBoard to be very easy to operate (P.EoU_1: $\tilde{x}$ = 1, iqr = 1) and to use (P.EoU_6: $\tilde{x}$ = 2, iqr = 1). The analogy with the post-it notes often used in physical scrum boards was well received; participants appreciated the flexibility provided by the dBoard in terms of freely arranging tasks at arbitrary locations (P.EoU_4: $\tilde{x}$ = 2, iqr = 1,5). However, even though still scored positively, the overall interaction with the dBoard received some negative scores (P.EoU_3: $\tilde{x}$ = 2, iqr = 1,5); a trend visible also in the question related to the easiness with which participants could make the dBoard act as desired (P.EoU_2: $\tilde{x}$ = 3, iqr = 2,5). These less aligned results will be further analyzed in the Discussion Section.

Feature Usefulness

When confronted with more specific questions about the usefulness of the different features provided by the dBoard, participants provided very positive scores. Especially with regards to the videoconferencing capabilities, all participants considered this feature extremely useful (Videoconference: $\tilde{x}$ = 1, iqr = 0) and provided comments like:

– *"The entire video is really cool – really helpful." – P3 – "You are actually interacting with your colleagues that you do not normally see." – P5*

The scrum board features were also received very well (Scrum board: $\tilde{x}$ = 1, iqr = 1); participants found the integration with TFS to be an essential feature for a digital version of a scrum board as it represents the main shortcoming of the physical counterpart, and they saw a significant value in the affordances provided by the dedicated setup mimicking physical scrum boards. One participant stated:

– *"One of the things I like of a Scrum board like this is that you have not the digital but the post-its like interactions, and you can walk up there, take a task up here, and put it over here. I know you can do it on your computer—it might be easier—but I just like that feeling, I like to be able to do that." – P5*

Moreover, the simplicity of the scrum board representation was also appreciated; as one of the participants ironically pointed out:

– *"[speaking about a feature] it is definitely also there [in Jira], but it would take a couple of days to find out where." – P7*

In relation to the proxemic feature of the dBoard to support gradual engagement and privacy, participants were less satisfied ('Proxemic': $\tilde{x}$ = 3, iqr = 1,5). If on the one hand, they valued the ability of mitigating the disturbance of a constant video feed from a large display through video blurring and sound removal:

– *"Without the blur, it would be like in a sport bar during a soccer match where everyone's attention is captivated by the game being streamed." – paraphrasing P2, – "I like the idea that when one team is working, they can see when a person is at the board." – P4*

On the other hand, the quick and automatic de-blurring of the video feed due to the detection of presence on the remote site was considered by some as potentially distracting.

Finally, the 'knock knock' feature was also very well received ('Knock knock': $\tilde{x}$ = 1, iqr = 1) insomuch that they suggested additional applications for it, as it will be presented in the Discussion Section. One participant stated:

– *"[...] we use Slack for things like 'do you have time for a Skype call?'. And that is basically the 'knock knock' right, so I think you got it pretty good there!" – P3*

DISCUSSION

We designed and implemented the dBoard to support different aspects of distributed agile software development. The design of the dBoard aimed to reduce the costs of conducting distributed standup meetings, getting awareness information and informally meeting with distant co-workers to resolve issues. The evaluation of the dBoard revealed a general high level of perceived usefulness and ease of use and several questions and comments were raised. In this section, we discuss the results of the evaluation and suggest that future collaborative systems might benefit from adopting the concept blending video and user task specific user interfaces.

The evaluation of the usefulness of the dBoard showed some mixed results. On one hand, the *overall* usefulness and the four distinct features of the dBoard were rated highly useful. On the other hand, the TAM specific usefulness parameters received more moderate results. We speculate that this difference is caused by the way the TAM scale is constructed. The scale evaluates perceived usefulness based on parameters such as *performance*, *productivity* and *effectiveness*. The dBoard is a multi-purpose tool designed to support a wide range of collaborative activities in distributed scrum; from the planned meetings, over information radiating to the informal presence awareness. These functions have not been designed with a specific focus on improving *speed* or *effectiveness* but as a way of supporting a long term collaborative practice.

While results of the evaluation showed some divergence in the perceived usefulness, the results pointed to a general high level of ease of use. The dBoard , however, still suffers from a few performance issues that can affect ease of use which was also reflected in the evaluation scores of the TAM (P.EoU_2 and P.EoU_3). During the evaluations, some interactions with the board such as dragging multiple items simultaneously around or performing swipe gestures suffered from 'lagging'. Despite the computational power of today, we experienced that implementing such a complex collaborative application as a web application could cause some performance bottlenecks. We believe however, that with the speed at which modern computers and browsers are evolving, transitioning to implementing rich collaborative tools as web applications is a promising direction. This has the main advantage that web applications are easy to deploy, they can be used on any operating system and can be updated by deploying a new version on the server rather than forcing users to manually update their applications.

The features of the dBoard that were scored in the evaluation were also very well received. During the evaluation however, several issues and suggestions for improvements were raised by the participants. A common theme of the participants' feedback was related to what happens if too much information is presented on the dBoard. Indeed, if lots of user stories and tasks are placed on the board, there's a risk of covering most of the video. Another line of comments was related to the usage of proxemics. Several participants mentioned that they would prefer to have the de-blurring of video triggered by the knock-knock feature rather than the proxemic interaction. We do believe however, that using proximity on the board is valuable as it allows for more than binary operations and it would allow adopting behavior that is based on the number of people sensed. Finally, some participants suggested that the dBoard should be able to include more sites. This is interesting and opens up new questions related to how to handle and position the videos from many participants and how to handle e.g. knock knock.

Collaborative Videos

Based on our work of designing, implementing and evaluating the dBoard, we argue that combining video with collaborative user interfaces using a 'collaborative window' metaphor in which the application is superimposed onto the video is a promising design guideline for collaborative applications. While such systems have been designed and implemented before (e.g. [25, 27]), this has mostly been with a focus on conveying eye-contact, gaze and gestures in video mediated communication. In this work, investigated the creation of a compelling collaborative application for a specific domain rather than solving the problems of eye-contact and gestures. Despite the camera parallax that exists in the dBoard, we believe that the idea of blending video and task specific user interface carries great value. In particular, we argue that such applications allows for a richer collaborative experience as they don't require users to constantly switch attention from the video to the tasks at hand.

These user interfaces however also open up for other questions. While we have demonstrated with the dBoard an example of combining video and a scrum board the question arises of how these two user interface layers relate to each other. In the example of SideBar for instance, the overlay is linked to the video by means of image processing [10] but in the dBoard there is no connection between video and overlay. An intersting line of research would be to investigate how to relate the layers to each other in a meaningful way. One example that could solve the observation made by several participants related to the amount of information on the screen could be to use image processing to arrange the user interface in such a way that it does not cover the place in the video where people are.

In the future, we are planning to keep developing the dBoard and to gather more feedback from practitioners. We are also pursuing the idea of designing more collaborative applications build on the metaphor of a collaborative window to gather more evidence on how such systems are able to support distributed collaboration. Lastly, we are looking into how such systems can be extended to include more than two sites.

CONCLUSION

In this paper we presented the design, implementation and evaluation of the dBoard – a digital distributed scrum board that blends together videoconferencing and scrum task management. This is achieved by bringing them together in a setup where the scrum board specific elements are superimposed onto the video. The dBoard was designed to provide support both as a passive information radiator from which the state of work can be collected, as a media space proving awareness about the presence of remote co-workers, and as an active meeting support tool. By sensing the proximity of people the dBoard provides a way to seamlessly switch between these three modes of operations.

We designed the dBoard based on observations of distributed scrum practices and implemented the system as a web application that runs on large multitouch enabled screens. The dBoard was evaluated in a scenario based setting with experienced scrum practitioners. The evaluation revealed that users found the dBoard both useful and easy to use. In particular, the combination of video and scrum board was very well received. Based on this work, we argue that adopting the metaphor of a 'collaborative window' as a means of designing collaborative videoconferencing systems can lead to new and interesting applications supporting distributed collaboration.

ACKNOWLEDGMENTS

Anonymized

REFERENCES

1. Pernille Bjørn, Morten Esbensen, Rasmus Eskild Jensen, and Stina Matthiesen. 2014. Does Distance Still Matter? Revisiting the CSCW Fundamentals on Distributed Collaboration. *ACM Trans. Comput.-Hum. Interact.* 21, 5, Article 27 (Nov. 2014), 27:1–27:26 pages.

2. Sara A. Bly, Steve R. Harrison, and Susan Irwin. 1993. Media Spaces: Bringing People Together in a Video, Audio, and Computing Environment. *Commun. ACM* 36, 1 (Jan. 1993), 28–46.

3. Alexander Boden, Frank Rosswog, Gunnar Stevens, and Volker Wulf. 2014. Articulation Spaces: Bridging the Gap Between Formal and Informal Coordination. In *Proceedings of the 17th ACM Conference on Computer Supported Cooperative Work & Social Computing (CSCW '14)*. ACM, New York, NY, USA, 1120–1130.

4. Michael Boyle, Christopher Edwards, and Saul Greenberg. 2000. The Effects of Filtered Video on Awareness and Privacy. In *Proceedings of the 2000 ACM Conference on Computer Supported Cooperative Work (CSCW '00)*. ACM, New York, NY, USA, 1–10.

5. Jed R. Brubaker, Gina Venolia, and John C. Tang. 2012. Focusing on Shared Experiences: Moving Beyond the Camera in Video Communication. In *Proceedings of the Designing Interactive Systems Conference (DIS '12)*. ACM, New York, NY, USA, 96–105.

6. Gregorio Convertino, Dennis C. Neale, Laurian Hobby, John M. Carroll, and Mary Beth Rosson. 2004. A Laboratory Method for Studying Activity Awareness. In *Proceedings of the Third Nordic Conference on Human-computer Interaction (NordiCHI '04)*. ACM, New York, NY, USA, 313–322.

7. Fred D. Davis. 1989. Perceived usefulness, perceived ease of use, and user acceptance of information technology. *MIS Q.* 13, 3 (1989), 319–340.

8. Paul Dourish and Sara Bly. 1992. Portholes: supporting awareness in a distributed work group. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '92)*. ACM, New York, NY, USA, 541–547.

9. Morten Esbensen and Pernille Bjørn. 2014. Routine and Standardization in Global Software Development. In *Proceedings of the 18th International Conference on Supporting Group Work (GROUP '14)*. ACM, New York, NY, USA, 12–23.

10. Morten Esbensen, Paolo Tell, and Jakob E Bardram. 2014. SideBar: Videoconferencing system supporting social engagement. In *Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom), 2014 International Conference on*. IEEE, 358–367.

11. Robert S. Fish, Robert E. Kraut, and Barbara L. Chalfonte. 1990. The VideoWindow system in informal communications. ACM Press, 1–11.

12. Sofiane Gueddana and Nicolas Roussel. 2006. Pêle-Mêle, a Video Communication System Supporting a Variable Degree of Engagement. In *Proceedings of the 2006 20th Anniversary Conference on Computer Supported Cooperative Work (CSCW '06)*. ACM, New York, NY, USA, 423–426.

13. Keita Higuchi, Yinpeng Chen, Philip A. Chou, Zhengyou Zhang, and Zicheng Liu. 2015. ImmerseBoard: Immersive Telepresence Experience Using a Digital Whiteboard. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems (CHI '15)*. ACM, New York, NY, USA, 2383–2392.

14. Steven Houben, Søren Nielsen, Morten Esbensen, and Jakob E. Bardram. 2013. NooSphere: An Activity-centric Infrastructure for Distributed Interaction. In *Proceedings of the 12th International Conference on Mobile and Ubiquitous Multimedia (MUM '13)*. ACM, New York, NY, USA, Article 13, 10 pages.

15. Hiroshi Ishii and Minoru Kobayashi. 1992. ClearBoard: A Seamless Medium for Shared Drawing and Conversation with Eye Contact. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '92)*. ACM, New York, NY, USA, 525–532.

16. Rasmus Eskild Jensen. 2014. Why closely coupled work matters in global software development. In *Proceedings of the 18th International Conference on Supporting Group Work*. ACM, 24–34.

17. M. Kuechler and A. Kunz. 2006. HoloPort - A Device for Simultaneous Video and Data Conferencing Featuring Gaze Awareness. In *Virtual Reality Conference, 2006*. 81–88.

18. Brian J. McNely, Paul Gestwicki, Ann Burke, and Bridget Gelms. 2012. Articulating everyday actions: an activity theoretical approach to scrum. In *Proceedings of the 30th ACM international conference on Design of communication (SIGDOC '12)*. ACM, New York, NY, USA, 95–104.

19. M. Paasivaara, S. Durasiewicz, and C. Lassenius. 2008. Distributed Agile Development: Using Scrum in a Large Project. In *Global Software Engineering, 2008. ICGSE 2008. IEEE International Conference on*. 87 –95.

20. Lene Pries-Heje and Jan Pries-Heje. 2011. Why Scrum Works: A Case Study from an Agile Distributed Project in Denmark and India. In *Agile Conference (AGILE), 2011*. 20 –28.

21. N. Roussel, H. Evans, and H. Hansen. 2004. Proximity as an interface for video communication. *MultiMedia, IEEE* 11, 3 (July 2004), 12–16.

22. Jessica Rubart. 2014. A Cooperative Multitouch Scrum Task Board for Synchronous Face-to-Face Collaboration. In *Proceedings of the Ninth ACM International Conference on Interactive Tabletops and Surfaces (ITS '14)*. ACM, New York, NY, USA, 387–392.

23. H. Sharp, H. Robinson, J. Segal, and D. Furniss. 2006. The role of story cards and the wall in XP teams: a distributed cognition perspective. In *Agile Conference, 2006*. 75–86.

24. M. Stefik, D. G. Bobrow, G. Foster, S. Lanning, and D. Tatar. 1987. WYSIWIS Revised: Early Experiences with Multiuser Interfaces. *ACM Trans. Inf. Syst.* 5, 2 (April 1987), 147–167.

25. Kar-Han Tan, I. Robinson, R. Samadani, Bowon Lee, D. Gelb, A. Vorbau, B. Culbertson, and J. Apostolopoulos. 2009. ConnectBoard: A remote collaboration system that supports gaze-aware interaction and sharing. In *Multimedia Signal Processing, 2009. MMSP '09. IEEE International Workshop on*. 1–6.

26. John C. Tang, Chen Zhao, Xiang Cao, and Kori Inkpen. 2011. Your Time Zone or Mine?: A Study of Globally Time Zone-shifted Collaboration. In *Proceedings of the ACM 2011 Conference on Computer Supported Cooperative Work (CSCW '11)*. ACM, New York, NY, USA, 235–244.

27. Jakob Zillner, Christoph Rhemann, Shahram Izadi, and Michael Haller. 2014. 3D-board: A Whole-body Remote Collaborative Whiteboard. In *Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology (UIST '14)*. ACM, New York, NY, USA, 471–479.

dBoard 2

Title of Paper

Facilitating Distributed Standup Meetings through Blended Video Conferencing and Task Management Technology

Authors

Morten Esbensen, *Paolo Tell, Jacob B. Cholewa, Mathias K. Pedersen, Jakob E. Bardram*

In Preparation

Submitted, Revised and Resubmitted, Rejected at CSCW 2016

Abstract

A daily standup meeting in front of a Scrum board is the pivotal coordination, communication and awareness vehicle in Scrum. Since the standup meeting relies on collocation and since software development increasingly takes place in a distributed global setup, it becomes a challenge to conduct such meetings when the team is distributed across multiple locations. This paper first presents a study of such challenges in a small software company working across two sites in differ- ent countries, and then presents a study of how the company deployed a combined digital Scrum board and video conferencing tool (called dBoard) that supports distributed standup meetings. The three months field deployment revealed how the dBoard was able to improve the physical meeting setup and bridge the geographical gap by supporting distributed Scrum meetings, while at the same time pointing out significant challenges related to the physical and organizational configuration and set up of this kind of technology.

Facilitating Distributed Standup Meetings through Blended Video Conference and Taks Management Technology

Morten Esbensen, Paolo Tell, Jacob B. Cholewa, Mathias K. Pedersen & Jakob Bardram
IT University of Copenhagen
Rued Langgaards Vej 7, 2300 Copenhagen S
{mortenq,pate,jbec,mkin,bardram}@itu.dk

ABSTRACT
A daily standup meeting in front of a Scrum board is the pivotal coordination, communication and awareness vehicle in Scrum. Since the standup meeting relies on collocation and since software development increasingly takes place in a distributed global setup, it becomes a challenge to conduct such meetings when the team is distributed across multiple locations. This paper first presents a study of such challenges in a small software company working across two sites in different countries, and then presents a study of how the company deployed a combined digital Scrum board and video conferencing tool (called dBoard) that supports distributed standup meetings. The three months field deployment revealed how the dBoard was able to improve the physical meeting setup and bridge the geographical gap by supporting distributed Scrum meetings, while at the same time pointing out significant challenges related to the physical and organizational configuration and set up of this kind of technology.

Author Keywords
Scrum board; task board; Scrum; agile software development; distributed software development; field study; dBoard

ACM Classification Keywords
H.5.3 Group and organizational Interfaces: Computer Supported Cooperative Work

INTRODUCTION
Agile software development practices such as Scrum are increasingly adopted in many software companies and have proven to facilitate more efficient delivery of software according to budget, time and quality [24]. Central to the Scrum methodology is the daily Scrum meeting: a practice in which software developers meet in front of a physical Scrum board situated in the team's shared workspace. The Scrum board is typically a simple whiteboard with post-it notes that provides an overview of the pending, ongoing, and finished software development tasks (e.g., feature implementations, enhancements or bug fixes). The Scrum board is updated during the daily Scrum meeting and represents the pivotal mechanism for coordination, communication, and status awareness in a Scrum team. As such, the Scrum board has three main features; (i) it is a task management tool providing information about tasks and their status, (ii) it is a tool guiding communication during the daily Scrum meeting, and (iii) it works as an information radiator from which it is easy to get information about the progress of work during the work day.

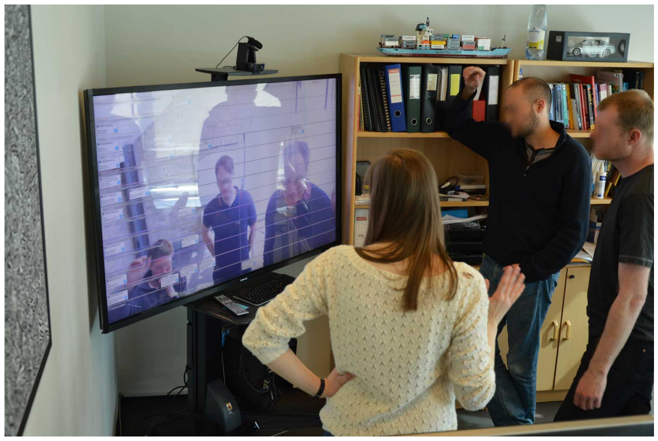


Figure 1. The dBoard in action during the Scrum meeting of a distributed team.

The features of the Scrum board rely heavily on collocation of the team members and the Scrum board: task management can only be done on the physical board in the team room, the daily Scrum meeting takes place in front of the board, and the team members must be collocated with the board for it to work as an information radiator. However, software development is increasingly being carried out across geographically distributed teams, and due to its success, Scrum is being applied in distributed arrangements as well [10]. In distributed software development setups, members of a Scrum team rely on computer mediated communication technologies to facilitate standup meetings as, rather then being collocated, the team is spread across different offices, countries, and potentially time zones. Studies of distributed Scrum has shown that videoconferencing is a core enabling technology – both large group videoconferencing as well as personal one-on-one video calls [11, 15]. However, videoconferencing technologies are not integrated with the other tools required during the meetings. In fact, videoconferencing often happens in connection with a shared task [4], and there is a need for systems that integrate videoconferencing and with the tools used during meetings [22].

This paper presents a longitudinal study of the Scrum work practices of a distributed software team working across two sites in Denmark and Germany, while employing different technologies. The study is divided into two main parts: the first part focused on obtaining a detailed understanding of the challenges arising when moving from a collocated Scrum setup to a distributed one; the second part revolved around a field deployment of the dBoard [1], which was deployed to support the Scrum practice across the two sites. The purpose of this study was twofold as it focused on the support provided by the dBoard in both the collocated setup as well as the distributed.

This study of distributed Scrum revealed a set of challenges of which some were addressed by the dBoard. For example, a traditional Scrum board is easy to use and access—it's just a whiteboard—whereas setting up remote video links and task management software is rather cumbersome and often disrupts the easy flow of a Scrum meeting. In this respect, the dBoard was much more straightforward deploy and use. However, the study also revealed a set of more fundamental challenges which were not addressed by the dBoard. For example, the organizational and physical set up of the Scrum processes and the board is important for the Scrum process to work. And if a tool like the dBoard is not physically and organizationally embedded into the offices and work practices of the team, it cannot work properly.

BACKGROUND AND RELATED WORK

Scrum is a software development framework that suggests a number of practices to structure work, team and meetings [19]. One central meeting in the framework is the Scrum meeting, also referred to as the 'daily Scrum' or 'standup meeting'. The Scrum meeting is a short meeting usually held every day in the morning in which participants are encouraged to stand up to prevent meetings prolongation. The purpose of the meeting is to get a daily status on the sprint and the meeting follows a simple pattern. All team members present three things: what they have been doing since the last meeting, what they will be working on and whether they see any impediments to their work. The Scrum meeting is not only effective for giving awareness of how the team is progressing but also helps in both the day-to-day coordination of the Scrum team [16] and in the co-articulation, i.e., the process of articulating ones actions in response to a team members daily report [13].

During the meetings it is customary for many teams to use a Scrum board. Containing all the tasks in a sprint (those that have been completed, those in progress and those where work still has not begun), the Scrum board provides an overview of the sprint status and thus acts as a tool facilitating the meeting. As such, the Scrum board helps reducing articulation work during a Scrum meeting [16].

In distributed Agile Scrum where a Scrum team is partly distributed across several locations, a core and recurring question is how to set up and use the Scrum board. The traditional Scrum board as a white board with post-it notes can only work in a distributed arrangement if the board is maintained and manually synchronized at all locations; alternatively, digital versions of a Scrum board can be used to automatically synchronize the Scrum board across distant sites. Video conferencing systems can also be used to facilitate Scrum meetings across distributed sites, however, even with video technology at hand, the lack of easy access to a collaborative Scrum board can cause a team to abandon the Scrum board altogether [6].

Scrum Boards

The Scrum board has received some focus from academia research and has been identified as an important coordination mechanism in terms of handling articulation work [3, 16] and as a learning tool in Scrum [13]. Several products such as Jira[1], TargetProcess[2], and Microsoft Team Foundation Server[3] offer digital Scrum boards and are often the choices selected in companies interested in digital Scrum boards. Such digital Scrum boards have been studied and it has been suggested that digital Scrum boards can have drawbacks compared to physical ones [20], however, a recent study also showed that digital card-walls such as Jira can provide support for distributed teams but also point out that there still does not exist a tool that supports distributed Agile meetings with the wall as a central artifact [9]. While few studies have suggested digitizing the Scrum board [17, 18] or, closely related, utilizing tabletops for Agile planning [8], in terms of field deployments of digital Scrum boards this is still largely an under-explored area. However, in a study of 'articulation spaces', Boden et. al. demonstrated willingness to adopt digital tools as means of Scrum boards in collocated software development [3].

In summary, the Scrum board has been identified as an important artifact in the Scrum development framework yet it still remains unclear how to provide support for daily Scrum meeting for distributed Scrum teams that include such a tool. Combined with the importance of videoconferencing for Scrum teams, we see an opportunity to explore an area in which videoconferencing is combined with the digital Scrum board designed for distributed teams.

CASE AND RESEARCH METHODS

Alpha is a small Danish software company that specializes in developing stowage-planning optimization software for the shipping industry. In 2013 Alpha entered a partnership with Beta—a German software provider that develops several products for the shipping industry. In 2014 a 10 year cooperation began in which Alpha was made responsible for the development of one of Beta's products. This contract resulted

[1]https://www.atlassian.com/software/jira
[2]http://www.targetprocess.com/
[3]https://msdn.microsoft.com/en-us/vstudio/ff637362.aspx

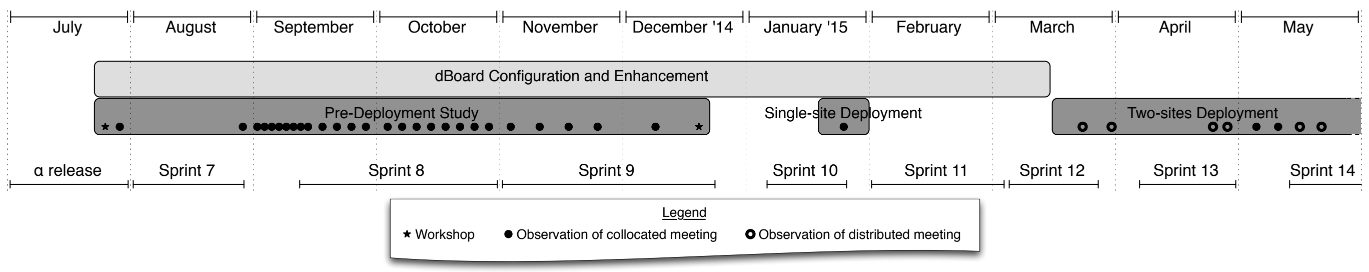


Figure 2. Overall outline of the study showing the four main phases; (i) pre-deployment study of the Scrum practices at Alpha; (ii) the configuration of the dBoard; (iii) the collocated deployment of the dBoard at Alpha; and (iv) the distributed deployment at both Alpha and Beta.

in a setup with developers located in both Denmark and Germany. The Danish part of the team consists of seven practitioners (i.e., five developers, the Danish project manager, and the Scrum master). The German part consists of four people (i.e., two developers, the German project manager, and one tester).

Figure 2 outlines the main phases of the study we conducted at Alpha and Beta, and hereafter we present relevant details and main the focus of them. The *pre-deployment* phase focused on how Alpha appropriated the Scrum practices for their local coordination and communication as well as how the Scrum framework facilitated their cooperation with Beta. This study took place over a period of four and a half months involving in total 27 site visits.

In parallel with the pre-deployment study, the dBoard system was configured and enhanced. The core dBoard system was at this point designed [1], but to clarify how the dBoard would fit the technical setup and work practices of Alpha and Beta, we conducted a design workshop in late July. The main requirements that emerged from this workshop were that the dBoard should integrate to their task management system (Microsoft Team Foundation Server (TFS)) and that some additional features related to more detailed task information and task sorting should be included. Relevant findings from the ongoing pre-deployment study were also incorporated into tailoring the dBoard for later deployment in this setup. In December, when the pre-deployment study was concluded, a second workshop was conducted to verify the configuration and adaptation of the dBoard. This was done using a scenario-based evaluation in which two major scenarios were enacted by team members from Alpha and Beta. In particular, we asked participants to enact a standup meeting and an ad-hoc meeting both performed using two separate location to mimic their distributed arrangement.

At the final workshop, it was decided to separate the deployment into two phases: a *single-site deployment* only including Alpha and a *two-sites deployment* with both Alpha and Beta. The purpose of the single-site deployment was to evaluate how well the dBoard was supporting Scrum-based task management, including its integration to the Microsoft Team Foundation Server (TFS) system. This initial installation turned out to be a wise approach, since changes and updates to the TFS setup had to be accommodated. The dBoard was deployed in the office of Alpha for two weeks in January '15 and, in March, the dBoard system was deployed at both Alpha in and Beta in Germany with the activation of the collaborative features of the system, which were put to test. The focus of this final phase was to observe how the dBoard system helped bridging the geographical gap between Alpha and Beta .

In total, the dBoard was deployed for three months, during which we made 9 site visits: 3 involving only the Danish team and 6 including both teams. The rather small number of site visits as compered to the pre-deployment study was influenced by several factors, which will be described later in the paper. All observations were performed by one person with the exception of two out of the 6 distributed standup meetings, which were observed concurrently by one observer at each side (i.e., one observing the team in Denmark and one observing the one in Germany).

We used empirical qualitative methods throughout our investigations, in particular participants observation and on-site interviews. Extensive note taking and photography was done during site visits, and all observed standup meetings were video recorded. During subsequent analysis, all observation notes were coded independently by two researchers following a grounded theory-inspired approach in iterative coding sessions that allowed concepts to emerge, video recordings were also reviewed and annotated, and important sequences were identified and studied in details by two researchers.

SCRUM MEETINGS IN APLHA

In early 2014, Alpha decided to switch to a Scrum inspired development process and they now follow many of the practices recommended by Scrum. Their development is arranged in sprints, planned during sprint planning meetings. Each sprint consists of a number of user stories broken down into tasks that are assigned to the different developers. The tasks and stories are managed using TFS. The team has also assigned a Scrum master and uses Scrum meeting throughout the sprint.

The Scrum guide suggests Scrum meetings to be held daily [19], however, the Danish team had settled on a model where meetings are arranged on an ad-hoc basis. Usually the meetings were held twice a week; however, when close to a delivery, the team had even less frequent meetings. During our four months observation we observed 27 of these meetings. The meetings were held at 10 AM and were initiated by the Scrum master or the project manager informing that

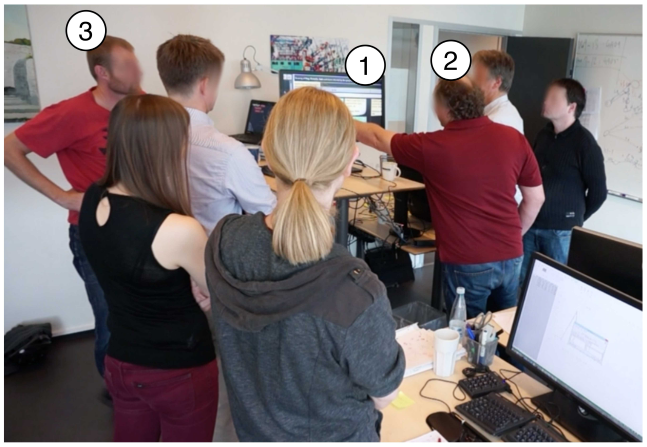


Figure 3. A collocated Scrum meeting at Alpha. With the Scrum board turned so the active person can point towards it (2) another person is not able to see it (3).

the daily meeting was about to start. The team then assembled around the Scrum master's desk which was raised to accommodate standing up (see Figure 3). The Scrum master launched the Scrum board plugin of TFS on his computer. This plugin shows all user stories, tasks and bugs from TFS as post-it notes sorted into rows by the user assigned to them. Meetings were organized as suggested in the Scrum guide; however, the team often engaged in longer discussions during these meetings to address implementation details of the tasks which caused meetings to prolong more than the 15 minutes usually suggested [19].

Results

While the team did not have meetings daily but only twice a week, and even though they included German developers only when visiting the Danish office, it was clear from our observations that the meetings helped the team coordinate their work, gave awareness of the state of the work and were a place for rich knowledge sharing.

The standup meeting provided a fora for the team to share progress and to foster continuous coherent understanding of the project status, which allowed the team members to coordinate, focus, and align their efforts. We observed that the meetings also allowed details to emerge often resulting in prompt identification of issues followed by immediate reactions expressed through brief orders from the project manager (e.g., *"make a bug!"*). Moreover, close to the end of a sprint, the overview provided by the Scrum board supported quick planning and reorganization of tasks across sprints. For instance, between sprint 7 and 8, the Scrum master in one occasion and the project manager in another requested all team members to act on the non-completed tasks by either completing them quickly or by moving them to the next sprint.

The standup meetings facilitated general awareness within the team. Often the project manager would ask questions like *"Is this closed?"* or *"What is that bug?"*, which would draw the team members attention to these tasks on the board and help build an awareness of the status of work among the team members. Furthermore, during meetings it was not uncommon that simple statements from one developer would catch the attention of another team member, who could then add information about the task, hence, adding to the shared awareness of the status of work in the team.

We also observed how the meetings were used extensively for knowledge sharing among the developers. In almost all meetings, discussions involving the specifics of an implementation task emerged frequently. After a person had taken her turn in the meeting explaining what she was working on, the team would initiate a discussion related to these tasks. While Scrum suggests longer discussions to be taken outside daily meetings, we observed how the team preferred to take these during the Scrum meetings even if it meant prolonging them and potentially, but not necessarily, loosing the attention of some participants. The meetings thus facilitated a large amount of knowledge sharing between the developers, which often resulted in clarification of very specific details regarding the implementation.

In line with other studies [24], our observations clearly showed that the Alpha team benefited from conducting regular Scrum meetings. But we also observed and identified a set of challenges related to: (i) the distribution of meetings, (ii) the use of the task management and Scrum software tools, and (iii) the way the meeting was done in terms of physical space and place.

No Distributed Meetings

We observed that meetings primarily included only the Danish staff. Only if team members from Germany were visiting the Danish offices, they would participate in the Scrum meetings. This absence of the German team in the daily meeting caused problems and confusion. In one instance, we observed how a task that was marked as 'Done' by a German team member caused confusion, since the Danish team doubted if this task had really been completed. With that developer not present, it was not possible to clarify this doubt until after the meeting. We also observed how the progress of team members in Germany was discussed during the daily meeting. With no team members present from Germany, it was hard for the Danish team to get a complete overview of the status of the sprint and what the German developers were working on. This resulted often in long phone calls between either the two project managers or two specific developers that, due to the task at hand, had to share part of the code base.

Setup Cost of Scrum Board Technology

A specific Scrum board plugin for TFS was used to facilitate the Scrum meetings at Alpha. However, there were several challenges and shortcomings in this tool. First of all, in contrast to a non-digital Scrum board—which has no setup cost—there was a significant overhead of establishing an 'adequate' view of the work using the TFS tool. Most of the time in fact, the Scrum master had to abandon what he was doing on his computer, switch to the Scrum board plugin, and set it up. And, if for example, he was compiling code on his computer, the team would have had to wait before they could launch the Scrum board. In more than one case, these 'setup costs' caused the meetings to be either delayed, due to

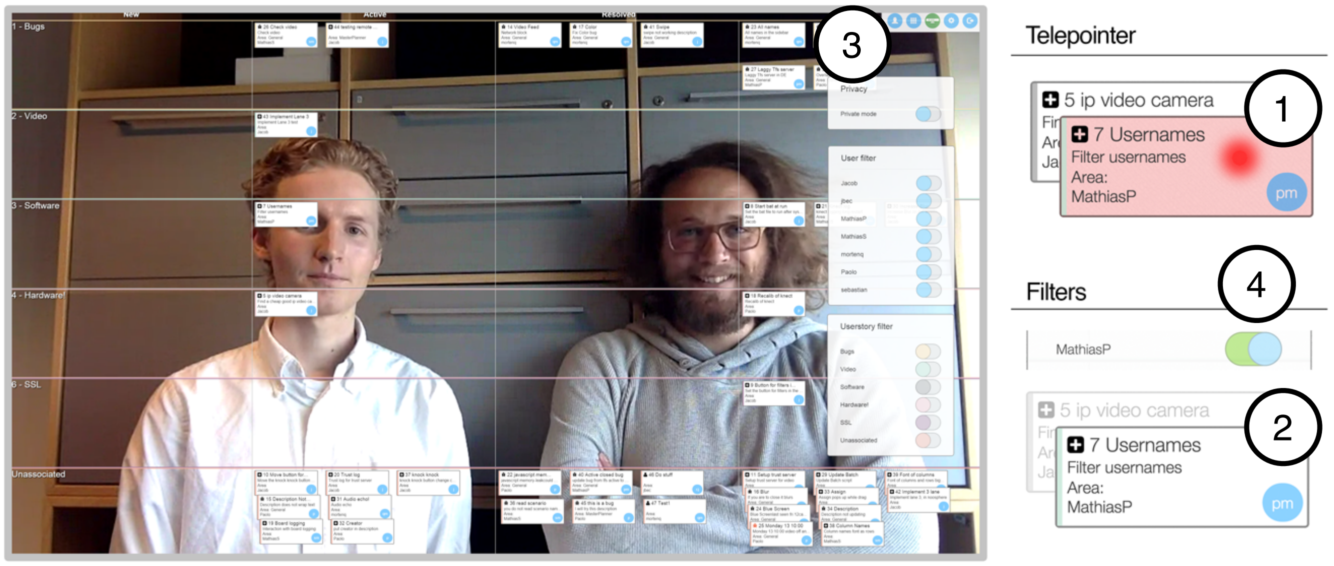


Figure 4. The UI of the dBoard with telepointers and highlight (1), tasks (1 & 2), menu (3) and filters (4).

the extra time spent on finding alternative means (e.g., other workstation to run the TFS plugin), or conducted without the use of the Scrum board.

User stories were typically broken down into separate tasks to be worked on by team members. In the TFS Scrum board, both user stories and tasks were displayed side-by-side using color coding to indicate whether it was a task, a user story or a bug. However, the team had problems relating these together, and we observed how team members had to open the details of a user story to understand which tasks were related to that story. The knowledge of the relationship between tasks and user stories is important as, when user stories are broken down into tasks, the progress of implementing a user story can be tracked by the number of related tasks completed.

Inappropriate Space and Place

The physical space and location of the Scrum meeting in the office of Alpha also posed signifiant challenges. The meetings were conducted by using the TFS Scrum board plugin on the Scrum master's computer. This resulted in a meeting setup as shown in Figure 3, in which the team members would gather around the Scrum master's desk and monitor (1). There are several challenges associated with this setup: the space is very limited and does not leave space for team members to stand in front of the display; the display is relatively small and hard (or even impossible) to read from a distance; and, only one person can interact with the board at a time. Figure 3 shows a typical meeting situation: while one person (2) is speaking and gesturing towards the board, another (3) is unable to see the board. To accommodate this problem, the team sometime used a magnification feature to zoom in on the board, but this just made it harder to navigate in the system as it required both scrolling and panning. We also observed how the Scrum master had to turn the display toward the person speaking for that person to see the information on it (e.g., Figure 3 (3)).

In summary, our observations of the daily meeting practices of Alpha provided insights into the benefits of such meetings, while also revealing a set of challenges the team faced. The meetings provided the team with a platform for rich coordination, awareness and knowledge sharing. However, there were also significant challenges encountered regularly during these meetings, specifically related to: the absence of German team members, the setup cost of the Scrum board tool and the constrained physical setup of the meetings in the office space.

DBOARD

The 'Distributed Scrum Board' (dBoard) was designed to support distributed software teams in their standup meetings. The dBoard hardware setup is shown in Figure 5 and the software UI is shown in Figure 4. Based on studies of Scrum practices and daily standup meetings at Alpha and other software companies, the dBoard was designed to integrate five core design features into one solution:

- *Virtual Window* – The dBoard should provide a 'virtual window' between two remote sites thereby enabling seamless communication between two distributed teams.
- *Scrum Board* – The dBoard should support task management (i.e., user stories) in a Scrum fashion by supporting direct manipulation of 'virtual post-it notes' on a grid-like task board.
- *Proximity-based Interaction* – The dBoard should adapt to the environment in which it is situated; in particular, it should adapt to whether people are in front of it or not and to how they are using it.
- *Minimal Setup Costs* – The dBoard should allow people to easily establish meetings with minimal setup costs.

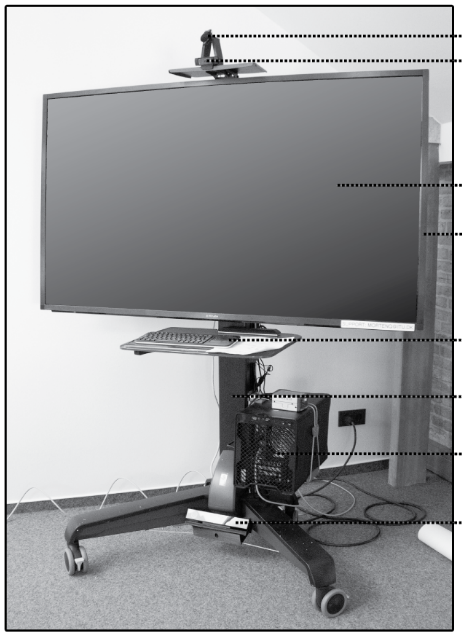


Figure 5. The hardware setup of one dBoard.

- *Tool Integration* – The dBoard should integrate to relevant digital software engineering tools like issue tracking and Agile project management used in the Scrum practices.

This section provides an overview of the dBoard features. A more detailed description of the technical architecture and the UX design of the dBoard is presented elsewhere [1].

Virtual Window

As shown in Figure 5 and 4, the dBoard is equipped with a 65" display with a 3840 x 2160 resolution, an HD camera and multitouch overlay in combination with a high-quality noise-reducing microphone and loud speakers. This configuration allow to create a 'virtual window' [7] in which two sites can see into each others' offices and engage in conversations in front of the 'window'. Touch events on the screen are relayed as 'telepointers' shown as red dots on both screens (Figure 4 (1)).

Scrum Board

A Scrum board is overlaid on top of the video on the dBoard screen using a content-on-people approach [12, 14] (Figure 4). The Scrum board applies a traditional grid-based layout with user stories represented as rows and development states represented as columns. Tasks are visualized as small 'virtual post-it notes'—small blocks with minimal information on them (see Figure 4 (2)). The position of a task on the board represents the user story it belongs to (row) and its current state (column). The Scrum board is synchronized across two connected dBoards to show the exact same view in both sites.

Tasks on the board are interactive: a task can be dragged anywhere on the board to update its status; a task can be assigned to another developer just by tapping the name on the task and by selecting a new one from the popup list and, a task-dedicated side panel can be revealed on the right-hand side of the board by using a 'tap-and-hold' gesture on the task.

The Scrum board also contains a number of ways to sort and filter the information presented on the board. From the menu (Figure 4 (3)) rows that normally show user stories, can be toggled to organize tasks by users instead. With this feature, it is easy to get an overview of which tasks are assigned to each developer. The dBoard also implements filtering for users and user stories. If a filter is selected, all tasks not belonging to that user or user story are dimmed out. Figure 4 (2) shows the filtering of tasks based on a user and the user filter handle respectively. Finally, the dBoard also supports two different forms of automatic sorting of tasks to re-organize their position and visibility by placing them into the correct row-column cell as well as by stacking them to maximize visibility.

Since the Scrum board runs on top of the virtual window and hence runs in a collaborative mode, the two boards are implementing a WYSIWIG approach [21]. Task movements on one board are updated in real time on the other board. Furthermore, just like telepointers are marked with red dots, a task that is touched and moved is highlighted in red, as shown in Figure 4 (1).

Proximity-based Interaction

By using the Kinect mounted beneath the screen, the dBoard automatically adjusts the video and audio based on the presence of people in front of it. The dBoard has three specific modes of presence: people present in both sides, people present in only one side and people not present in either side. If no people are present at either side, the audio is turned off and video is blurred to the extent that movement is still identifiable but recognizing people is hard. In this mode, the dBoard is designed to work as a Porthole-style awareness window [5]. If people are in front of the board in only one office, the video is slightly less blurred on both dBoard's and the audio is disabled. Finally, if people are in front of the dBoard in both offices, the full video and audio is turned on. The dBoard also has a 'privacy button' that—instead of blurring the video—removes the video entirely if no people are in front of the board. This allows the dBoard to be used only as a passive Scrum board without the potential distractions of the background video feed.

Minimal Setup Costs

The dBoard is designed to conduct daily standup meetings and other meetings in an easy manner without any overhead in terms of setting up video equipment and/or software, or to establish communication links. The dBoard is an alway-on, dedicated tool for meetings, and meetings are started (and stopped) automatically when people gather in front of the board. The dBoard application is automatically started and the video and audio link established when the computer is turned on or restarted. All components of the system are mounted on a stand with wheels making the dBoard easy to move around and position in an appropriate place in the office. Hence, there is no need for dedicated 'video conference rooms' and the board can be moved around and placed in different locations of an office just as a physical Scrum board (e.g., on a white board, a wall or on a window).

While the 65", ultra-high resolution (i.e., 3840 x 2160) monitor used for the dBoard does provide a large amount on screen real-estate, it is still smaller in size than a white board typically used for a physical Scrum board. Indeed, this limits the amount of tasks that can be displayed on the dBoard, however, we chose this approach over multiple screens as it allowed us to mount all components on an easy to move setup that reduces the setup costs of the dBoard.

Finally, the dBoard also implements a 'knock-knock' gesture, which allows a user to knock on the screen (like knocking on a window). This knocking is relayed to the dBoard which will produce a knocking sound signaling that someone at the other end is requesting attention. This allows for very easy and ad-hoc meeting initiation.

Tool Integration

The dBoard is designed to integrate with issue tracking systems for software engineering. For the deployment at Alpha, the issue tracking system integrated was TFS. This is a two-way integration: all tasks including their information and status are synchronized with the dBoard and when a task is changed in the dBoard—for example by moving a task from one status column to another—the information is synchronized back to TFS. This two-way synchronization, however, puts a few limitations on the use of the dBoard, of which the most important one is that TFS puts constraints on status changes. The TFS workflow used by Alpha and Beta would not allow tasks to move from the 'New' state to the 'Resolved' state. Therefore, if a user moved a task on the dBoard from 'New' to 'Resolved', the state change would not be allowed by TFS and the task post-it note would be moved back to the 'New' state on the board. This was in conflict with the original design goal of allowing any movement of the task post-it notes on the dBoard, but since Alpha insisted on having TFS as the 'ground truth', we had to compromise.

DEPLOYMENT

For the evaluation of a combined videoconferencing and scrum board tool, the dBoard was deployed in two phases as shown in Figure 2. In the first phase, we deployed the dBoard at Alpha and, in the second phase, at both Alpha and Beta.

One-site Deployment

The one-site deployment took place at Alpha, and focused on the use of the dBoard as a Scrum board for collocated use. Therefore, the collaborative features, including the video and proximity-based interaction, were disabled, turning the dBoard into a digital and interactive Scrum board. This deployment strategy was chosen, as it gave us the opportunity to evaluate and test the hardware and software in one location before moving to a distributed deployment.

Setting

The dBoard was deployed at Alpha for a period of two weeks. As the dBoard was designed to be an 'information radiator' (like a regular Scrum board), the original intention was to

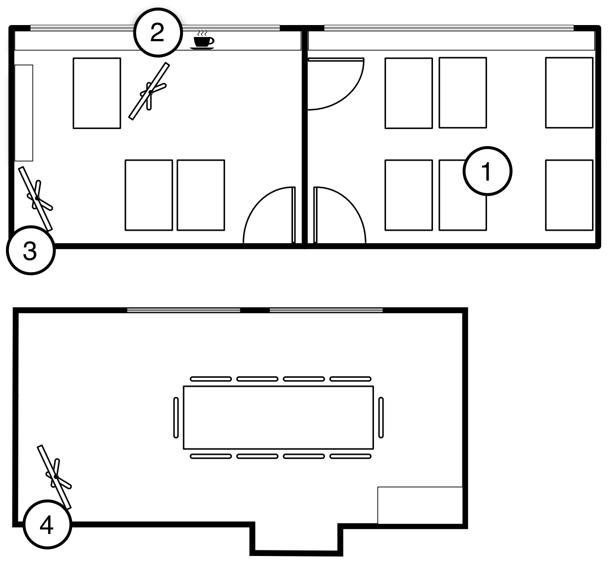


Figure 6. The office plan of Alpha (top) and the meeting room of Beta (bottom)—(1) the location of the Scrum master's desk; (2) the locations of the dBoard at the one-site deployment; (3) & (4) the location of the dBoard during the two-site deployment.

place it in the shared office space of the Scrum team. However, due to space limitations the dBoard was not placed in the team office, but in the room close to the coffee machine (Figure 6(2)).

Results

During this phase, Alpha only conducted one meeting using the dBoard, but during the two-site deployment phase, additional two meeting involving only the Danish team were held. Hence, in total three meetings using the dBoard in a collocated Scrum meeting at Alpha were conducted; there are the basis for this analysis. The meetings with the dBoard were conducted in a similar way to the ones the team members would perform with the TFS plugin: at 10 AM the team would assemble and explain what they had been working on since the last meeting, what they were currently working on, and whether they foresaw impediments. During our observations, we saw how the dBoard supported the team in their Scrum meeting practices. The high-resolution large screen provided plenty of space to display all tasks in the sprint, and the area used for the meetings had sufficient space for all developers (see Figure 1). The relationship between user stories and tasks was also clear from the dBoard visualization. As such, it was evident that the prior challenges of poor task visualization and lack of space and visibility were addressed with the introduction of the dBoard. Moreover, since the dBoard was a dedicated 'always-on' public display, there was limited setup cost associated with it, and the team could simply walk up to it and start the meeting.

For the first meeting, the team used the different features of the dBoard extensively. The team made extensive use of the side panel presenting detailed information about a specific task or user story while explaining finished and ongoing work. One the one hand, the usage of this panel quickly diminished and, for the later collocated meetings as well as the distributed meetings, it was rarely used. This was surprising to us because much of the information on the side panel was requested by the team during the workshops. On the other hand, the user-filter feature of the dBoard extensively used throughout the meetings for turn-taking might explain such reduced use. As tickets were not anymore arranged by users but by user story, the task context (i.e., which task belonged to which user story), which was one of the main reasons to investigate the detailed information on the TFS, was now always clear. Usually the project manager would control turn-taking among the team members, and use the filter to display only the tasks relevant for the team member talking. The fact that the user filter was widely used during the meetings also meant that the turn-order was decided by the order of this menu.

What was also interesting, however, was that the teams did not interact much with tasks on the dBoard during the meetings. As the dBoard synchronizes its information with the TFS system used by the companies to handle the development process, the board often was in the right configuration even before the meetings started. This meant that few changes (such as opening, closing or re-assigning tasks) had to be made during the meeting. We did however, observe a few instances where such changes were made. Despite the dBoard's design for such activities, the team followed the practices that they used during the collocated meetings in which the board was used primarily as a tool for guiding the Scrum meeting and for a platform to initiate knowledge sharing with occasional changes being made to the tasks presented on the board.

Finally, due to the fact that the dBoard was placed in the 'wrong' location–namely in the office with the coffee machine rather than in the team's office–we observed that it did not work as an information radiator and did not provide the team with any shared awareness of the status of work. In fact, the dBoard was turned off while not used during the meetings, which implied that it was standing idle most of the time.

Two-site Deployment

In the second phase, the dBoard was deployed both at Alpha and Beta. The focus of this second phase was to study how the dBoard would support a distributed Scrum team.

Setting

During this phase, all features of the dBoard were enabled. At Alpha, the dBoard was moved to the corner of the office (Figure 6(3)), since the desk behind it was used by a new employee and was blocking the line of sight to the other room. At Beta, the plan was to deploy the dBoard in the team's space to allow for easy access to the system, for easy ad-hoc meetings through the knock-knock feature and to foster mutual awareness by having the dBoard alway-on and working as an information radiator. However, due to space limitations in the team space at Beta, the project manager insisted on the dBoard to be moved into a meeting room, which was even situated on a different floor than the team space (Figure 6(4)).

Results

With the dBoard positioned in the meeting room, the features dedicated to awareness and ad-hoc meetings were clearly not used. The dBoard was designed to function as a meeting support tool as well as an information radiator with a graceful way to switch between these two modes. However the placement of the boards turned them into a meeting support tool for pre-planned meetings. Even though the placement at Alpha was not optimal either, they would still be able to see the display on a regular basis (e.g., when getting coffee) and to hear any knock-knock being initiated from Beta. But clearly–as a collaborative tool–the placement at Beta also affected the Danish team; with no opportunity to engage with–or be engaged by–the German team outside pre-planned meetings, the two teams often turned off the dBoard after a meeting.

Analyzing specifically the Scrum meetings conducted via the dBoard, it was evident that: first, the system enabled the two teams to engage in distributed meetings, which was identified as an issue during the pre-deployment study; and second, the blended videoconferencing and Scrum board features provided good support for the team during the meetings. As an example, the German project manager during the standup meeting slot of a Danish developer noticed a task about which he had a concern. Used to the previous practice without the dBoard, he stated thathe wanted to discuss it over the phone. However, the Danish developer were able to further discuss the task which instantly stimulated the German project manager to propose that the task should be assigned to a German developer he believed to be more suited for solving it, hence, removing the need for a subsequent phone call. The German developer, also present to the meeting, agreed, and the task was immediately re-assigned to the new person. This snippet, which lasted roughly 3 minutes, clearly showcases how the dBoard provided support to communication between the two locations, improved the awareness of each others' work, facilitated the identification of potential risks, and allowed quick reactive ad-hoc coordination.

The video also facilitated other interesting interactions for which the system was not designed. Once, the Danish office received a list of bugs from the client via email in which the German team was not cc'ed; in this occasion, they printed the screenshots of these bugs and showed these printouts to the German team through the dBoard camera. Even though we did not design for bringing in external documents to the dBoard, the team found a way around this, and these printouts–eight in total–were discussed in detail in the meeting.

We also observed how the rich knowledge sharing activities seen during the collocated meetings continued to be present during the distributed meetings but with the German team members included. As with the collocated meetings, often these sessions were initiated based on talks about a specific task on the Scrum board. This was interesting as the German project manager is a domain expert in the field of shipping and cargo vessels and during most of the meetings, he was able to share knowledge about the specifics of container vessels–information important to the Danish developers.

DISCUSSION

The dBoard provided the team at Alpha and Beta with a tool for facilitating both collocated and distributed Scrum meetings. As opposed to before, the German team was now present in almost all Scrum meetings, and the dBoard helped alleviating some of the problems they had experienced before. In particular, the combination of video and task management provided the team with a physical workspace to communicate and discuss their work while orienting themselves towards the work at hand. And, the cost of setting up meetings was reduced significantly, which lead to more efficient meetings. We also observed how the dBoard supported the articulation work of the collaboration, which has been pointed out to be core to the success of Scrum [16]. As opposed to before, the team was now able to coordinate and discuss their work during the Scrum meetings and the need for contacting remote team members after meetings was reduced.

Our study also confirmed how the dBoard was able to lower the setup costs of the Scrum meetings but also revealed a set of new problems and challenges related to the introduction of the new technology.

Digitalization and Synchronization

The synchronization of the dBoard with TFS caused interesting usage patterns. In particular, we observed how the teams interacted very little with the board as it in most cases was in its correct state even before a meeting had started. The team extensively planned, carried out and updated their progress on tasks and user stories from their personal computers and little work on the board was needed during meetings. This highlights an important consideration regarding digital Scrum boards. The synchronization of data with other systems is a valuable benefit of a digital system that, however, reduces the need for explicit interactions with the board when compared with a physical board. These findings echo previous research on the advantages of drawbacks of physical and digital tools [20, 23].

Another point that became apparent was how the workflow structure of TFS influenced the usage of the dBoard during the meetings. While we designed the board so that a tasks could be moved freely on the board, we also wanted to ensure that the board always reflected the state of the underlying TFS. We therefore had to disallow certain state changes that were not valid in the TFS workflow. This meant that it was not allowed for some types of tasks to transition from the 'Active' to the 'Resolved' state and trying to perform such transitions in the board would force the task to be repositioned back to its previous state. This behaviour was observed in a few instances, and it consistently generated confusion for users. This highlights another tradeoff of using a digital Scrum board as compared to a traditional physical one. On physical Scrum boards tasks can be placed anywhere and it is up to the team to ensure that the board is in a state that conforms to the teams' practices. On digital boards where workflow constraints from underlying systems are applied, tasks have to follow these workflows which eventually can cause confusion if users are unaware of these constraints when interacting with the board. Therefore, designers of interactive Scrum boards in which underlying constraints are applied should take into consideration making explicit to users what kinds of state transitions are allowed or disallowed.

Physical Location

The physical placement of the dBoard in the meeting room at the offices of Beta severely influenced its usage and the awareness and ad-hoc meeting features were not used. The decision to put the dBoard in the meeting room rather than in the workspace of the developers was taken by the German project manager, and he explained that there were two reasons for not having the dBoard in the office. First, the open offices at Beta were occupied by people working on different projects—not just the four people working with Alpha. The project manager did not want to disturb these other people with meetings at the board when they were working. Second, the dBoard is more bulky than a physical board due to the stand and therefore occupies more space than a physical board that can be hanged on a wall. The German offices were already quite occupied and the project manager argued that there was little room to place the dBoard.

With the dBoard deployed in a meeting room rather than in the office space, the board lost its effect of serving as an information radiator and was not used for any ad-hoc meetings even though such features were designed. This highlights a challenge in the design of technologies that aim to support different collaborative activities. The physical location of such technologies is crucial to their adoption and usage. The dBoard was designed to integrate videoconferencing and Scrum board into one tool as such integration is needed [22]. However, in doing so, the German team decided that partly due to its videoconferencing features, the dBoard was to be placed in a meeting room and not in the office space. In designing tools that blend together videoconferencing and task-specific application, it is important to consider how such tools will be viewed and in turn deployed by the intended users.

Interestingly, in our efforts to create a tool with as low setup costs as possible, we also created a tool that requires some space to setup. The stand of the dBoard (see Figure 5) takes up more space in an office compared to a wall-mounted screen, however, we chose this to lower overall the setup cost. Both the German and the Danish side chose to place the dBoard in rooms not ideal for a Scrum board—in Denmark in the office with fewest developers and in Germany in a dedicated meeting room. This highlights another tension in the design: the movable stand reduced the setup costs of the system but enforced constraints on the location in which it could be deployed eventually resulting in the teams' decisions to position them in sub-optimal locations.

Meeting Frequency

During the time of the deployment, we saw a decrease in the frequency of Scrum meetings. While this decrease can't be attributed solely to the introduction of the dBoard, we did observe how the team at Alpha decided not to have meetings when the team at Beta was not available. This is an interesting observation as it demonstrates how trying to introduce technologies to facilitate closer collaboration across distances can cause a change in the existing work patterns. Transitioning to distributed work can significantly complicate the work of setting up meetings [2] and, despite our effort to design the dBoard to be easy to use for meetings, the teams' usage of Scrum affected the use of the dBoard.

This finding highlights the importance of carefully both implementing Scrum as a framework and selecting the technologies for enabling the collaboration. With the dBoard, the teams at Alpha and Beta were able to carry out distributed Scrum meetings in front of an interactive Scrum board – an activity which was hard to perform before; however, despite the deployment of this tool, they did not have daily meetings as suggested by Scrum. An efficient implementation of Scrum in a distributed context requires not only the appropriate technology but also the ability and willingness to implement the processes of Scrum. While this implementation of Scrum has to be tailored to the challenges of working distributed [11] it still needs to be taken seriously.

Organizational and Physical Embeddedness

We categorize the aforementioned problems and challenges that we encountered during the deployment of the dBoard into the degree of organizational and physical embeddedness of a technology. These categories affect the adoption of such a technology.

With organizational embeddedness, we refer to the degree to which an organization's practices align with technology. The dBoard was designed to facilitate distributed daily Scrum meetings with minimal setup costs, but in our study we observed how the team did not have such frequent meetings which was partly caused by the ad-hoc meeting structure that the team had adopted. We also observed how few interactions with tasks happened during the meetings as the status of sprints were kept up to date on a daily basis by the developers.

The physical embeddedness refers to the physical embodiment of a technology. What we observed here was that while the dBoard in Alpha was placed in the everyday workspace, at Beta the board had been placed in a large meeting room. This placement of the dBoard effectively removed both the dBoard's function of being an information radiator as well as the one of being a tool supporting ad-hoc meetings as it was only visible during pre-planned meetings. The physical embeddedness does not only concern the physical surroundings in the environment but also the physical form of the technology itself. In particular, a combination of the office layout of Beta and the large stand of the dBoard influenced the German team decision to place it in a meeting room.

CONCLUSION

The Scrum board is an important tool in the Scrum development framework, however, its collocated nature causes problems for adopting the board into distributed software development. In this paper, we presented two studies of distributed Scrum meetings. In the first study, we reported on an observational study of the Scrum meeting practices of a small Scrum team. The study showed that the Scrum meeting provided a platform for coordination, awareness and knowledge sharing but also revealed problems related to the absence of remote team members, the digital Scrum board tool and the space and place of meetings. Based on this study we deployed the dBoard—a digital Scrum board and videoconferencing tool—and studied how the team conducted their scrum meetings.

This second study showed that the new technology was able to provide support for distributed Scrum team in conducting Scrum meetings but also pointed out new challenges. First, we observed how the synchronization of the dBoard with an underlying task tracking system caused confusion when interacting with the board as the task tracking system enforced constraints on the dBoard that were not clear to the users. Second, we argue that the design of collaborative system intended to support different activities might eventually be used for only one single purpose. The dBoard was designed to support many practices around a Scrum board (i.e., information radiator, ad-hoc meetings and standup meetings) but was used only for planned meetings. Lastly, we stressed that both appropriate technologies as well as serious implementation of practices are required for a successful adoption of Scrum, and a thorough dedication to one of the two cannot resolve problems in the other.

ACKNOWLEDGMENTS

Anonymized for review

REFERENCES

1. Anonymized. 2015. The dBoard: a Digital Scrum Board for Distributed Software Development *(ITS'15)*. In submission to the ACM International Conference on Interactive Tabletops and Surfaces (ITS'15).
2. Pernille Bjørn and Lars Rune Christensen. 2011. Relation work: Creating socio-technical connections in global engineering. In *ECSCW 2011: Proceedings of the 12th European Conference on Computer Supported Cooperative Work, 24-28 September 2011, Aarhus Denmark*. Springer London, 133–152.
3. Alexander Boden, Frank Rosswog, Gunnar Stevens, and Volker Wulf. 2014. Articulation Spaces: Bridging the Gap Between Formal and Informal Coordination. In *Proceedings of the 17th ACM Conference on Computer Supported Cooperative Work & Social Computing (CSCW '14)*. ACM, New York, NY, USA, 1120–1130.
4. Jed R. Brubaker, Gina Venolia, and John C. Tang. 2012. Focusing on Shared Experiences: Moving beyond the camera in video communication. In *Proc. DIS 2012*. ACM.
5. Paul Dourish and Sara Bly. 1992. Portholes: supporting awareness in a distributed work group. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '92)*. ACM, New York, NY, USA, 541–547.
6. Morten Esbensen and Pernille Bjørn. 2014. Routine and Standardization in Global Software Development. In *Proceedings of the 18th International Conference on Supporting Group Work (GROUP '14)*. ACM, New York, NY, USA, 12–23.

7. Robert S. Fish, Robert E. Kraut, and Barbara L. Chalfonte. 1990. The VideoWindow System in Informal Communication. In *Proceedings of the 1990 ACM Conference on Computer-supported Cooperative Work (CSCW '90)*. ACM, New York, NY, USA, 1–11.

8. Y. Ghanam, Xin Wang, and F. Maurer. 2008. Utilizing Digital Tabletops in Collocated Agile Planning Meetings. In *Agile, 2008. AGILE '08. Conference*. 51–62.

9. Stevenon Gossage, Judith M. Brown, and Robert Biddle. 2015. *Understanding Digital Cardwall Usage*. Technical report tr-15-01. Carleton University - School of Computer Science.

10. E. Hossain, M.A. Babar, and Hye young Paik. 2009. Using Scrum in Global Software Development: A Systematic Literature Review. In *Global Software Engineering, 2009. ICGSE 2009. Fourth IEEE International Conference on*. 175 –184.

11. Emam Hossain, Paul L. Bannerman, and Ross Jeffery. 2011. Towards an understanding of tailoring scrum in global software development: a multi-case study. In *Proceedings of the 2011 International Conference on Software and Systems Process (ICSSP '11)*. ACM, New York, NY, USA, 110–119.

12. Andreas Kunz, Stefan Dehlin, Tommaso Piazza, Morten Fjeld, and Thomas Olofsson. 2010. Collaborative Whiteboard: Towards Remote CollaBoration and Interaction in Construction Design. In *Proc. 27th International Conference on Applications of IT in the ABC Industry Accelerating BIM Research Workshop*. 132–140.

13. Brian J. McNely, Paul Gestwicki, Ann Burke, and Bridget Gelms. 2012. Articulating everyday actions: an activity theoretical approach to scrum. In *Proceedings of the 30th ACM international conference on Design of communication (SIGDOC '12)*. ACM, New York, NY, USA, 95–104.

14. Thomas Nescher and Andreas Kunz. 2011. An Interactive Whiteboard for Immersive Telecollaboration. *Vis. Comput.* 27, 4 (April 2011), 311–320.

15. M. Paasivaara, S. Durasiewicz, and C. Lassenius. 2008. Distributed Agile Development: Using Scrum in a Large Project. In *Global Software Engineering, 2008. ICGSE 2008. IEEE International Conference on*. 87 –95.

16. Lene Pries-Heje and Jan Pries-Heje. 2011. Why Scrum Works: A Case Study from an Agile Distributed Project in Denmark and India. In *Agile Conference (AGILE), 2011*. 20 –28.

17. Jessica Rubart. 2014. A Cooperative Multitouch Scrum Task Board for Synchronous Face-to-Face Collaboration. In *Proceedings of the Ninth ACM International Conference on Interactive Tabletops and Surfaces (ITS '14)*. ACM, New York, NY, USA, 387–392.

18. Jessica Rubart and Frank Freykamp. 2009. Supporting daily scrum meetings with change structure. In *Proceedings of the 20th ACM conference on Hypertext and hypermedia (HT '09)*. ACM, New York, NY, USA, 57–62.

19. Ken Schwaber and Jeff Sutherland. 2011. The Scrum Guide - The Definitive Guide to Scrum: The Rule of the Game. `http://scrum.org`. (2011).

20. Helen Sharp, Hugh Robinson, and Marian Petre. 2009. The Role of Physical Artefacts in Agile Software Development: Two Complementary Perspectives. *Interact. Comput.* 21, 1-2 (Jan. 2009), 108–116.

21. M. Stefik, D. G. Bobrow, G. Foster, S. Lanning, and D. Tatar. 1987. WYSIWIS Revised: Early Experiences with Multiuser Interfaces. *ACM Trans. Inf. Syst.* 5, 2 (April 1987), 147–167.

22. John C. Tang, Chen Zhao, Xiang Cao, and Kori Inkpen. 2011. Your Time Zone or Mine?: A Study of Globally Time Zone-shifted Collaboration. In *Proceedings of the ACM 2011 Conference on Computer Supported Cooperative Work (CSCW '11)*. ACM, New York, NY, USA, 235–244.

23. Steve Whittaker and Heinrich Schwarz. 1999. Meetings of the Board: The Impact of Scheduling Medium on Long Term Group Coordination in Software Development. *Computer Supported Cooperative Work (CSCW)* 8 (1999), 175–205. Issue 3.

24. Laurie Williams, Gabe Brown, Adam Meltzer, and Nachiappan Nagappan. 2011. Scrum + Engineering Practices: Experiences of Three Microsoft Teams. In *Proceedings of the 2011 International Symposium on Empirical Software Engineering and Measurement (ESEM '11)*. IEEE Computer Society, Washington, DC, USA, 463–471.