SOFTWARE PROCESS IMPROVEMENT AND PRACTICE
*Softw. Process Improve. Pract.* 2008; **13**: 527–544
Published online in Wiley InterScience
(www.interscience.wiley.com) DOI: 10.1002/spip.402

# Using Scrum in a Globally Distributed Project: A Case Study

**Research Section**

Maria Paasivaara,*,† Sandra Durasiewicz and Casper Lassenius
*Software Business and Engineering Institute, Helsinki University of Technology, FIN-02015 TKK, Finland*

While seemingly incompatible, combining global software development and agile practices is a challenge undertaken by many companies. Case study reports on the successful use of agile practices in small distributed projects already exist. How these practices can be applied to larger projects, however, remains unstudied. This paper reports a case study on agile practices in a 40-person development organization distributed between Norway and Malaysia. Based on seven interviews in the development organization, we describe how scrum practices were successfully applied, e.g. using teleconferencing and web cameras for daily scrum meetings, synchronized 4-week sprints, weekly scrum-of-scrums, and Jira for backlog management. Non-scrum agile practices included nightly builds, automated testing, and team rooms. Supportig global software development practices, e.g. frequent visits, unofficial distributed meetings, domain expert networks, and annual gatherings are described. Positive experiences of using scrum included improved communication, trust and motivation, as well as better perceived quality. Challenges included misunderstood requirements, lack of videoconferencing possibilities, and awkward communication in distributed meetings due to cultural and geographical distance. Copyright © 2008 John Wiley & Sons, Ltd.

KEY WORDS: global software development; agile software development; scrum

## 1. INTRODUCTION

Global software development (GSD), including outsourcing, subcontracting, and partnerships has become mainstream in software development. Several challenges related to GSD have been identified (Mockus and Herbsleb 2001, Holmström *et al.* 2006) and solutions to these have been proposed, e.g. dividing work into separate modules and making them independent in order to minimize communication between sites (Herbsleb and Grinter 1999). However, for software projects developing genuinely novel products, making a clear modular structure and minimizing communication might be difficult or even impossible. Projects with uncertain requirements and implementation technologies cannot provide clear requirement specifications to all parties up front (Paasivaara and Lassenius 2006).

In collocated software development this kind of project scenario led to the introduction of agile methods. These methods assume that software development is an empirical rather than a defined process and needs a different way of working than that provided by the traditional waterfall model (Royce 1970, Schwaber and Beedle 2002).

* Correspondence to: Maria Paasivaara, Software Business and Engineering Institute, Helsinki University of Technology, FIN-02015 TKK, Finland
†E-mail: Maria.Paasivaara@hut.fi

Copyright © 2008 John Wiley & Sons, Ltd.

Agile methods are a set of philosophically related iterative and incremental software development approaches. According to the agile manifesto, agile development emphasizes individuals and interactions over processes and tools, working software over comprehensive documentation, customer collaboration over contract negotiation, and responding to change over following a plan (Agile Alliance). Several agile methods have been developed, the most well known of which probably are scrum (Schwaber and Beedle 2002) and extreme Programming (XP) (Beck 2000).

While agile methods are supposedly effective in projects facing high uncertainty (Braithwaite and Joyce 2005), their application to GSD is not perfectly straightforward, as they rely heavily on face-to-face communication (Cockburn 2002), which is difficult or expensive to arrange in the GSD context. Thus, to gain the benefits of agile methods in GSD, the practices need to be modified to compensate for the lack of face-to-face communication.

Previous case studies (Simons 2002, Farmer 2004, Nisar and Hameed 2004, Fowler 2006) have shown that agile methods such as scrum (Schwaber and Beedle 2002) and XP (Agile Alliance) can be successfully customized to distributed projects. Even distributed versions of these agile methods – distributed scrum (Sutherland *et al*. 2007) and distributed extreme programming, DXP (Kircher *et al*. 2001) – have been developed.

Nevertheless, advice on pairing agile software development and GSD, also referred to as distributed agile development (DAD), is scarce. There are only a few reported experiences on applying DAD to industrial projects (e.g. Yap 2005, Smits and Pshigoda 2007). In particular, the literature is lacking in advice for large projects, as existing publications are mostly limited to the use of XP in small and middle-sized GSD projects (Boland and Fitzgerald 2004, Ngo-The *et al*. 2005, Hogan 2006). Although there are some experience reports discussing aspects of scrum in GSD (e.g. (Jensen and Zilmer 2003, Berczuk 2007)), we know of no case studies on distributed scrum projects with the aim of thoroughly describing the applied practices and their benefits and related challenges. This paper presents a single-case study of a mid-sized distributed software development project that started to apply scrum.

The paper is structured as follows. First, we present experiences reported in the literature on using agile practices, especially scrum practices, in distributed projects. Then, we describe our case study method and our findings. The results include experiences on how scrum practices were applied in our case project, what kind of challenges were faced and benefits gained, and what kind of other supporting practices were taken into use. Finally, we discuss our findings and suggest future research topics.

## 2. RELATED WORK

Distributed software projects with volatile requirements and uncertain implementation technologies need effective practices to be organized and managed successfully. We performed a systematic literature review on practices used in distributed agile software development projects. Based on our findings from the literature we divided the practices into two groups: (1) agile practices, and (2) supporting GSD practices. Agile practices include practices that come from an agile method and are either used as such or applied to distributed projects. Supporting GSD practices include practices that help tackle challenges arising from project distribution.

### 2.1. Agile Practices

The literature offers some advice especially on how to use scrum and XP practices in distributed agile projects. Basic scrum practices include daily scrum meetings, sprints, sprint planning meetings, sprint demos, retrospective meetings and backlogs. Next, we briefly present these practices and advice collected from literature regarding their usage in distributed projects. Finally, we present a few XP practices that are also used in distributed projects.

The most frequently used scrum practice is the daily scrum meeting, as it provides at least a partial solution to one of the biggest challenges in GSD (Mockus and Herbsleb 2001) and in DAD – communication (Simons 2002, Poole 2004, Ramesh *et al*. 2006). The daily scrum meeting normally takes 15 minutes, during which each team member answers the three scrum questions (Schwaber and Beedle 2001):

- What did you do since the last scrum meeting?
- Do you have any obstacles?
- What will you do before the next meeting?

Sutherland *et al.* recommend answering these three scrum questions before the scrum meeting via e-mail to shorten the time needed for teleconferencing (Sutherland *et al.* 2007). This approach also helps to overcome language barriers. Berczuk reports that Skype with a speaker and a microphone provides sufficient voice quality and is easy to set up for the scrum meetings (Berczuk 2007). The use of Web conferencing for daily scrums has also been described (Jensen and Zilmer 2003, Danait 2005). According to Danait (2005), daily scrum meetings both enhance communication and provide a coordination mechanism for everyone in the project. Jensen and Zilmer (2003) add that these meetings greatly improve cross-team project management. Burndown charts displayed on the Wiki pages for cross-location viewing provide visibility of the project progress (Danait 2005).

Iterations in scrum are called sprints, and normally last four weeks, but can be shorter. Before a sprint starts, a sprint planning meeting is arranged, in which the team plans the next sprint based on the backlog – a list of items prioritized by the product owner. When a sprint ends, the team demonstrates the new functionality to the product owner, the customer, or other interested parties. Finally, in the retrospective meeting, the team discusses the process and practices used, and if needed, makes changes to their way of working before the next sprint starts.

According to Holmström *et al.* (2006), sprint planning and retrospective meetings improve communication, coordination and team cohesion in a distributed project. Usually the whole team participates in a planning meeting (Beck 2000, Schwaber and Beedle 2001). Due to time-zone differences, it can be difficult to find a sufficient block of time that suits all parties (Berczuk 2007). Thus, Layman *et al.* (2006) suggest that only lead developers take part in the planning meetings. Simons (2002), on the other hand, emphasizes the importance of the participation of the whole team to get the viewpoints of all team members. He suggests that pre-work should be done before the actual planning meeting: customers and developers should clarify as many issues as possible before the meetings to keep the actual planning meeting short, since these kind of remote meetings held via Web conference (Layman *et al.* 2006) or phone (Simons 2002) are arduous. Berczuk (2007) reports that sprint review meetings can be centred on the local team and the product owner to minimize the number of remote meetings.

Demonstrations of working functionality can be arranged for customers at the end of every sprint (Danait 2005, Fowler 2006). They might be held using videoconferencing with desktop sharing (Paasivaara and Lassenius 2006). Jain (2006) reports that a customer could try out a new feature using VNC to connect to the developer's computer.

In addition to the scrum practices, DAD literature mentions a few XP practices that have been applied to distributed projects. Distributed pair programming has been studied especially in several distributed student projects. The onsite customer is another XP practice that has been discussed in the DAD literature. Onsite customer means that a customer representative works collocated with the XP team and is available to answer questions and transfer his or her business knowledge to the development team. In a distributed project it might be difficult, especially for an offshore team, to have a business customer working physically on their development site (Simons 2002). Instead, a proxy or a remote customer may help transfer business domain knowledge from the customer to the developers, and thus enhance communication (Nisar and Hameed 2004). The remote customer communicates with the development team either through videoconferencing or e-mail (Kircher *et al.* 2001). Another possibility is to send prototypes to the customer for testing and commenting (Layman *et al.* 2006). A proxy customer is helpful when the real customer is unavailable to answer questions. The proxy customer can make decisions on behalf of the real customer (Layman *et al.* 2006) and interface with the real customer when questions arise that he or she cannot answer (Simons 2002).

## 2.2. Supporting GSD Practices

In addition to agile practices, the literature on DAD reports practices that tackle challenges arising from distribution. We refer to these as supporting GSD practices.

The challenges caused by distribution include, for example, communication problems (Kircher *et al.* 2001, Simons 2002, Poole 2004, Ramesh *et al.* 2006, Berczuk 2007, Sutherland *et al.* 2007), lack of close physical proximity (Yap 2005, Ågerfalk and Fitzgerald 2006, Holmström *et al.* 2006), lack of team cohesion (Ramesh *et al*. 2006), lack of shared context and knowledge (Simons 2002, Yap 2005), and unavailability of team members (Yap 2005).

The supporting GSD practices strive to provide solutions to these challenges. Frequent visits are used to build and maintain collaboration relationships between distributed team members (Fowler 2006). A good collaboration relationship needs to be created for the remote communication to work effectively. There are two kinds of visits: seeding visits and maintaining visits. Seeding visits occur early in the project. Their aim is to build a relationship (Kussmaul *et al*. 2004, Fowler 2006) and they should be intense during the early development cycles (Ramesh *et al*. 2006). Maintaining visits are shorter and they aim to maintain the collaboration relationship (Fowler 2006). Ramesh *et al*. (2006) recommend that people from both onshore and offshore sites should travel. Customers and product managers should visit the development team. Developer representatives should travel to the customer's site. According to Braithwaite and Joyce (2005) and Danait (2005), team members should continually rotate between the sites, and at least one team member at any time should be away from his or her home location. This helps distant teams get along better and develop peer-to-peer relationships (Danait 2005). It is easier to maintain trust with remote team members if you have worked collocated with them in the past (Braithwaite and Joyce 2005).

A practice that particularly enhances communication between distributed teams is multiple communication modes (Braithwaite and Joyce 2005). Team members should be provided with several different kinds of communication media that can also be applied in parallel, i.e. individual and conference telephone, teleconference, videoconference, e-mail, instant messaging, Wiki and desktop sharing. A major part of the missing face-to-face communication is substituted by these tools to alleviate distance (Braithwaite and Joyce 2005).

Mirroring (Hogan 2006) and balanced sites (Braithwaite and Joyce 2005) can be used to reduce the dependency on the other site. These practices mean that each role in a team at one site has a counterpart on the other site. These persons may work closely together. For example, special knowledge on architecture is good to have on both sites.

For successful collaboration, team members need to gain a common understanding on how the teams at different sites work, think, communicate, and in general deal with the various issues and problems that arise from the development effort. Therefore, it is useful to send an experienced engineer for a longer period of time to the other site to facilitate cultural exchange (Poole 2004). The ambassador (Nisar and Hameed 2004, Hogan 2006) or rotating guru (Yap 2005) are similar practices that aim at more than just cultural exchange. The ambassadors report lessons learned and set future directions for the project. They participate in daily meetings and retrospectives of the visited team (Hogan 2006). Business-oriented ambassadors provide business context information to the offshore team (Braithwaite and Joyce 2005). Rotating gurus, who usually are senior team members, provide initial training and mentoring to the other site. As pointed out by Nisar and Hameed (2004), this is workable only for slightly larger budget projects.

The synchronization of working hours that is a widespread practice in GSD is equally important in DAD. Constant communication is possible only through the maximization of overlapping work hours. For example, early morning shifts for the onshore site and late evening shifts for the offshore site let the teams 'share the pain' of synchronizing the work schedule (Ramesh *et al*. 2006).

## 3. RESEARCH METHOD

The research presented in this paper is a single-case study (Yin 1994) of a large distributed product program using an agile process. We used purposeful sampling (Patton 1990) when looking for a suitable case. Our aim was to find a globally distributed project that already for some time had used an agile method or at least a collection of agile practices. The large globally distributed IT company that we contacted searched for suitable projects for our study, this chosen case being one of them.

The product program, henceforth referred to as EnergySoftware, develops a software product for oil and energy companies. The product is already in use in several customer companies and current development aims at creating new versions. Every new version is a new 'project' and the development of the whole product is called a 'program'. The development work is distributed between two countries, Norway and Malaysia, where our case company has its own offices. We describe the case program in more detail in Section 4.1.

We collected data using semi-structured, open-ended interviews that were recorded. We asked our interviewees to tell in their own words about the project, and which practices they have been using and how, instead of asking whether they have been using some specific practice. For this reason, not every interviewee would have mentioned every practice in use. However, by interviewing several persons we believe that most practices used have been mentioned. Altogether we performed seven interviews, each lasting one to two hours. In Norway, we interviewed four persons, each in a face-to-face interview, with one researcher asking questions and the other one taking notes. We were able to interview only one person from Malaysia face-to-face, while this person was visiting Norway. In this interview, we had one researcher interviewing, but not taking notes. Since we could not visit Malaysia for cost reasons, we interviewed two additional persons from Malaysia over the phone, using SkypeOut calls that were recorded. Our Norwegian interviewees included a product development director, a scrum master and two developers. All three Malaysian interviewees were developers. One of them had recently left the project.

We sent all recordings to an outside professional transcription company. One researcher that had participated in the interviews checked each transcription by listening to the tape and correcting the transcription mistakes.

The qualitative data analysis was done by one of the researchers. Data was analysed by qualitative coding according to instructions by Miles and Huberman (1994). The researcher coded all interviews using the qualitative data analysis software Atlas.ti, developing separate codes for all the identified practices, and mentioned benefits and challenges. The detailed codes were created during the coding, thus, all the codes arose from the data. Altogether the coding provided eight main code groups that totaled 77 subcodes. The idea of this two-level coding was to create a few main categories of codes, and then give detailed names for each sub-code making the code names useful when interpreting the data. The eight main categories that arose were: benefits of using scrum in a distributed project, challenges encountered when applying scrum to a distributed project, distributed agile practices, known agile practices, new agile practices, known GSD practices, possible quotations, and other interesting information. Here are a few examples of our codes: 'B_Improved trust after starting to use Scrum'; 'DAD_Distributed retrospective meeting'; 'Known-agile_Nightly builds'; and 'Z_Reason to start using Scrum'. The first letters or words tell the main category, e.g., B means benefits of using scrum, Ch for challenges encountered when using scrum, DAD for distributed agile practice and Z for other interesting information, and the last words give a detailed name for the code.

## 4. RESULTS

### 4.1. Case Description

The case organization is an IT company that has a globally distributed organization. The company's main activities are in Europe, and it has offices in other parts of the world, e.g. in Asia. The company is experienced in carrying out distributed projects, but using agile methods in them is a new experience.

The case selected for study is a large product development program with a history spanning more than ten years. Many large customers all over the world are using the product. New product versions are released approximately twice a year, and between the main releases, service packs are shipped. Currently, six old versions are maintained, since not all the customers upgrade with every new release.

The product organization has grown over the years from 18 persons to a current size of 190 employees. The product organization consists of two major groups: the service organization, which is distributed all over the world, and the product development organization, which is distributed in two locations, Norway and Malaysia. The service organization sets up new releases for customers, e.g. by configuring the system, and in some cases making minor modifications. The product development organization has approximately 20 persons in Norway and another 20 persons in Malaysia. The Malaysian development organization is slowly growing. This study focuses purely on the product development organization.

At the time of our study, the organization had used scrum for 1.5 years. Before adopting scrum, the development process was a combination of a traditional waterfall model and an iterative way of working. Releases were made with a time span of between six and nine months. The process contained lots of up-front planning activities and a large amount of documentation. The old process was considered very rigid and bureaucratic. All changes were considered negative, and caused a lot of re-planning. Development was done in increments. The quality after early increments was not considered good enough, and everything tended to work only after the last increment, just before a release. In addition, there was a perceived customer need for more agility. The organization used the same way of working for five years: another driver for change. The project manager in Norway proposed adopting scrum, and discussed it with key development personnel, who agreed to try it. Management approved a three-month scrum trial. The trial was successful, and the organization has been using scrum ever since.

Scrum adoption started with the teams being given a brief description of the process, basically stating that the old processes were not mandatory anymore, and that developers could use whatever process they wanted. However, there were some minimal requirements as to which practices to apply, as well as some rules:

- specifications should be written for everything that is implemented,
- a detailed design document should be produced,
- there had to be evidence of all testing activities performed, and
- the main goal for each sprint is to deliver release quality software.

The overall product consists of five modules or 'products' as the teams call them. A customer can buy one or more 'products'. Since the program is constantly developing new versions, each release project is called 'a project'. The product development organization is divided into seven teams. Five of them are built around the different modules or 'products', and each of them has a separate product owner. The product owners are not considered team members, perhaps because they travel a lot collecting requirements. The two remaining teams include a framework team and a maintenance team. The maintenance team is considered the most important, requiring the most experienced personnel, since the team members must know the whole product, and their work has direct and constant impact on customer satisfaction. The maintenance team does not have a separate product owner. Instead, all five product owners can give the maintenance team fixes to be done regarding their own products. The scrum master of the maintenance team works as the product owner for that team and coordinates the maintenance requests coming from the five product owners.

The number of persons in each team varies between release projects and iterations, mainly because different releases emphasize different modules. The team size varies from two to nine persons. Some teams are distributed between Norway and Malaysia. Whether a team is distributed or not can vary between iterations. All product owners are located in Norway. In some teams, all the developers are in Malaysia and only their product owner in Norway. There are also two scrum masters and one back-up scrum master in Norway and one scrum master in Malaysia. This means that one scrum master may have several teams to work with.

The development of the product started in Norway over ten years ago. The product program started to build an offshore development organization in Malaysia three years ago as the result of a decision to move part of the development to a lower cost country for cost reasons. Malaysia was chosen as the offshore location because the product program already had a small service organization and customers there. It was difficult to find developers with both sufficient IT skills and knowledge of the oil and energy business in Malaysia. Therefore, the company decided to hire clever graduates with good IT skills and train them regarding business knowledge. At the time of the interviews, approximately 60% of development effort was done in Malaysia. In Norway, the organization had the most experienced experts, like architects and product owners. The development organization is depicted in Figure 1.

The time difference between the two sites is seven hours during the winter and six hours during the summer. However, the daily working time is longer in Malaysia and they also have longer breaks. Thus, the overlapping working time for the sites is normally a couple of hours during the winter and three hours during the summer.

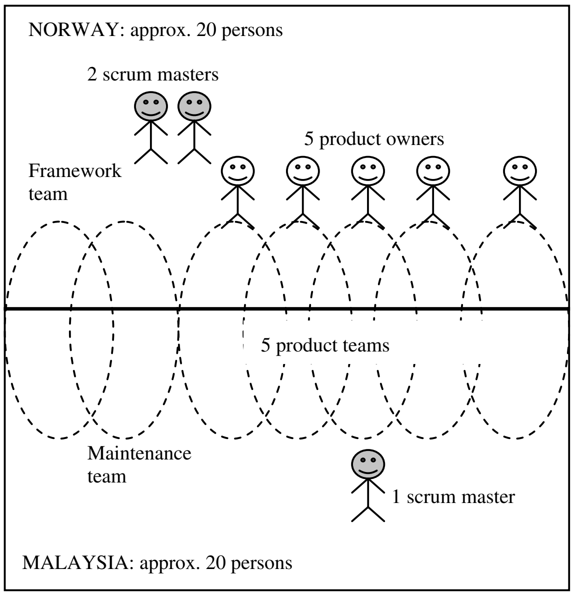

Figure 1. Product development organization

## 4.2. Scrum Practices

In this section we describe how basic scrum practices were applied in the distributed product program studied. In addition to describing the application of the practices, our interviewees mentioned challenges and benefits encountered when applying the practices in a distributed setting. The interviewees mentioned significantly more benefits of scrum practices compared to their earlier way of working, than challenges related to the application of scrum practices. After presenting each practice we also discuss the challenges and benefits mentioned. The main findings regarding the application of scrum practices and related challenges and benefits are collected in Table 1.

### *4.2.1. Daily Scrum Meetings*

Daily scrum meetings between sites were arranged using telephone conferencing and Web cameras. Application sharing was also possible, but was not normally needed. The meetings took place during the two hours that the teams had common working time. The meetings for different teams were consecutive, and took place in the same meeting room. The first meeting started at quarter to nine Norwegian time, the next at nine, and so on. This made it easy to participate in several meetings, if needed. This was the case, for example, for the scrum masters. In addition to the team members, the product owner normally joined the meeting. Even if the team had members only from Malaysia, the meeting was still distributed, since the product owner was in Norway. After the three scrum

Table 1. Distributed scrum practices

| Distributed scrum practice | How applied to a distributed project? | Challenges | Benefits |
|---|---|---|---|
| Daily scrum | All daily scrums 15 minutes distributed meetings. Besides developers and scrum master, also team's product owner from onsite participated. | In the beginning, meetings lasted only a couple of minutes. Improved when participants learned to report a suitable amount of useful information. | All interviewees considered very useful: mentioned as the most useful scrum practice. |
| | One meeting room from both sites reserved with teleconference connection and Web cameras, teams changed in 15-minute intervals. | Cultural differences in reporting impediments, difficult especially for Asians. | Brought transparency between sites. |
| | | | Revealed problems early on. Enabled to create contacts across sites and encouraged informal communication after the daily meetings. |
| Weekly scrum-of-scrums | Half-hour distributed meetings with one representative from each team and all scrum masters. Teleconference and Web cameras were used. | – | Distributed information between the teams. |

Table 1. (*Continued*)

| Distributed scrum practice | How applied to a distributed project? | Challenges | Benefits |
|---|---|---|---|
| | Three scrum questions answered regarding each team and two additional questions: 'Have you put some impediments in the other teams' way?' and 'Do you plan to put any impediments in the other teams' way?' | | Revealed possible problems early on. |
| | | | Opened discussion channels and encouraged informal communication between the teams. |
| Sprints | Synchronized 4-week sprints used in all development teams and in the framework team. | – | Short iterations brought transparency between sites, provided frequent monitoring opportunities and revealed problems early on. |
| | 2-week sprints in maintenance team to enable fast feedback to bugs found. | | |
| Sprint plan-ning meeting | The meeting was divided into three parts: distributed meeting during three synchronous working hours with product owner explaining backlog items followed by consecutive site-specific parts. | Time-zone difference made it difficult to arrange longer meetings. | Gave a possibility for team members from all sites to participate, to ask clarifications, to understand tasks and to commit to common goals. |
| | Distributed part arranged using teleconferencing and application sharing. | Cultural and language differences caused silence of some participants.<br>Difficult to recognize speakers when not seeing their faces. | Brought transparency to the project. |
| Sprint demo | Arranged as distributed meetings using teleconferencing and application sharing.<br>Participants: team, scrum master and product owner. | – | Brought transparency to the project.<br><br>Prevented problems by providing a frequent monitoring opportunity between the sites.<br>Ensured the understanding of the requirements, especially regarding the offsite. |
| Retrospective meeting | Arranged as distributed one-hour meetings directly after sprint demos using teleconferencing and application sharing.<br>Three questions discussed: 'What has been good during this sprint?', 'What has not been that good?' and 'What kind of improvements could we do?' | The output from these meetings was not yet utilized effectively. | – |
| Backlog | Separate backlogs for each team in Jira, updated by product owner, all team members can access.<br>All product owners can add new issues to the maintenance backlog. | – | All team members can access and pick items and monitor progress.<br><br>Have been very satisfied with the tool. |

questions were answered, a discussion typically took place, in which questions were asked and answered.

In the beginning, the interviewees reported that it was difficult to encourage everybody to talk and to tell enough about their tasks and impediments. This was the case especially for persons coming from an Asian culture as a manager from Norway described it:

*'We have seen also that it has been a challenge, particularly for the Malaysian side to report impediments, because they can say: ''Since yesterday I've done these and these things, tomorrow I plan to do this and this, but I really have to have some feedback from that person to be able to do this'', and then ''No impediments''. They will report they have no impediments. But of course they have an impediment because they are waiting for something, and they are not really sure at all they get it. (...) We have been working a bit with kind of encouraging them to be more direct in communicating and telling things as they are. But it's really, they will never get really to the level where we are due to the culture of course.'*

In the beginning, the daily scrum meetings lasted only a few minutes, a problem that was tackled by the scrum masters encouraging everybody to talk and tell more about their tasks and impediments. At the time of the interviews, the meetings took 15 minutes and communication was reported to be much better. If more discussion or clarifications were needed, the team scheduled additional meetings after the scrum meeting.

According to our interviewees, the daily scrum meetings encouraged team members to communicate more also outside the meetings, which was seen as one of the greatest benefits of daily scrums. Daily scrum meetings also provided a good way for everybody to get an overview of the project situation, and made it easier than before to monitor the offshore situation. Moreover, problems were identified quickly, since now it was difficult to hide problems over a longer time. Most of the interviewees mentioned the above benefits: increased transparency to the other site, getting a good overview of what was happening in the project, and enhanced communication across sites. One of our interviewees even stated:

*'I think that [daily scrum meetings] was the best thing that happened to these distributed teams.'*

#### *4.2.2. Weekly Scrum-of-Scrums*

A half-hour scrum-of-scrums meeting was arranged once a week. One team member from each team participated in this meeting. The team decided who was going to participate; the participant was not always the same person, but varied as needed. In addition to these team members, all scrum masters typically participated in this meeting. During the scrum-of-scrums, the three scrum questions were answered from a whole team perspective. Thus, each team representative told what his or her team had been doing since the last meeting, what it planned to do before the next meeting and what kind of impediments they had. Moreover, two additional questions were answered: 'Have you put some impediments in the other teams' way?' and 'Do you plan to put any impediments in the other teams' way?' The goal of these questions was to ensure successful integration.

The weekly scrum-of-scrums enhanced communication between the teams, and helped the teams be aware of what the other teams were doing, and to know in advance whether the work in some other team was going to have an impact on their own. Thus, the weekly scrum-of-scrums seemed to be a very useful practice and no one mentioned any challenges related to it.

#### *4.2.3. Sprints*

The five product teams and the framework team had synchronized 4-week sprints. This meant that all sprints started and ended at the same time. They had the same code trees, and at the end of a sprint they built an environment for the sprint demonstrations. Even though the teams during a release project stayed quite stable, it was possible to change the emphasis between different products or modules, between the sprints, and move persons from one team to another.

The maintenance team was the only exception to the 4-week sprint cycle, as their sprint cycle was two weeks. The reason for this was that hot fixes were released every two weeks.

In addition to normal sprints, the program was using 'design sprints', during which mainly design work took place. They did not want to have a separate way of working when doing design work, since they already had one way of working that people knew. Thus, scrum principles and practices were applied also to design work, which according to our interviewees, had been working well. Thus, when starting to do something totally new, the team might start with a design sprint with daily scrum meetings, etc.

### 4.2.4. *Sprint Planning Meetings*

The sprint planning meetings were normally divided into three phases: distributed meeting, local meeting in Norway and local meeting in Malaysia.

The first part was time-boxed for three hours. Malaysian and Norwegian team members participated in this virtual meeting through teleconferencing. For application sharing, Microsoft NetMeeting was used. In these meetings, Web cameras were not used. In the meetings, the scrum framework, a document that describes how scrum is applied, was typically reviewed first. In particular, any adjustments done since the last sprint planning were highlighted. An adjustment could be, for instance, a new concept for doing peer reviews. Thus, sprint planning sessions could have this kind of additional role of a training session.

Subsequently, the product owner started to go through items in the backlog, and the team asked questions. Before the meeting, the product owner had prioritized the backlog and made preliminary estimates for the backlog items. This phase of the meeting lasted until lunch in Norway, when it was time for the Malaysian team to finish their workday.

The Norwegian team typically continued the meeting for the rest of the day, dividing backlog items into more detailed tasks and adjusting the estimates made by the product owner. Moreover, the team made at least initial assignments of the tasks to different team members.

The Malaysian team continued the work the following morning. They met locally and discussed and commented on the draft plan they had received from Norway. They might do some adjustments and also add their names to some tasks. If needed, the issues raised were discussed together in the next daily meeting.

Since the maintenance team had two-week sprints, the team also held sprint planning meetings more often, twice a week, and always in a distributed fashion. In addition to the team and the combined maintenance team scrum master and product owner, all other product owners were invited, giving them an opportunity to influence which fixes to add to the next update. Since customers sometimes needed fixes right away, the team had a fast-track routine for handling such requests. To accommodate for this, the team left a capacity buffer of 20% for handling fast-track issues. The buffer needed was possible to forecast at least to some level, since it depended on the situation in the customer installation projects.

One of the challenges mentioned regarding planning meetings was that people from the Malaysian teams did not ask enough. The Norwegian team members felt that this would have been a good opportunity to transfer business-related knowledge to Malaysia if they just would have used this opportunity by asking questions. Otherwise, the planning meetings were considered useful, and team members were satisfied that they could affect the planning.

### 4.2.5. *Sprint Demos*

Demos were arranged using the same technology as in the sprint planning meetings: teleconferencing and application sharing. In addition to the team members, the product owner and scrum master participated. The team prepared an agenda for the demonstration, according to which each issue was gone through. If the quality seemed to be good, everybody applauded.

As an improvement to the demos, a pre-demo had been instituted before the real demo to let the audience know what to expect, what to look for, and to prepare questions.

### 4.2.6. *Retrospective Meetings*

Retrospective meetings took place directly after the demos, following a short break. The team, the product owner, and the scrum master participated in the retrospective, which was arranged as a distributed meeting. It was time-boxed for a maximum of one hour, during which the team discussed three questions: 'What has been good during this sprint?'; 'What has not been that good?'; and 'What kind of improvements could we do?'

While the teams still practiced retrospective meetings, they felt that they were not really effectively using the improvement ideas that came out, or following up that improvements were really carried out, or what their impact had been. Moreover, they felt that implemented improvements that could benefit all teams were not effectively shared between the teams.

To solve these problems, the organization was implementing a transition backlog in which the improvement ideas could be saved and prioritized, their implementation monitored, and good practices shared with other teams.

*4.2.7. Backlogs*

The program used a tool called Jira for managing backlogs. Each team had its own backlog in Jira and all team members could access it. The backlogs were updated by the respective product owners. The maintenance team had a backlog of its own, to which all product owners could add new issues. The process of adding issues to the maintenance backlog was a bit different from what the other teams had: all customers had access to Jira, where they reported bugs found. The service organization first checked each issue, and if they found a product-related bug they moved that task over to product development and assigned it to the product owner of that specific area. The assigned issues were prioritized using categories describing their criticality. The product owner who received the bug analysed it, and when he marked it as 'verified', it was automatically moved to the maintenance backlog. The product owner decided, as well, to which maintained versions a fix was going to be done. All teams were very satisfied with this tool.

### 4.3. Additional Agile Practices

In addition to the scrum practices discussed above, the product program used three other agile practices: nightly builds, automated testing, and team rooms. Next, we briefly explain how these practices were applied in this product program.

*4.3.1. Nightly Builds and Automated Testing*

The teams checked in their code at least daily to CVS, a centralized version control system located in Norway. It could be accessed by all team members. Every night, the whole product was built and a set of automated tests run. If the build was unsuccessful, the team that broke it would fix errors and rebuild the product after the errors had been fixed.

*4.3.2. Team Rooms*

The basic principle was that in Malaysia and in Norway there was one room for each team, making it easy for collocated team members to have on-the-spot discussions. When a person switched teams, he or she was also relocated into the new team's room. The project had gradually moved to this. When, e.g. Malaysian team members visited Norway they shared the room with the Norwegian team members.

### 4.4. Supporting GSD Practices

In this section, we discuss GSD practices that were successfully used in this distributed product program to support the scrum practices.

*4.4.1. Unofficial Distributed Communication*

Our interviewees felt that after starting to use scrum there had been significantly more one-to-one communication between the sites than before. This kind of communication often took place after the official daily scrum meetings, as one of our interviewees explained:

> *'So actually a lot of opportunities opened up. That means during the daily scrum you can just tell your counterpart ''I want this assistance. So shall we meet on another meeting or shall we stay on the phone after this?'' You can choose, so there were a lot of opportunities in place to make sure the communication between Malaysia and Norway was there. So Scrum made it possible.''*

Unofficial communication took place through teleconferences, chat, e-mail, or over the Internet using headsets. Some team members also used Web cameras. Tool choice seemed to depend both on the purpose of the communication, e.g. chat was used to ask short questions or for checking whether the other party was available to receive a phone call, and user preferences. Some individuals preferred synchronous voice communication, while others with limited language skills preferred written communication. After starting to use scrum, the amount of voice communication had increased, as the threshold to call or arrange ad-hoc meetings had lowered.

Our interviewees emphasized that it was important to offer several different good quality communication tools for distributed teams, so that everybody can choose a suitable tool for his or her communication purposes. Videoconferencing was not used at the time of the interviews, only Web cameras, even though videoconferencing would have been preferred, if just the bandwidth between the sites would have allowed for it. The challenges faced with the current communication tools were mainly related to difficulties to understand and explain difficult issues when not being face-to-face.

### *4.4.2. Frequent Visits*

During the first six months after starting to apply scrum, the project had several persons from Norway working full-time in Malaysia. At the time of the interviews, several team members traveled frequently between the sites. The visits normally lasted between two and four weeks, which made it possible for a team to really work together. Our interviewees thought that especially during critical project phases, it was important to collocate the team, e.g. for the last iteration before a release or for the first iteration, when most of the planning takes place.

An interviewed Norwegian manager explained the reasons for collocating developers of one team for a certain period:

> *'One reason could be that this is an area where the team in Malaysia hasn't really got up to speed, they need some more intensive working together with people with more expertise to get really up to speed. So then maybe you say that OK, they come over here for the next sprint and work, sit in the same room as the rest of the team to get much more into it, and that really is efficient, really efficient. Then they get back and can bring back a lot of knowledge and also of course they build stronger contacts for the people here. They can go out in the evenings or something and get closer to each other from that perspective, so that helps a lot for the communication. Another reason there could be that we see that it is a higher risk when we run a mixed team and we have a critical delivery. It is a higher risk that we do not reach that target if we sit distributed. So we may do it just to reduce the risk, because then it's easier to monitor exactly where we are if we have more people in the same room, and it's also easier to take actions, and we can ensure that at least the people are not sitting and waiting for clarifications. So that would be another typical reason for us to do it.'*

At the moment, people travel on a need basis. In the beginning, they had a travelling plan for the Norwegians. Now they noticed that the Malaysian team members travel more often. Thus, they are thinking of taking into use a travelling plan for the Norwegians again to make them travel more. Most Norwegians are subject area experts, whose knowledge would be beneficial to share in Malaysia. Offsite personnel found it extremely useful to meet the onsite experts face-to-face and ask questions and discuss difficult issues. Some of our interviewees, especially from offsite, hoped that also onsite personnel would travel more often to allow more offsite persons an opportunity to meet them. At the time of the interviews, approximately half the project team members had visited the other site, and all had met face-to-face in the annual gatherings.

The frequent visits provided good opportunities for getting to know persons from the other site, discuss difficult issues, and get a better picture of the project. Face-to-face meetings also increased trust between team members and encouraged them to continue communication after the visits. It seemed to be important to arrange visits not only in the beginning of the project, but quite frequently during the project, as well. The visits normally lasted at least a couple of weeks. Thus, the visits were not just short trips to meetings, but instead longer stays during which distributed team members could really work together.

Even though our case program arranged quite a lot of trips, none of the interviewees mentioned any problems of arranging trips due to cost reasons or limitations to travel because of costs. Instead, it seemed that everybody found the current model of frequent visits as very useful and even more visits were hoped for.

### *4.4.3. Annual Gathering*

The program had arranged social gatherings during the whole program, including the service organization. The last gathering was in Rome, and all 190 persons travelled there for a long weekend. The product development organization held a half-day session together during which they heard and discussed future actions and had a team building exercise. The rest of the time was used for common social events. Everybody we interviewed considered it a successful event and managers thought that it was an investment in the future, as one interviewee stated:

*'I think that it's something that people are talking about still, and that is something that is good for people and for the company, as well, to do something like this every now and then. And yes, it is expensive but, I'm very glad that the boss is seeing this as an investment for the company.'*

*4.4.4. Domain Expert Networks*

To connect the service organization distributed around the world to the product development organization, the company had started to build domain expert networks around the product owners approximately half a year after starting to use scrum. The reason for this was the limited capacity of single-product owners. The idea was that each product owner would have a network of four to seven persons consisting mainly of experts from the service organization, but also of some experienced individuals, e.g. architects, from the product development organization. The domain expert network would support the work of a product owner, who easily gets a lot of work and may become a bottleneck. Another benefit is that this network helps experts on the service side to have a link to product development, otherwise they may easily feel frustrated and their expertise may be left partly unused. The challenge of this network structure has been to keep it alive, mainly the responsibility of the product owner. At the time of the interviews, the project had one well functioning domain expert network and three that were not yet working that well. The domain expert networks had teleconference meetings every second week, where they exchanged information and discussed challenges, solutions, priorities, and new backlog items. The experts from these networks were allowed to add new backlog items to the maintenance backlog, even though the final responsibility of the backlog management still remained with the product owner.

*4.4.5. Centralized Version Control*

The project had a centralized version control system, a CVS server, in Norway. It could be accessed by all team members through VPN. The teams checked in their code at least daily.

*4.4.6. Visiting Engineer During the First Iteration*

When scrum was taken into use in the Malaysian office, one engineer, a scrum master from the Norwegian office, travelled to Malaysia and stayed there during the first sprint. He had studied scrum, but it was still quite new to him. He worked together with the teams and they jointly found solutions on how to apply scrum.

*4.4.7. Onsite System Expert*

When the Malaysian office was established, two persons from Norway moved there. One of them runs the office, and the other one is an expert who knows the customers' business, and how different components of EnergySoftware work.

*4.4.8. Release Steering Committee Meeting*

The program had a release steering committee that met at least every second month and planned the main directions for the next releases. The participants of these meetings were all product owners and organization heads from both services and development sides.

### 4.5. Challenges Faced

When applying scrum to this distributed product program, the teams faced challenges especially due to the geographical and cultural distances between the countries.

*4.5.1. No Possibility for Videoconference*

One challenge at the time of the interviews was that the network connections between the sites were not fast enough for videoconferencing. However, they hoped that this could be solved soon in order to make it possible to see people and their reactions in meetings. Web cameras, which were in use, were not considered good enough, since the resolution was poor, and there were transmission delays. So, instead of seeing the reactions on peoples' faces it was possible to see only a small picture that was not even being continuously updated.

*4.5.2. Silence Caused by Distance*

In the beginning, the daily scrum meetings were very short when team members did not know how to communicate and how much to tell. Later on, after a period of daily practice, the problem seemed to have been solved. However, the same problem was still present in some sprint planning sessions, in which the discussion was less guided. Especially, the Norwegians thought that the Malaysian team members were very silent. For example, one Norwegian interviewee commented on the sprint planning sessions:

*'It can go maybe 20 minutes before we hear anything or we have to ask: are you still there?'*

The Norwegians felt that they themselves were very eager to ask questions from the product owner in these meetings, almost attacking him with questions, whereas the Malaysians seemed to prefer just listening. Thus, the Norwegians felt that they could not be sure whether the offsite team had really understood everything. They were hoping to find a solution to this.

#### *4.5.3. Misunderstanding Requirements*

Misunderstandings were common. A developer might have had a totally different picture in his head about some functionality than a product owner had. This was especially the case when developers were sitting far away from the product owner, but it had happened in Norway, as well. In the worst case, the misunderstanding might not be noticed until in the sprint demo. To verify the correct understanding, the product owners had learned to ask follow-up questions. By asking such questions, they made a developer explain in his own words what he was planning to do or what the functionality should do. The follow-up question could, for example, be: 'Do you have an idea how you are going to code this?' These kinds of questions could be asked at the end of daily scrum meetings to begin a discussion.

### 4.6. Positive Experiences

Even though facing some challenges, the overall experience on starting to use scrum in this mid-sized distributed project was very positive. Based on our interviews, it seemed that the whole organization was satisfied with the decision to start to use an agile method. They saw many things that had helped them make this change successful: their organization was already very experienced, they had competent people, they got commitment from management and they had a version control system and regular builds in place.

The agile practices were considered very suitable for distributed projects, as one of our interviewees stated:

*'I think the strength of working in an agile way when we are different, and in different locations is that you actually have more frequent communication. And also focus on issues, impediments, and that you solve them on the way and do not wait until they make a big risk for the project. So I think that is very much, it is very valuable. But I think also with that said, the optimal way of working is to gather all people in the same room in the same place. But as we are now in a global world that is not the way it goes. And then you need to find the best techniques for overcoming the challenges it is to work with other people in other corners of the world.'*

Taking agile practices into use had many benefits according to our interviewees. The improved communication was clearly the biggest benefit that all interviewees mentioned. Improved trust, motivation and quality were other oft-mentioned benefits. Next, these benefits are discussed in more detail.

#### *4.6.1. Improved Communication*

The biggest benefit of scrum seems to be significantly improved communication between the sites.

Our interviewees explained that scrum offers a very structured way of communication, which was regarded as a good thing, as one of the interviewees from Malaysia put it:

*'So it was a better way of working because there was a really, really structured way of communication. Especially when you are working from two different locations, communication is everything. And agile definitely looks into that, a lot of communication, very structured within the Malaysian team, between teams and also within the Norwegian team. (...) Before this I think you just worked at your desk and then you got your assignment and then you just logged.'*

Our interviewees mentioned that communication quality is better than before and communication is much more frequent, especially between the sites. There is also more one-to-one communication between the sites, e.g. after the daily scrum meetings.

A big difference between the current situation and the situation one year ago was perceived. Now everybody has to talk, and is trained to talk in the scrum meetings. There is a feeling that the scrum setting has forced increased communication, which is considered very beneficial, as one of the interviewees from Norway said:

*'''So we are actually forced to communicate in a Scrum setting and I think that is good.'' Also Malaysian team members saw it as a good thing that Scrum encouraged them to communicate, as one of them told us: ''So now by giving them [young Malaysian team members] this slot that at Scrum time you stand up and say all this, it builds confidence in them. They are becoming very happy with that, their language is improving, their confidence is improving. Previously, they were not so brave to call to Norway and say ''I have a problem''. (...) But now you are given an opportunity. (...) And I feel the people who are in the scrum team, especially these local graduates (...), now being exposed to an European company, having Scrum in place gets all the teams really working. Yeah, it really is a very good thing that is happening from that.'*

The same interviewee continued, saying that the Malaysians were not anymore as afraid to contact Norwegians informally, e.g. by phoning them as previously:

*'Well, as I said earlier, when the Asian culture, we are very reserved. (...) As a manger I say ''Okay, you have this problem. You have written to them, the e-mail has not been answered. Why don't you just pick the phone and talk to them?'' They will never do that, because of fear, because they think their language is good enough or they think that they are not welcome. (...) When we first started the Scrum, everyone was like ''umm, ahh, umm, ahh'', now you can't stop them from talking. (...) They started communicating and the relationship between Norway and Malaysia now is very good, because they can talk, because they are talking every time. (...) [Earlier] they just played safe on the e-mail: ''I sent in the e-mail, I sent in the reminder.'' ''Please pick up the phone and ask him, if you don't get the result!'' ''No no no, I will send him another reminder.'' But now that has dropped out. I think the e-mail has taken a secondary role to direct communication, after Scrum came.'*

Surveys to all team members after each release project had already been conducted before scrum adoption. These project surveys asked developers about, e.g. processes used, testing concepts, code reviews, documentation, product owners, and cooperation and communication. The scores for communication and collaboration had risen continuously since they started using scrum, currently being at a high level. One of the scrum masters commented on this saying:

*'This gives a good indication that Scrum works.'*

### 4.6.2. *Improved Trust*

The interviewees reported that scrum, with more frequent communication, has had a very positive effect on the level of trust between the sites. It is easier for onsite personnel to trust that offsite is really doing good work and can also be given more demanding tasks, since they can monitor on a daily basis what is happening offsite. One of the Malaysian team members had noticed this and saw it as a good thing:

*'I think the trust is better now. Because initially, when you had the waterfall model, there was always this feeling that the Malaysians were not good enough to develop the tougher part of it, so let's keep the higher and tougher part in Norway, and let's give the mundane or routine stuff to the Malaysians, that way we are okay. But now I think, since they have this communication on the spot, it doesn't matter, because they can see it happening and if they think Malaysia can't handle it, they can really monitor it. (...) So they know (...) what you're doing because you are reporting daily, so they have an eye on it. (...) if Malaysia picks something that they think is quite critical, they don't have any fear with that because they are daily looking at what's happening. So that helped as well. (...) That was a great improvement.'*

### 4.6.3. *Improved Motivation*

The team members are now more motivated than before. A scrum master explained:

*'They motivate themselves (...) They get clear and quick clarification so they don't have to wait. They get access to the right people at the right time and everyone has the same value in the team (...) That they can have an impact on the work that they and also their team is doing, is something that makes you feel good.'*

A manager from Norway had received same kind of feedback from Malaysia:

*'People report back that they feel they have more influence, (...) and they also are closer in the loop when things are decided, so and it's a more interesting way of working.'*

*4.6.4. Improved Quality*

Our interviewees reported improvements in product quality. Although increments had been used before, the quality of the product had not been good after the first increments, and the product had typically worked properly only at the end of the last increment. Now, a working demo was built at the end of each iteration. Thus, the quality at the end of an iteration was quite close to release quality. However, some also said that they now knew less about the quality. Earlier, there was a separate testing team that did rigorous testing in the end, and everything was documented. Now that team did not exist anymore. Even though the overall feeing was that the quality was now better, one of the improvement targets was to improve monitoring and documentation of quality, so that they could easily get more information about the quality level.

## 5. DISCUSSION

In this paper, we described how agile practices based on scrum had been applied to a mid-sized distributed software product development program. The overall experiences in using scrum in a distributed setting were very positive. All our interviewees stated that according to their experience, an agile method like scrum, is well suited to distributed settings. It actually helps to mitigate the biggest problem of GSD projects, communication, by giving frequent possibilities to communicate across distributed sites. It could be even said that scrum practices almost force distributed team members to communicate frequently and really learn to communicate, which can be seen as very positive.

The main contribution of this paper is a detailed description on how scrum practices were successfully applied to a mid-sized distributed software product development program, and what kind of challenges and benefits this application had brought. Besides the scrum practices the product program had used some GSD practices to support the distributed way of working. Frequent visits was one of the most important GSD practices they had successfully used. Meeting face-to-face both in the beginning and during the project and really working together was one of the success factors in this project. Thus, it seems that a suitable combination of scrum practices supported by selected GSD practices can lead to successful distributed collaboration. We hope that the experiences collated in this paper are useful for other companies planning to apply scrum or other agile practices to distributed settings. Next, we summarize some lessons learned from the case study.

### 5.1. Lessons Learned

In the process of adopting and applying scrum, our case organization learned several valuable lessons. We discuss some of them briefly below.

1. Plan the process change to scrum properly. Take into use all the basic scrum practices according to the book. A manager from Norway gave a piece of advice to others starting to use scrum in a distributed project:

'I think that the first thing is that if you decide to do it, then you need to do it properly. You cannot start using scrum or agile half-way, then you won't be able to take out the benefits.'

2. Arrange proper scrum training if the teams have not used scrum before. Materials or classroom trainings are not enough. You need to have an experienced person available to support and help teams during the first sprints.
3. Travel enough also during the project. Arranging face-to-face visits for distributed team members in the beginning is important to build a good relationship, but also later on to maintain collaboration. Visits lasting a few weeks, during which team members can really work together are important especially when doing challenging tasks.
4. Encourage frequent communication, both during meetings and informally. The biggest benefit from scrum seemed to be the practices that require frequent communication. However, the practices do not work if people do not learn to communicate. Moreover, it is important to also communicate informally outside the meetings. Frequent formal communication seems to

support the emergence of frequent informal communication.

5. Provide a set of communication tools. Different kinds of communication and various personal preferences require different tools. Tools and connections that guarantee good voice and picture quality are important to decrease the perceived distance. Arranging a possibility to really see people in distributed meetings, e.g. with good quality videoconferencing could also improve interaction.
6. Arrange at least some overlapping working time for distributed teams during which meetings and synchronous communication can take place. When using scrum, distributed meetings are a very important practice and after daily scrum meetings, there needs to be at least some common time during which team members can have one-to-one discussions, ask questions and solve problems.

### 5.2. Limitations

As a single-case study in a single organization, the study is clearly limited. We were able to interview only seven persons. Interviewing additional people from different teams and different roles might have complemented the picture especially regarding the Malaysian organization. However, regarding the application of scrum practices to distributed settings we received quite a good picture, since in the last interviews we did not learn of any new practices that were not mentioned in the earlier interviews. Thus, regarding the practices, the data seemed to be saturated. The main benefits and challenges of scrum described in this paper were each mentioned by several interviewees. Thus, we believe that we have captured the most important of these. However, there were several additional challenges and benefits that were mentioned only by one person each.

### 5.3. Future Research

In the future, we plan to strengthen our results by studying additional distributed projects using agile practices, and compare the way they have applied agile practices to distributed settings, the challenges faced, and positive experiences gained.

We also think that our results raise the need for additional and more focussed research. Of particular interest would be to better understand the role of frequent communication, here exemplified by the daily scrums as a mechanism for overcoming the challenges of distance in GSD projects. It is possible that the benefits are not limited only to DAD projects.

## ACKNOWLEDGEMENTS

The authors gratefully acknowledge the financial support of the Academy of Finland (Grant No. SA 107636, 2005-2007) and the Finnish Funding Agency for Technology and Innovation (MaPIT project). We would also like to thank our case company and all the interviewees.

## REFERENCES

Ågerfalk P, Fitzgerald B. 2006. Introduction. *Communications of the Acm* **49**(10): 26–34.

Agile Alliance. Manifesto for Agile Software Development, http://agilemanifesto.org. Cited 25.6.2008. Washington DC, USA.

Beck K. 2000. *Extreme Programming Explained: Embrace Change*. Addison Wesley: New York.

Boland D, Fitzgerald B. 2004. Transitioning from a Co-located to a globally-distributed software development team: a case study at analog devices inc. *Proceedings of the ICSE Workshop on Global Software Development*. Edinburgh: Scotland.

Braithwaite K, Joyce T. 2005. Xp expanded: Distributed extreme programming. *Proceedings of the 6th International Conference on Extreme Programming and Agile Processes in Software Engineering*, XP 2005. Springer: 180–188 Shefheld, UK.

Cockburn A. 2002. *Agile Software Development*. Addison-Wesley, Longman Publishing Co., Inc.: Boston, MA.

Danait A. 2005. Agile offshore techniques – a case study. *Proceedings of the Agile Conference*, July 24–29, 214–217 Denver, Co, USA.

Farmer M. 2004. Decisionspace infrastructure: agile development in a large, distributed team. *Proceedings of the Agile Development Conference*, Salt Lake City, UT, USA.

Fowler M. 2006. Using an agile software process with offshore development, (Available: http://martinfowler.com/articles/agileOffshore.html) Referenced: 19.12.2007.

Herbsleb J, Grinter R. 1999. Architectures, coordination, and distance: Conway's law and beyond. *IEEE Software* **16**(5): 63–70.

Hogan B. 2006. Lessons learned from an extremely distributed project. *Proceeding of the Agile Conference*, July 23–28 Minneapolis, MN, USA.

Holmström H, Conchùir E, Ò, Agerfalk PJ, Fitzgerald B. 2006. Global software development challenges: a case study on temporal, geographical and socio-cultural distance. *Proceedings of the International Conference on Global Software Engineering, ICGSE'06*, Florianopolis, 3–11.

Jain N. 2006. Offshore agile maintenance. *Proceedings of the Agile Conference* Minneapolis, MN, USA.

Jensen B, Zilmer A. 2003. Cross-continent development using scrum and Xp. *Proceedings of the XP*. Springer: Berlin, 146–153.

Kircher M, Jain P, Corsaro A, Levine D. 2001. Distributed extreme programming. *Proceedings of the International Conference on eXtreme Programming and Flexible Processes in Software Engineering*, Sardinia, May 20–23.

Kussmaul C, Jack R, Sponsler B. 2004. Outsourcing and offshoring with agility: A case study. *Proceedings of the 4th Conference on Extreme Programming and Agile Methods*. Springer Berlin/Heidelberg: Calgary, 147–154.

Layman L, Williams L, Damian D, Bures H. 2006. Essential communication practices for extreme programming in a global software development team. *Information and Software Technology* **48**(9): 781–794.

Miles MB, Huberman AM. 1994. *Qualitative Data Analysis*. Sage Publications: Thousand Oaks, CA.

Mockus A, Herbsleb J. 2001. Challenges of global software development. *Proceedings of the Seventh International Software Metrics Symposium*, (METRICS 2001, IEEE), 182–184 London, UK.

Ngo-The A, Hoang K, Nguyen T, Mai N. 2005. Extreme programming in distributed software development: A case study. *Proceedings of the International Workshop on Distributed Software Development*, August: Paris, France.

Nisar M, Hameed T. 2004. Agile methods handling offshore software development issues. *Proceedings of the INMIC 2004, 8th International Multitopic Conference*, 417–422 Lahore, Pakistan.

Paasivaara M, Lassenius C. 2006. Could global software development benefit from agile methods?. *Proceedings of the International Conference on Global Software Engineering, ICGSE '06*, 109–113 Costa de Santinho, Florianopolis, Brazil.

Patton MQ. 1990. *Qualitative Research and Evaluation Methods*. Sage Publications: Newbury Park, CA.

Poole CJ. 2004. Distributed product development using extreme programming. *Proceedings of the 5th International Conference, XP 2004*. Springer Berlin/Heidelberg: Garmisch-Partenkirchen, 60–67.

Ramesh B, Cao L, Mohan K, Xu P. 2006. Can distributed software development be agile? *Communications of the ACM* **49**(10): 41–46.

Royce W. 1970. Managing the Development of Large Software Systems. Reprinted from *Proceedings, IEEE WESCON*, 1–9.

Schwaber K, Beedle M. 2001. *Agile Software Development with Scrum*. Prentice Hall: New York.

Schwaber K, Beedle M. 2002. *Agile Software Development with Scrum*. Prentice-Hall: Upper Saddle River, NJ.

Simons M. 2002. Internationally Agile, *InformIT*, March 15th.

Sutherland J, Viktorov A, Blount J, Puntikov N. 2007. Distributed scrum: Agile project management with outsourced development teams. *Proceedings of the 40th Annual Hawaii International Conference on System Sciences, HICSS*, 274a–274a Walikoloa HI, USA.

Smits H, Pshigoda G. 2007. Implementing scrum in a distributed software development organization. *Proceedings of AGILE 2007*, 371–375 Washington, DC, USA.

Yap M. 2005. Follow the sun: distributed extreme programming development, Proceedings of the Agile Conference, July 24–29, 218–224.

Yin RK. 1994. *Case Study Research, Designs and Methods*. Sage Publications: Thousand Oaks, CA.

Copyright © 2008 John Wiley & Sons, Ltd.