NTNU

Norwegian University of
Science and Technology

A Case Study of Coordination in Distributed Agile Software Development

Steinar Hole

Master of Science in Computer Science
Submission date: June 2008
Supervisor: Torgeir Dingsøyr, IDI

Norwegian University of Science and Technology
Department of Computer and Information Science

Problem Description

Today, many companies develop software in project teams with members located in different parts of the world with different culture. This assignment's objective is to study the phenomenon of global software development. For theses, it will be of interest to do field research in global projects.

Assignment given: 15. January 2008
Supervisor: Torgeir Dingsøyr, IDI

Abstract

Both global software development and agile approaches have gained significant popularity. Companies even show interest in applying agile approaches in distributed development to combine the advantages of both approaches. This is done despite their differences in key tenets. In their most radical forms, agile and global software development can be placed in each end of a plan-based/agile spectrum because of how work is coordinated. This study describes how four global software development projects applying agile methods coordinate their work. The findings show that there are at least three approaches to distributed Scrum; local Scrum independent of remote team's approach, multiple Scrum teams coordinated with Scrum of Scrums and geographic transparency and a single distributed Scrum team. It was also found that trust is needed to reduce the need of standardization and direct supervision when coordinating work in a global software development project, and that electronic chatting supports mutual adjustment. Further, co-location and modularization mitigates communication problems, enables agility in at least part of a global software development project, and renders the implementation of Scrum of Scrums possible. Proper mechanisms to provide transparency are needed to achieve mutual adjustment.

Preface

The University of Science and Technology (NTNU) has provided me with subjects focusing mostly on traditional approaches to software development during this five year master program. When starting my preparations for my master's thesis, I was curious to explore what innovative approaches might exist in the industry. Half a semester was invested in a pre-study, a literature review on "Communication in global software development - Joining virtual teams and agile software development". But there is always more to explore. The full time of the next semester was dedicated to this thesis.

The industry is ahead of research in both global software development and agile development. My supervisor, Torgeir Dingsøyr, advised me to do a case study to investigate how they correlate. Nils Brede Moe, my advisor, introduced me to Mintzberg's coordinating mechanisms and helped me realize the benefits of using them as a framework when analyzing distributed agile projects. They have both been invaluable sources of knowledge, critique and advices. Nils Brede Moe and I did even co-author a research paper that was base on this thesis, "A case study of coordination in distributed agile software development." The research paper was accepted by the European Software Process Improvement (EuroSPI) conference in Dublin, September 2008, and I'm quite thrilled to be allowed to present my findings at the conference.

I really appreciate the invaluable input received from the project participants of the investigated companies; it was nice to get to know you. A big thanks to the proofreaders of both the paper and my thesis, respectively Hamish Barney and Odd Nordland, and Martin Stige and Marianne Prestvik Hole. The biggest thanks go to Nils Brede Moe for his patient help, interesting discussions, inputs and questions, and to Torgeir Dingsøyr for providing this exciting challenge to me. This research is supported by the Research Council of Norway under Grant 174390/I40.

Trondheim, June 10th 2008

_____

Steinar Hole

Contents

1 Distribution and agility **1**
1.1 Increasing distribution ... 1
1.2 Introducing agility ... 2
1.3 The challenge ... 3
1.4 Outline ... 3
2 Coordinating through agility **4**
2.1 Searching for knowledge ... 4
2.2 Agile methods and Scrum ... 6
Common traits ... 6
Scrum ... 8
2.3 Coordinating mechanisms ... 11
Coordinating distributed software development ... 12
Coordinating mechanisms in Scrum ... 14
2.4 Transparency and awareness ... 16
Transparency and coordination ... 17
Transparency and Scrum ... 18
3 Research design and approach **19**
3.1 Scope ... 19
3.2 Study context ... 20
3.3 Data sources and analysis ... 20
4 Agility in global software development projects **22**
4.1 Project India I ... 22
Coordinating software development work in the India I project ... 22
Transparency ... 23
4.2 Project India II ... 24
Coordinating software development work in the India II project ... 24
Transparency ... 25
4.3 Project Eastern Europe ... 25
Coordinating software development work in the Eastern Europe project ... 26
Transparency ... 27
4.4 Project USA ... 27
Coordinating software development work in the USA project ... 28
Transparency ... 30
5 Discussion **31**
5.1 Coordination in agile global software development ... 31

5.2 The effect of transparency 34
6 Conclusion and future work **36**
A References **37**
B Glossary **40**
C List of figures **42**
D List of tables **43**
E Research Plan **44**
F Interview guides **48**

“It is not by consolidation, or concentration of powers, but by their distribution, that good government is effected.”

Thomas Jefferson (1762-1826)

1 Distribution and agility

Jefferson argues that collaboration between powers is better than centralized and concentrated control. Although his statement is about democracy, it has a growing relevance for software developing organizations. Coordination through centralized control has been the suggested best practice for global software development (Cataldo et al., 2007, Sangwan et al., 2006, p. 96), but later approaches seem more eager to distribute the power (Moe et al., 2008). Introduction of agile methodologies defy the established form of government by focusing on self-organizing teams. Distribution of power has become a relevant topic with distributed agile software development.

1.1 Increasing distribution

Globalization[1] force organizations to consider global competition and resources (Levitt, 1983). Outsourcing has been a buzzword since the mid 1980s, allowing organizations to focus on their core competencies while handing over responsibility for certain processes to others (Hirschheim and Dibbern, 2006). During the 1990s, better global information technology infrastructure led to outsourcing beyond the organization’s national border, more precisely dubbed offshore outsourcing. Global inter-organizational software development, including outsourcing, subcontracting and partnerships, is becoming increasingly common (Paasivaara and Lassenius, 2004).

Multinational organizations are another case where services are distributed beyond national borders. The distinction from outsourcing coined the terms internal offshoring and offshore insourcing. “Offshore” stems from US organizations mostly seeking overseas, like Ireland or India. The term has later been recognized as a description of any sourcing beyond a national border. The more precise term “nearshoring” has later been introduced to describe offshoring between countries where the time zone difference is insignificant. Prickladnicki, Audy et al. (2007) make a distinction between the terms by showing how they describe different distributions related to organizational and geographical borders (Table 1).

[1] "Globalization is a process of interaction and integration among the people, companies, and governments of different nations, a process driven by international trade and investment and aided by information technology." (www.globalization101.org). A more academic definition from the social sciences would be Robertson’s: “Globalization as a concept refers both to the compression of the world and the intensification of consciousness of the world as a whole.” Robertson, R. (1992) *Globalization: Social theory and global culture*, Sage Publications Inc.

Table 1: Organizational and geographical distributions (Prikladnicki et al., 2007)

	National	Global
Internal	**Shared Services** or **Internal Domestic Supply**	**Internal Offshoring** or **Offshore Insourcing**
External	**On-shore Outsourcing** or **Outsourcing**	**Offshore Outsourcing**

Many organizations turn toward global software development, software development distributed beyond national borders, in an attempt to produce cheap higher-quality software with the shortest development cycle possible (Moe and Smite, 2007). Global software development is becoming the norm by promising potential advantages like global resources, attractive cost structures, round-the-clock development and closeness to local markets (Damian and Moitra, 2006). The promises are intuitive. To unleash the potential, methods and tools for distributed software development are designed to enable geographically dispersed team members to share programming tasks and development practices (Canfora et al., 2006). Methods and tools are also needed to mitigate global software development problems related to coordination, communication, control (Agerfalk and Fitzgerald, 2006), and increased complexity (Carmel and Agarwal, 2001).

1.2 Introducing agility

There is a demand for approaches able to deal with the increasing complexity of software development, because coordination becomes more difficult when complexity increases (Kraut and Streeter, 1995). A family of potential approaches that has received a lot of attention from software engineers and software researchers the later years has adopted the term “agile” (Abrahamsson et al., 2003). Agile software development is introduced as a software development approach promoting teamwork, innovation, flexibility, and communication (Agerfalk and Fitzgerald, 2006).

Agile development approaches and global software development approaches differ significantly in their key tenets, e.g. regarding coordination mechanisms (Ramesh et al., 2006). Global software development focuses on command-and-control and formal communication. The desired organizational structure is mechanistic, which means that it is bureaucratic with high formalization. Agile or change-driven development focuses on leadership-and-collaboration and informal communication. The desired organizational form is organic, which means that it is flexible, participative, and encourages cooperative social action. Therefore, applying agile principles to global software development marks an intersection of two seemingly incompatible approaches. Still, Ramesh et al. (2006) demonstrate how a balance between agile and distributed approaches can help meet the challenges with incorporation of agility in distributed software development.

Despite the differences, there is a growing interest in assessing the viability of using agile practices for distributed teams (Agerfalk and Fitzgerald, 2006). Several reports claim that it can be done successfully (Berczuk, 2007, Farmer, 2004, Fowler, 2003, Holmstrom et al., 2006, Nisar and Hameed, 2004, Korkala and Abrahamsson, 2007, Ramesh et al., 2006, Sulfaro, 2007, Sutherland et al., 2007).

1.3 The challenge

This study has been motivated by the work of Ramesh et al. (2006) and Sutherland et al. (2007) to investigate how work is coordinated when introducing agile methods in a global software development environment:

1. How are tasks coordinated in global software development teams applying agile methods?
2. How does the level of geographical transparency affect the level of mutual adjustment?[2]

While Ramesh et al. (2006) encourage a balance between mutual adjustment and direct supervision in distributed agile software development, Sutherland et al. (2007) claim that mutual adjustment can be achieved by proper geographical transparency. The intention of this research is to explore coordinating mechanisms in teams that implement agile practices in distributed software development, identify changes in the way work is coordinated, and consider geographical transparency's impact on that coordination.

1.4 Outline

This report seeks to answer the research questions through a literature review and a multiple case study. The material is organized like this:

- **Coordinating through agility** (p. 4) presents a literature review on agile software development and coordinating mechanisms.
- **Research design and approach** (p.19) describes the research method in detail.
- **Agility in global software development projects** (p. 22) presents the results from a multiple case study on agile methods and practices applied to four global software development projects.
- **Discussion** (p. 31) discusses the research questions.
- **Conclusion and future work** (p. 36) summarizes the findings, concludes this research and states further investigations to undertake.

[2] Geographical transparency is the availability of appropriate knowledge or information for coordinating across multiple sites (see 2.4 Transparency and awareness p. 16).

2 Coordinating through agility

A literature review of research regarding global software development, agile methods and virtual teams was carried out as a preparation for this thesis. Literature reviews are fortunately able to give a deeper understanding while also contributing in creation of new theories (Webster and Watson, 2002). While the focus of the preparing study was on the communication challenges when joining virtual teams and agile software development, the review has been adapted to focus on coordinating mechanisms and transparency. This section presents the reviewed theories to establish a foundation that both illuminates the research questions and underlie the later presented multiple case study. Parts of this section are reproduced from the preparation study.

2.1 Searching for knowledge

The articles for the review were collected in four steps. First, a set of seven articles (Agerfalk and Fitzgerald, 2006, Borchers, 2003, Damian and Moitra, 2006, Herbsleb and Mockus, 2003, Herbsleb et al., 2005, Krishna et al., 2004, Ramesh et al., 2006) were provided by the advisors before the start of the study. Several of these are introductory articles giving an overview of global software development primarily. Second, every article in a special issue of IEEE Software (Volume 23, Issue 5) on global software development was browsed for the concepts of agility, virtual teams and communication. The same was done with the papers submitted to IEEE International Conference on Global Software Engineering 2006 and 2007.

Browsing and reading the articles inspired the construction of a keyword table (Table 2) that was used in the third step, a search of online databases (Table 3). The search strings were composed of words across the concepts treated, but not in an exhaustive way. This was considered appropriate because the goal of the review was to gain a proper founding understanding of the various concepts and their relations. The keyword table and list of articles were dynamically expanded throughout the review. Fourth, the references of the articles that were considered most relevant for the goal of the review were searched for even more relevant articles.

Table 2: Concepts and keywords used when searching for literature

Global	Multinational, international, offshore, offshoring, outsource, outsourcing, globalization, globalisation, distributed, dispersed, distance, culture, temporal, timezone, time zone
Agile	XP, Xtreme Programming, Extreme Programming, Scrum, DSDM, Dynamic Systems Development Methodology, Adaptive Software Development, Crystal (Methods), Feature-Driven Development, Pragmatic Programming, Lean Development, Agile method, Agile methodology, light weight
Virtual teams	Virtual organization, geographically dispersed team, information technology, technology mediated
Communication	Coordination, collaboration, communication theory, communication theories, communication mode, communication model
Coordination	Coordinating mechanism, mutual adjustment, direct supervision, standardization, transparency, agile transparency, geographical/vertical/horizontal transparency, awareness

Table 3: Literature databases used when searching for literature

ACM Digital Library	portal.acm.org
BIBSYS Ask	ask.bibsys.no
Google Scholar	scholar.google.com
IEEE Xplore	ieeexplore.ieee.org
ISI Web of Science	Portal.isiknowledge.com/portal.cgi?DestApp=WOS

Articles were included if their main focus was on global software development, agility, virtual teams, communication, coordination, or a combination of these. Communication is a mechanism underlying coordination and control (Carmel and Agarwal, 2001), and was therefore considered a concept that embraced these as well.

Articles were considered to be of higher prestige if published in “Communication of the ACM”, “IEEE Software”, “MIS Quarterly” or “Organization Science”, somewhat less prestige if published in “IEEE Transactions on Software Engineering”, “Journal of Management”, “Journal of Management Information Systems”, and conference proceedings were considered least prestigious although above un-reviewed publications. Less prestigious material would only be used if the article had a well founded base for its claims, and was preferably commonly cited relative to its date of publication (close to or above ten cites per year).

Qualitative, quantitative and industry experience papers were included, while expert’s opinion articles were less regarded. When considering agility, the main focus was on papers that dealt with Scrum. Studies that were clearly not about agility, global software development or coordination of work were excluded, together with papers lacking rigor, credibility and relevance. No particular review strategy was made beyond reading literature continuously with a constant focus on how well the concepts merged relevant to communication, coordination and collaboration.

2.2 Agile methods and Scrum

A large variety of software development approaches emerged during the last century, while only a few of them became mainstream (Abrahamsson et al., 2002). These traditional approaches are characterized by detailed planning and documentation. Some claim that they merely present a fictional image of control, and that the provided tracking of status is only symbolic (Nandhakumar and Avison, 1999). Agile methods can be seen as a reaction to plan-based or traditional methods (Dyba and Dingsoyr, 2008)

In February 2001, the "Manifesto for Agile Software Development" [3] was created by seventeen people with desires to find alternative approaches to software development. Each of them played a prominent part in the opposition of the prevailing software development processes, which they considered rigid, heavyweight and too focused on documentation. Their response, summarized in the manifesto, clarifies their focus by valuing:

- **Individuals and interactions** over processes and tools
- **Working software** over comprehensive documentation
- **Customer collaboration** over contract negotiation
- **Responding to change** over following a plan

They also clarify that they do not disregard the unemphasized items on the right; they just value the emphasized items on the left more. This also indicates their view that there are no contradictions between e.g. following a plan and responding to change.

Agile software development comprises a number of practices and methods (Erickson et al., 2005, Cohen et al., 2004, Abrahamsson et al., 2002). Among the most known and adopted agile methods are Extreme Programming (XP) (Beck and Andres, 2004) and Scrum (Schwaber and Beedle, 2001). XP focuses primarily on the implementation of software, while Scrum focuses on agile project management (Abrahamsson et al., 2003). In this study the focus is on Scrum since Scrum is an agile approach to the management of software development projects (Erickson et al., 2005, Cohen et al., 2004, Abrahamsson et al., 2002), and thus focuses on the coordination of work.

Common traits
Agility is related to concepts like quickness, dexterity or nimbleness. To achieve this agility, the software development process is stripped of as much heaviness and rigidness as possible (Erickson et al., 2005). The gained agility is proposed to handle change responsively. This leads to one of the notions used to describe agile methodologies; they are characterized as being change-driven (Moe et al., 2008).

Agile methodologies intend to allow system requirements to change during development, and welcome this change through frequent communication and interaction. Because a detailed and complete plan up front demands a complete knowledge about every aspects of the application(s) under development and its/their interactions with the environment, agilists consider it a better approach to elicit the details in transit, and even allow changes to the architecture as the fundamental understanding grows (Abrahamsson et al., 2002).

[3] www.agilemanifesto.org

The response to change has not been built into the traditional plan-driven software development processes (Nerur et al., 2005). The waterfall model places the parts of software development in successive phases, starting with requirements before design, implementation, testing and maintenance (Pressman, 2005, p. 47). Another approach, the V-model, attempts to improve the relationships between the phases of the waterfall model by focusing on how the later phases fulfill the earlier, e.g. how acceptance testing verifies that the requirements are met and integration testing verifies that the architecture design is followed. The spiral model is close to the waterfall model in that it keeps the phases apart, but the phases are recurring by returning to the first phase after the last, resulting in an iterative approach (Pressman, 2005, p. 54). Introduction of iterations increase the flexibility of the software development process, but the traditional software development processes still have a heavy focus on up-front planning and tracking of progress according to the plan. The focus on plans leads to the notion of plan-driven software development as a term describing the traditional approaches. Nerur et al. (2005) provide a comparison of traditional and agile software development in Table 4.

Table 4: Comparison of traditional and agile development (Nerur et al., 2005)

	Traditional	**Agile**
Fundamental Assumptions	Systems are fully specifiable, predictable, and can be built through meticulous and extensive planning.	High-quality, adaptive software can be developed by small teams using the principles of continuous design improvement and testing based on rapid feedback and change.
Control	Process centric	People centric
Management Style	Command-and-control	Leadership-and-collaboration
Knowledge Management	Explicit	Tacit
Role Assignment	Individual – favors specialization	Self-organizing teams – encourages role interchangeability
Communication	Formal	Informal
Customer's Role	Important	Critical
Project Cycle	Guided by tasks or activities	Guided by product features
Development Model	Life cycle model (Waterfall, Spiral, or some variation)	The evolutionary-delivery model
Desired Organizational Form/Structure	Mechanistic (bureaucratic with high formalization)	Organic (flexible and participative encouraging cooperative social action)
Technology	No restriction	Favors object-oriented technology

How can agile software development eliminate the rigidness and heavyweight of plan driven software development? According to Cockburn (2002, p. 178-179), the following principles increase the chance of success:

1. Two to eight people in one room
2. On-site usage experts
3. Short increments

4. Fully automated regression tests
5. Experienced developers

The first principle enhances communication and community, while the second results in short and continuous feedback cycles. This shows how agile methodologies focus on close collaboration through continuous communication, both within the team and with the customer or someone in a customer role. Plans are evaluated and reconstructed frequently. The third principle indicates how agile software development uses short iterations to complete the product in parts, allowing the evaluation of finished parts to guide the development of the rest of the product. It also enables quick repair during the development, instead of a long testing phase at the end. The fourth principle is also concerned with continuous improvement, while the fifth shows how agilists value people.

Abrahamsson et al. (2002) claim that software development is agile when it is incremental, cooperative, straightforward and adaptive. It is incremental when the product is developed in rapid cycles with small software releases. When customer and developers collaborate with constant communication, it is cooperative. Straightforward relates to how easy the method itself is to learn and modify, while adaptive relates to the ability to make last moment changes.

Despite the common focus, a wealth of methodologies has emerged, ranging from lists of practices to frameworks applicable to different contexts. In this study, only one of these approaches are selected and used as a representative of agile software development.

Scrum

Scrum was created with the perspective of industrial process control in mind, treating development processes that are not completely defined as a black box (Schwaber, 1995). The concept of black box is that one cannot see how input is converted to output within the black box, resulting in unpredictable output. Scrum's underlying approach is therefore to accept the existence of uncertainty and try to control this uncertainty, by managing both defined and black box processes.

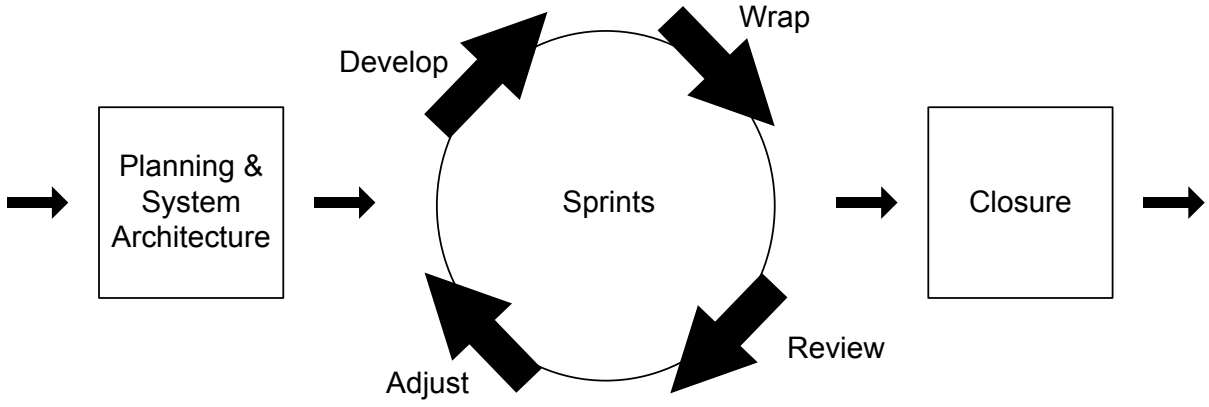


Figure 1: The Scrum methodology (Schwaber, 1995)

The planning and system architecture phase, and the closure phase are well defined. Inputs, outputs and processes are controllable through the defined framework of these phases. The planning phase may have multiple iterations, but are considered linear, like the closure phase. As we can see from Schwaber's (1995) Figure 1, the sprint phase comes between the well defined first and last phases. The sprint phase contains the uncertainty and control of these

uncertainties. Iterating through sprints demand continuous control to "avoid chaos while maximizing flexibility" (Schwaber, 1995). The sprints are non-linear and flexible, allowing quick response to change in the environment, like change in resources, quality demands, requirements or the market. A more detailed description of each phase follows according to Schwaber and Beedle (2001, p. 31-88) and Abrahamsson et al. (2002).

Planning To begin with, requirements are placed on a product backlog, which is a prioritized list of functionality desired in the product. It is considered unnecessary and even undesired to strive for a complete list of functionality to begin with, as more items are placed on the product backlog when identified during the process. The initial requirements should reflect the goal of the product. The constraints of the project are also defined in this phase. Schedule and cost are estimated and project teams are defined and allocated. Some time should also be used to assess risk and assign controls to the identified risks.

The architecture of the system is created in the planning phase. It is not desirable to make too much change to the architecture later, as changes to the architecture implies changes to the code that should comply with the architecture. Therefore, changes to the architecture should first and foremost relate to implementation of new functionality. Architecture is also created on the lower levels during sprints, but the system architecture and high level design should be done up front. This includes conceptualization and analysis related to system architecture and high level design.

Sprints Software is developed by the self-organizing team in increments called "sprints" (Abrahamsson et al., 2002). The sprint duration varies ideally between two and four weeks. Each sprint starts with a sprint planning meeting, where a release is defined based on the items on the product backlog. The product backlog comprises a prioritized and constantly updated list of business and technical requirements for the system being built or enhanced (Abrahamsson et al., 2002). Backlog items can include features, functions, bug fixes, requested enhancements and technology updates. Multiple stakeholders can participate in generating product backlog items, such as customer, project team, marketing and sales, management and support (Abrahamsson et al., 2002).

The product owner decides which backlog items should be developed in the following sprint. Usually, items that are related are selected, broken down into more detailed backlog items and put on a sprint backlog, which then contains a list of the functionality to implement during a single iteration. Only items the team can commit to complete during one sprint are selected. Items from the product backlog are broken down in smaller items for the sprint backlog and each task is estimated. The team does this in close collaboration (Abrahamsson et al., 2002).

Development begins after the sprint planning meeting. The team coordinates on a daily basis. Every day contains one daily 15 minutes Scrum meeting where each team member addresses three questions:

- What did you do since last Scrum meeting?
- What are you going to do until next Scrum meeting?
- What are the obstacles in your way?

The goals of the Scrum meeting include focusing the effort of developers on the backlog items, communicating the priorities of backlog items to team members,

keeping everyone informed of team progress and obstacles, resolving obstacles as quickly as possible, tracking progress in delivering the backlog functionality, and addressing and minimizing project risk (Rising and Janoff, 2000).

At the end of each sprint is the sprint review where the team presents their accomplishments, issues and problems are resolved and the product backlog is revised by adding, changing or removing items. As stated by Rising and Janoff (2000), "anything can be changed" at the end of a sprint. In addition to the sprint review that is focused on the product developed, Berczuk (2007) suggests a retrospective immediately after the review. The retrospective's aim is to evaluate the process, identifying what is working and what is not. Customer presence is encouraged during planning and reviews and occasionally during daily scrums and retrospectives. Figure 2 illustrates the flow of work involved in sprints.

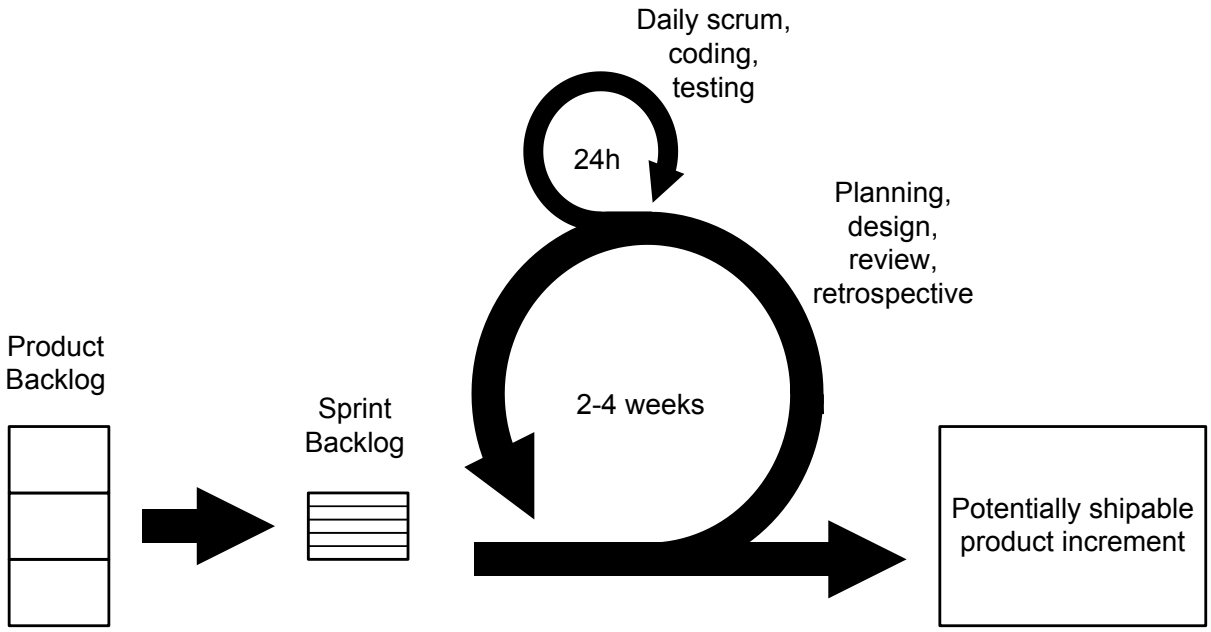


Figure 2: Cycles in the Scrum methodology (Schwaber, 2004, p. 6)

Closure means preparation for general release. Final integration and user documentation of the product is completed. Sometimes a final integration-sprint is run to prepare the product for general release.

Roles Although the Scrum team is considered autonomous and self-organizing, some roles are needed to maintain the process. The person(s) filling a role may change from time to time.

- **The Scrum master** replaces the traditional project manager and is a facilitator or coordinator in charge of solving problems that prevents the Scrum team from working effectively, as well as removal of impediments to the process and shielding of the team from unnecessary external influence (Erickson et al., 2005, Cohen et al., 2004, Abrahamsson et al., 2002).
- **The product owner** is selected by the Scrum master, the customer and the management to be responsible for the project by managing and providing the product backlog. He must also take part in estimation.
- **The customer** takes part in evaluation of backlog items.
- **The Scrum team** (5-9 people) consist of designers and developers that are empowered to organize themselves to complete the tasks they commit

to for each sprint, through partaking in effort estimation, creation of sprint backlog and revision of the product backlog.

2.3 Coordinating mechanisms

Mintzberg (1989, p. 100) states that every organized human activity involve the division of labor into various tasks, as well as the coordination of those tasks to accomplish the activity. The tasks need to be coordinated because of various dependencies between them, like a need for access to the same resources. Coordination is the management of these dependencies (Malone and Crowston, 1994). If there are no dependencies, there is nothing to coordinate. But if a task depends on the result of another task, or if the execution of a task depends on the expertise of the person that performs the task, the dependencies must be managed. Coordination of work is therefore an important aspect of teamwork and team leadership (Salas et al., 2005).

There are three basic coordinating mechanisms that seem to describe the fundamental ways in which organizations can coordinate their work (Mintzberg, 1989, p. 101):

- Mutual adjustment – which achieves coordination by the simple process of informal communication
- Direct supervision – in which coordination is achieved by having one person take responsibility for the work of others whose work interrelates, by issuing instructions and monitoring their actions, and thus enforcing control
- Standardization – of which there are four types: Standardization of work processes, standardization of outputs, standardization of skills (as well as knowledge) and standardization of norms

While no organization can rely on a single coordinating mechanism, mutual adjustment and direct supervision are almost always important. Still, all the mechanisms mentioned will typically be found in every reasonably developed organization (Mintzberg, 1989, p. 103).

A further explanation of the various standardization mechanisms follows according to Mintzberg (1989, p. 101-105). Standardization of work processes achieves coordination by specifying the work processes of people whose work interrelates, e.g. by assembly instructions or job descriptions. When outputs are standardized, coordination is achieved by specifying the results, e.g. an architect’s plan that specifies the resulting house. Standardization of skills and knowledge achieve coordination by training the workers to handle different types of tasks, e.g. a surgeon and an anesthetist. Finally, standardization of norms achieves coordination by the norms that pervade the work, i.e. how the belief in values like ethnic supremacy justified slavery. The change of such a norm may change how work is coordinated.

The three basic coordinating mechanisms fall into an order regarding complexity of organizing (Mintzberg, 1989, p. 101-102). Simple tasks with simple dependencies are easily coordinated by mutual adjustment, but direct supervision tends to be added and take over as the primary means of coordination when work becomes more complex. Further complexity leads to coordination by standardization (of work processes or norms, then of outputs or of skills), while even more complexity revitalize mutual adjustment as the

primary coordinating mechanism (Figure 3). Mintzberg (1989, p. 102) consider it a paradox that mutual adjustment is the mechanism best able to deal with both the simplest and the most complex forms of work.

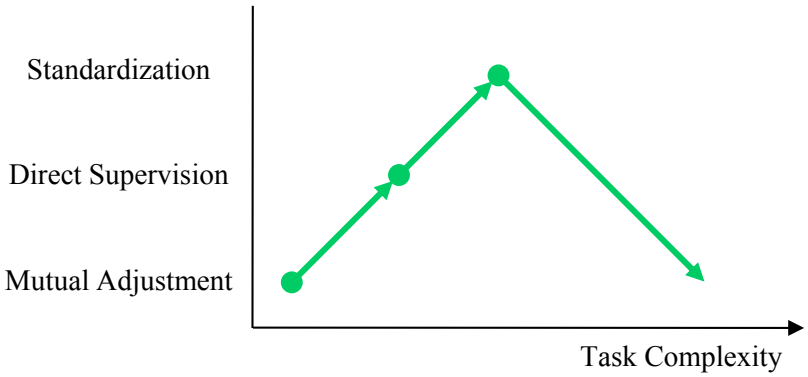


Figure 3: The change of primary coordinating mechanism with increasing task complexity (Smite et al., 2008).

Coordinating distributed software development

Coordination, together with communication and collaboration, are recognized as the key enablers of software development processes (Layman et al., 2006). While there is no single cause for the problems in software development, a major factor is the problem of coordinating activities while developing large software systems (Kraut and Streeter, 1995). Kraut and Streeter (1995) mention scale of software projects, inherent unpredictability of software specifications and tasks as well as the interdependence of software components as some of the factors that lead to the necessity of efficient coordination between the different work groups involved in the development process (Amrit, 2005).

Even with a heavy demand for successful coordination of software development, the challenge is even bigger when distribution is introduced. The key difference between collocated and global software development is coordination over distance (Herbsleb, 2007). The distance has unfortunately a negative impact on the coordination (Herbsleb and Mockus, 2003), because many of the mechanisms that coordinate the work in a collocated setting are absent or disrupted in a distributed project (Herbsleb, 2007). Herbsleb consider this the fundamental problem of global software development.

Global software development has traditionally relied mainly on formal mechanisms (coordination by standardization), which exploit detailed architectural design and plans to address impediments to team communication induced by geographical separation and reduce the need for communication (Ramesh et al., 2006, Agerfalk and Fitzgerald, 2006). The formal mechanisms are chosen because of the traditional perspective that promotes a mechanistic production-line approach to software development (Dyba, 2000, Nerur and Balijepally, 2007). Boden et al. (2007) claim that, while there are two approaches to solve the coordination problems, namely to reduce the need for frequent informal communication (mutual adjustment) or to ease the informal communication by technical means; a stronger focus on formalization is not a solution.

Cataldo et al. (2007) provide a list of coordination mechanisms for distributed software development that represents the best practices of the software engineering literature. The mechanisms mentioned are either provided by standardization and direct supervision, or means to achieve mutual adjustment

(Table 5). As we can see from the table, there is a balance between all three coordinating mechanisms. Cataldo (2007) claims that lateral communication (mutual adjustment) are beneficial even in cases where the level of interdependency between remote teams are low, due to coordination breakdowns that still occur when the proposed mechanisms are in place. It seems like coordination breakdowns fall back on mutual adjustment to re-establish coordination.

Table 5: Best practices from literature on distributed software development (Cataldo et al., 2007). Conformity with mutual adjustment (MA), direct supervision (DS) and standardization (S) is based on Mintzberg (1989, p. 101).

Mechanism	**Purpose**	**MA**	**DS**	**S**
Centralized structure	Centralize critical decisions and establish clear paths of communication		X	
Early identification of dependencies	Reduce dependencies amongst tasks assigned to remote teams.		X	X
Documentation	Reduce the need to communicate amongst remote teams by having access to detailed design decisions.			X
Change, configuration and integration management processes	Identify relationships, manage, control, audit and report on the changes made to the software.		X	X
Periodic commits	Increase awareness by making ongoing changes to the system available to all the remote teams.	X		
Daily builds	Reduce the potential for integration problems by identifying them early.	X	X	
Communication tools	Allow for exchange of information amongst teams when other coordination mechanisms are not sufficient.	X		
Periodic meetings	Status and definition of tasks. Relay information from remote teams to others.	X		

As a contrast to the traditional approaches, agile development relies on people and their creativity rather than on processes (Cockburn and Highsmith, 2001), and emphasizes informal communication (mutual adjustment) as the primary coordinating mechanism (Nerur et al., 2005). The major challenge of applying agile methods or practices in a global software development context is to balance the coordinating mechanisms (Figure 4). However, there are obvious conflicts when trying to balance mutual adjustment, direct supervision and standardization, as direct supervision and standardization overrides mutual adjustment.

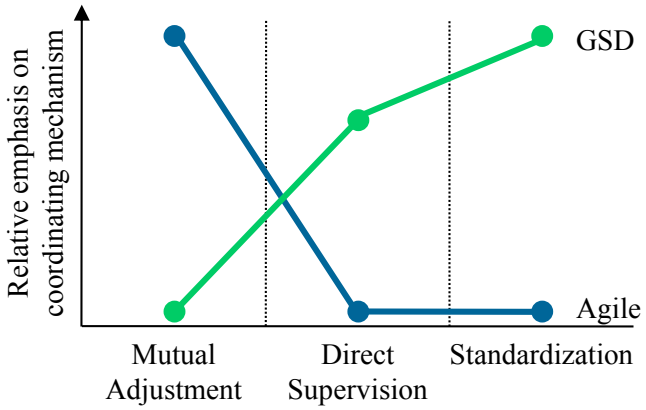


Figure 4: Relative emphasis on coordinating mechanisms: Agile development relies purely on mutual adjustment, while global software development (GSD) emphasizes standardization and some direct supervision.

The benefits of agile development considering coordination are the advantage of mutual adjustment as the best coordinating mechanism for complex coordination, as mentioned above. Since distribution of the software development increases complexity (Carmel and Agarwal, 2001), mutual adjustment should be considered as the best coordinating mechanism. This reinforces the challenges of communication in distributed software development, as mutual adjustment is made more difficult by distribution, due to communication impediments. It is also worth mentioning that Mintzberg did not seem to consider distributed work when he claimed that mutual adjustment is best for the most complex coordination, and therefore a balance with other coordinating mechanisms might be appropriate.

Coordinating mechanisms in Scrum

Mintzberg (1989, p. 101-104) says that the standardization of work processes achieves coordination by specifying the work processes of people carrying out interrelated tasks. He exemplifies by mentioning time-and-motion studies, which seek to make processes more efficient through removal of unnecessary actions; resulting in a specific description of how to perform a repeatable task. He describes standardization of work processes as the imposition of operating instructions, job descriptions, rules, regulations and the like.

The concept of standardization of work process can be better understood by considering its substitutes; standardization of skills and knowledge, and standardization of norms. Instead of standardizing by imposing rules through a detailed description of how to perform work, standardization of skill and knowledge makes sure the standards are learned and therefore inherent in the person that works (Mintzberg, 1989, p. 104). Likewise, the internalization of standards through norms ensures that members of the organization acts and makes decisions in accordance with the common belief, the standard (Mintzberg, 1989, p. 104). In the light of these, according to Mintzberg (1989, p. 104), substituting standardizations, the standardization of work process becomes another way to lock the actions of people to a pre-determined standard.

It is arguable that Scrum (or any other agile approach) is such a standardized process, from which one would conclude that standardization, and not mutual adjustment, is the primary coordinating mechanism of Scrum. It is also clear that agile development imposes a very high level of discipline on the performers. Examples would be the discipline needed to perform daily meetings, valuable

feedback, testing and reviews. The need for such discipline indicates quite some rigidness in agile development as well.

It could even be argued that mutual adjustment and direct supervision are standardized processes, and then conclude that the only coordinating mechanism is standardization. This is not in accordance with Mintzberg (1989, p. 103), who consider that mutual adjustment and direct supervision live side by side with each other and with standardization, although the emphasis varies. Coordinating mechanisms used in an organization may also vary depending on the organizational level, e.g. mutual adjustment within a team, and direct supervision between team and supervisor. Mintzberg (1989, p. 103, 110) states that the primary reliance on a specific coordinating mechanism is indicative of what kind of organization it is, while there still are other coordinating mechanisms in use. It is therefore clear that, while there might exist standardization that facilitate mutual adjustment or direct supervision, they are still distinguished coordinating mechanisms.

Since the distinction between the three coordinating mechanisms is justified, it is important to identify the one emphasized in agile development. It is true that agile development seeks to provide processes for software development, but it is important to remember that the provided processes are dynamic within the frameworks provided. While practices like pair-programming and continuous integration (Extreme Programming (Beck and Andres, 2004)) are standardized frameworks for action, they are also dynamic in the sense that the processes should be continually optimized. More important is their purpose to facilitate mutual adjustment. Since no coordinating mechanism can prevail on its own (Mintzberg, 1989, p. 103), it is logical that agile development also contain traces of standardization, regardless of which coordinating mechanism is primary.

The traditional software development processes are highly focused on standardization of output, and somewhat on standardization of skill and knowledge, and of work process (Nerur and Balijepally, 2007). This is clearly in contrast to agile development’s focus on continuous adjustment of output, collaboration of people and adaption of process (Nerur and Balijepally, 2007).

Structures that rely on standardization for coordination may be defined as bureaucratic, those that do not as organic, and vice versa (Mintzberg, 1989, p. 104). An organization’s structure is organic relative to how organic its environment is (Mintzberg, 1989, p. 108). Innovative organizations have very organic structures to respond to their organic environment. Mintzberg (1989, p. 113) note that innovative organizations rely on mutual adjustment as the primary coordinating mechanism by welding various units into multidisciplinary teams of experts that coordinate within and between themselves through mutual adjustment.

Innovative organizations disperse power over different decisions to different places in the organization, in contrast to centralization of power. Teams become empowered to coordinate their own work, both internally and related to other teams’ work. This is similar to self-organizing teams, typical for agile development (Nerur and Balijepally, 2007). When software organizations deploy experts in multidisciplinary teams that carry out projects in a complex and dynamic environment, they can be classified as innovative, and mutual adjustment should then be the most important coordinating mechanism (Moe and Smite, 2007).

Mintzberg (1989, p. 105) also mention liaison devices related to mutual adjustment. Liaison devices refer to a whole series of mechanisms used to encourage mutual adjustment within and between units. They range from liaison positions to fully developed matrix structures (Mintzberg, 1989, p. 105). Scrum (and any other agile approach) should therefore be considered a liaison device, as a way to create and improve a dynamic software development process by providing a process framework that encourages mutual adjustment. Processes should not be imposed unless they are needed, which is also the statement of agile development regarding documentation (Nerur et al., 2005). Standardization is discouraged if it is not needed.

On a higher level, Scrum enforces some rigidness concerning how to approach software development. A system of values are laid as a foundation encouraging the organization to value the collaboration of people, embrace change, include the customer and produce useful results. A framework is built upon this foundation. Scrum and other agile approaches present processes that facilitate mutual adjustment. While the processes enforce discipline, even more so than in traditional development, they are not detailed specifications of step-by-step solutions to known problems, which is what Mintzberg talk about when he mention the standardization of work process. Still, one could consider the agile approaches to be high-level standardization of work process, and to a degree, standardization of norms (i.e. values and beliefs). The question is then: “Which coordinating mechanism is dominating in agile development?” Comparing the ideology behind agile development; flexibility in response to a complex changing environment, to Mintzberg’s description of innovative organizations, it is clear that mutual adjustment should be the primary coordinating mechanism in agile development. It can therefore be considered the goal of agile development to achieve mutual adjustment, and the conclusion is that the theory on agile development emphasizes mutual adjustment.

2.4 Transparency and awareness

Little research can be found on transparency. The concept is frequently used, but has seldom got the full focus, and it is usually not explained, as the authors often expect the reader to grasp the concept intuitively. This section will therefore try to create an understanding of the concept by referring to how it is used in the literature.

Transparency or visibility is a concept that is used to describe whether the appropriate knowledge or information for coordinating is available or not. The concept is frequently used in close relation to coordination, primarily through direct supervision and mutual adjustment, where actors need to be made aware before they react. Transparency makes everyone fully aware of productivity, progress toward goals, competence of people to do their jobs, willingness of people to work together toward project goals and the ability to build completed products on time (Schwaber, 2007, p. 18). Geographical transparency refers to transparency across multiple sites, as used by Sutherland et al. (2007).

Transparency as a concept is often applied in one of two dimensions, vertical or horizontal. The vertical dimension often relates to direct supervision and encourages the lower levels of an organization to provide the proper knowledge for coordination and control on higher levels. The horizontal dimension relates primarily to mutual adjustment within groups, where awareness, a consequence of transparency, is the main focus of research.

The common definition of awareness is provided by Dourish and Bellotti (1992): “Awareness is an understanding of the activities of others, which provides a context for one’s own activities”. Awareness enables team members who are aware of each other’s interdependent tasks to coordinate better (Damian et al., 2007). The need for awareness therefore depends on the degree to which team members must coordinate because of dependencies (Gutwin et al., 2004).

Herbsleb (2007), who consider coordinating at a distance to be the main challenge of distributed work, claims that it is the reduced level of communication that in turn reduces the team’s awareness and therefore impede coordination. Damian et al. (2007) agree that the awareness needs of distributed teams appear to be greater than those of small co-located teams.

The lack of awareness results in a lack of context. A person on one site tend to be unaware of what people on other sites are doing day by day, their availability for communication and their concerns (Herbsleb, 2007). The lack of awareness makes it difficult to identify who does what and who has what expertise (Herbsleb and Mockus, 2003). Cataldo et al. (2007) suggest periodic commits of code to the common repository to increase awareness by making ongoing changes to the system available to all the remote teams. Another approach is to use presence awareness technology to ease the initiation of distributed communication, as when a notification of someone’s presence online serves as a reminder (Herbsleb and Mockus, 2003). The need to communicate can also be reduced by increasing awareness through transparency. Introducing transparency can reduce the communication overhead in the long term by making information accessible without the need for explicit communication or inquiries (Sangwan et al., 2006, p. 154-155).

Transparency and coordination

The lack of vertical geographical transparency may result in a feeling of lacking control from management (Smite et al., 2008), which has a negative impact on direct supervision. The team has a much better idea of the current project status than the managers have, but this can be resolved by increasing the transparency (Sangwan et al., 2006, p. 76). Vertical transparency is needed to achieve control in global software development (Layman et al., 2006).

If dependencies are not made visible to someone who can coordinate them, the lack of transparency will postpone the awareness of incompatibilities that otherwise could be managed, making dependencies remain unresolved and in need of further coordination. Discrepancies are commonly not discovered before integration (Herbsleb and Grinter, 1999). A solution that brings transparency of work progress to all partners is short iterations. Both the developers, project managers and customers can frequently get a good picture of how the project is progressing (Paasivaara and Lassenius, 2006).

It is also claimed that informal contact and task awareness are crucial for applying mutual adjustment and direct supervision in distributed software development (Smite et al., 2008). With absolute transparency of communication and documentation, the central team may assume the role of facilitator instead of an authority to report to (Sangwan et al., 2006, p. 155), which is a transfer of coordinating mechanism from direct supervision to mutual adjustment. The term “absolute transparency” indicates that information is both vertically and horizontally available beyond geographical borders.

The main benefit of awareness in a distributed software development project is in simplifying communication and improving coordination of activity. That is why the need for awareness depends on the degree to which developers must coordinate (Gutwin et al., 2004). Teams working on well modularized tasks would therefore have less need for awareness, while teams utilizing mutual adjustment have a high need for awareness and transparency.

A reduced level of geographical transparency will have a negative impact on coordination by mutual adjustment because of the general need for awareness to coordinate (Damian et al., 2007). Task awareness is crucial for applying mutual adjustment (Smite et al., 2008). Increased geographical transparency will enable mutual adjustment as mentioned above.

Transparency and Scrum

Scrum relies on transparency (Schwaber, 2007, p. 155) and makes everything visible to everyone (Rising and Janoff, 2000). For instance progress can be made visible through delivered increments and burndown charts. Sutherland et al. (2007) even claim the success of a geographically transparent Scrum project with 56 developers in multiple dispersed teams. They state that it is difficult to achieve, but possible. Geographical transparency was achieved through a single global build repository, one tracking and reporting tool, daily meetings across geographies and good engineering practices (Sutherland et al., 2007). The daily Scrum meeting is credited as crucial for coordination (Sutherland et al., 2007).

3 Research design and approach

The goal of this research is to understand how the introduction of agility affects coordination of tasks in a global software development environment, and how transparency affects coordination. While the literature review has established an understanding of transparency and coordination, it is important to study practicing teams to verify theory and to reveal further aspects regarding coordinating mechanisms in agile and distributed software development.

This thesis report on a multiple case holistic study (Yin, 2003, p. 39-43), in which one phenomenon was studied in several projects in one company and in one project in another company. In a multiple case study, each case must be selected carefully so that it either a) predicts similar results or b) predicts contrasting results but for predictable reasons (Yin, 2003, p. 47). Option a) was chosen for the three cases with common context, while option b) was chosen for the fourth case in another company.

The case studies were prepared by generating a research plan (Appendix E, p. 44). The research questions were listed in the plan, together with a schedule and hypotheses to support or refute by observations. Yin (2003, p. 29) claims that, according to empirical research methods, a theory can be tested by observance of predictions that are logically derived from the theory. If the predictions are supported, the theory is strengthened, but it is refuted if the predicted results are lacking (Yin, 2003, p. 29). It is therefore important for the validity of empirical case studies to state predictions that can be supported or refuted.

To achieve proper internal validity, propositions were generated for both of the research questions. It was also written what kind of support or lack of support to anticipate. A total of seven and five propositions were prepared for respectively the first and the second research question.

3.1 Scope

Global software development is a diverse activity, ranging from a project manager with a geographically distant team solving a simple task, to multiple teams in multiple locations, cultures and time zones solving complex and interdependent tasks. An equivalent diversity can be found in agile software development, where different methodologies and approaches emphasize different aspects and varying context. This diversity calls for a narrowing of the scope and a clarification of the aspects considered.

It would be a good idea to balance the cases with both internal offshoring and outsourcing, to identify differences in coordinating between the two. This should make it possible to reveal possible differences in coordination due to variations in management approach, and variations in transparency created by organizational borders. Two of the selected cases use internal offshoring, while two of them use outsourcing.

It would also be beneficial if the cases were culturally similar, or at least some of them were culturally similar. That would allow for better identification of cultural impact. The two outsourcing cases have remote teams in India, which should allow for similar culture. It is also beneficial if the cases can be conducted within a single company, because the similar context would reduce the possibilities of

variations in the results caused by external or contextual variations. Three of the four cases are studied in the same company.

To overcome the diversity of agile software development, a suitable agile methodology is chosen. Abrahamsson et al. (2002) have compiled a comprehensive review of literature resulting in definition, classification and comparison of ten agile software development approaches, including Extreme Programming (twelve software development practices taken to the extreme), Crystal family (several related methodologies applicable based on heaviness) and Dynamic Systems Development Method (focused on fixing time and resources allocated, while adjusting the amount of functionality accordingly). Scrum was chosen for this study because of its focus on coordination of work (Section 2.2, p. 6). All the four cases use Scrum or some Scrum practices.

3.2 Study context

This study was done in the context of a larger action research program, where several companies have introduced elements from agile development in response to identified problems. The primary software company is medium-sized with approximately 150 employees in four major departments. The projects studied in this company were all using Scrum for the first time. However, the company was experienced with using global software development.

The second company, from which the last case was selected, is a multinational company with several thousands of employees in departments all over the world. The case study was conducted in a department with responsibility for an underlying core system used by several other departments and affiliates, and the software development is of innovative nature. The company is quite experienced with using both agile methods and global software development.

The case in the second company was selected because of the relatively low level of experience with agile methods in the primary company. The low level of experience is quite understandable granted that Scrum was recently introduced for process improvement. The second company had been using Scrum for three years and their project was therefore considered to be a more mature Scrum project.

3.3 Data sources and analysis

To address the research questions, semi-structured interviews were conducted with the persons most responsible for coordination of work in the four projects, i.e. a Scrum master, a project manager, a product owner and a group manager. One person was selected from each project. The interviews lasted from 30 to 70 minutes, and aimed at understanding how Scrum was applied in a global software development context. The interview guide was based on the three coordinating mechanism as proposed by Mintzberg (1989, p. 101), in addition to questions related to Scrum. The focus was on understanding coordination of work, communication within and between the teams, feedback-sessions, planning and estimation, use of documentation, roles and specializations, and how decisions were made. All the interviews were transcribed.

During the interviews a graph was presented. The graph showed the relative emphasis on the various coordinating mechanisms in global software development and in agile software development. The interviewees were then

asked to indicate the relative level of the various coordinating mechanisms used between local and remote sites in their projects. Their indications were then compared to what they said in the interviews and some were slightly adjusted. The graphs are presented together with the results in the next section (Figure 5, Figure 6, Figure 7 and Figure 8).

While the interviews were the primary source of data for this study, access was given to previously collected data on the primary company. This was data collected through the action research program. The data contained previous interviews with several team members, the Scrum master and the product owner of one of the projects. There were also pictures and an observation log connected to the same project.

The transcripts of the interviews were read through and statements were extracted if they dealt with one of the coordinating mechanism or with Scrum. Other statements were extracted if they illuminated the necessary context of the cases. The statements were then merged to tell an accurate and verifiable story about each project regarding their use of Scrum and coordinating mechanisms.

4 Agility in global software development projects

We now present the three global software development projects under study, how Scrum was implemented in these projects, and how work was coordinated in the projects.

4.1 Project India I

The project is partly outsourced to India. The goal of the project is to develop a system for integrity management of pipelines both offshore and onshore. Today several customers are interested in buying the product, and so far three contracts have been signed. One of the biggest challenges in this project is to align requirements from potential customers from all over the world. Scrum was introduced one year after the project had started.

The project consists of six developers working full time (one is a Scrum master), two GUI designers, one product owner, and one project manager working 50% on this project. Four of the developers are situated in India together with one tester. To improve communication one of them is in periods moved to Norway.

The sprints usually last three weeks, and ends on a Friday with a retrospective and review meeting. The next sprint is planned the following Monday. The team organizes a 15 minutes Scrum meeting every morning discussing project related issues. The product owner is usually attending all the Scrum meetings.

Coordinating software development work in the India I project.
Before using Scrum the team relied on standardization and direct supervision when coordinating work with their Indian team. In the beginning, the remote team was given some easy tasks specified by the Norwegian team. The Scrum master said: *“The quality was varying in the beginning, and we thought they should only concentrate on the testing. Then they said ‘No, this is not fun, please give us something more exiting to work on’, so we gave them different tasks, and this worked pretty well.”*

After using Scrum for 6 months the project had implemented all the Scrum practices, and felt they were succeeding with continuously improving their Scrum process. The team tried to work as if they were all collocated, ignoring the geographical and time differences. The Scrum master said: *“Being distributed is a big barrier. We used a lot of time on discussions between people in the two sub-teams. It didn’t work. The solution was to appoint one of the remote developers the role of a local Scrum master. Then we mostly communicated with her. It was much more efficient to delegate the responsibility to one person.”*

It was hard to achieve mutual adjustment because of the time consuming communication. To improve the communication it was decided to let the Indian Scrum master stay in Norway for a period. The Scrum master said: *“This improved the situation a lot. The productivity increased while she was here. The important issue is to communicate with only one person.”* She was participating in all the Scrum meetings as an integrated member of the team while situated in Norway. The Scrum master gave her credit and said that: *“She is very good at coming up with ideas and show initiative.”* At the same time it was also decided to let the remote team work on its own module.

Even though they started applying Scrum, and assigned a member of the remote team as a local Scrum master, the coordination between the two teams are still described as a traditional way of developing software. During the planning meetings in Norway, the local team plans and suggests initial estimates for all the tasks in the project, and then assigns tasks to their remote partner. The remote team turns the tasks into sub-tasks, and provides new estimates. In the end, the Norwegian team checks the results. The Scrum master said: *"We decide what tasks are appropriate for them. Tasks are assigned to them, and verified."* They are clearly coordinating the remote team through direct supervision.

The Norwegian Scrum master, the Scrum master from India and one of the Norwegian developers had frequent meetings (2-3 times a week) with the remote team. This was a kind of distributed Scrum meeting. In the meetings between the two sub-teams they relied on chat and e-mail. The Scrum master said: *"We tried to use telephone-conferences, but it didn't work very well, because of language problems. Written communication is easier to understand. Extensive use of chatting even makes it possible to ask a question right away. It takes time to organize a telephone conference."* He continued: *"It was also difficult to use only 15 minutes on the telephone. It usually took an hour. Chat is better."*

The Scrum master from India is involved in the planning and daily meetings standing equal to the other members, but then she coordinates the remote team by deciding who should do what. While they abandoned the use of output standardization, they have not achieved a very high level of mutual adjustment towards the remote team. Their primary coordinating mechanism is direct supervision (Figure 5).

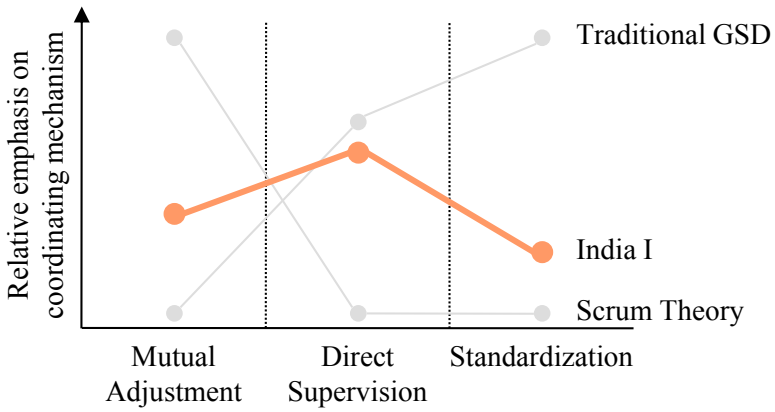


Figure 5: Relative emphasis on coordinating mechanisms between the onshore and offshore teams: More emphasis is placed on direct supervision and mutual adjustment than on standardization.

Transparency

The remote and local team are working on the same codebase and development environment. This allows for some transparency of who has done what. They have also managed to establish an automated build system, which allows everyone to see the progress or impediments if the build fails. When the Indian Scrum master is in India, scrum meetings are semi-daily, as the Scrum master said: *"Although not daily, we have regularly meetings equivalent to daily scrums, but just including me and the remote Scrum master, and possibly one more."*

When collocated, she takes part in the regular daily scrum. The Scrum master claims that this is beneficial, because: *"When she was collocated with us we had*

an increase in productivity. It was quite substantial." It seems that the transparency increased some, because the Scrum master said: *"Before she joined the team, the remote team implemented exactly as specified even when they received an obviously erroneous specification. They must have understood that it was a mistake, but the point, for them, was to have their back clear. This has improved."* While this is more a cultural issue, the Scrum master would have been able to resolve it earlier if he was continuously aware of their work.

4.2 Project India II

The project is partly outsourced to India. The goal of the project was to develop a complex software system for quality audits in organizations. This project represents the second release of the system and will provide multi user support. Two departments of the studied company are involved, each acting as an internal customer responsible for contracts with their own international customer.

The project consists of a product owner, who is also a project manager, and an architect from Norway, while development is outsourced to India. Four remote developers are working 100% on the project, one of them as a team leader. In addition a few remote developers contribute part time on the project. The Indian team members are given specialized responsibilities, like GUI.

Scrum was applied from the inception of this project because, according to the product owner, *"our customer didn't understand the creation of an old-fashioned functional specification, so we thought: Okay, let's try an agile approach."* They agreed on a contract that allowed the use of a backlog with a constantly updated list of business and technical requirements, and continuous deployment of short deliveries. The backlog was maintained by the product owner. In addition to the described Scrum practices, they used continuous integration and semi-automatic deployment, and code reviews.

Coordinating software development work in the India II project.
The project started after the first initial backlog was created by the product owner. After the initial design was created, the work was then planned and divided into sprints in cooperation with the Indian team. This failed. The product owner said: *"I quickly gave up defining the sprints together with the remote team."* She continued: *"It was very difficult because of problems with the communication. [...] We didn't understand each other, and there were cultural differences, too."*

The product owner explained how they changed their way of coordinating work, after finding it too time consuming to do the sprint planning in cooperation with the remote team: *"We started sending them work-packages specified in detail, but we realized it would much to do this for each work package."* The solution was then to create a principal work plan and then further specify and document backlog items with use-cases described in documents.

The product owner and the remote team leader communicate daily, often several times a day. She said: *"The team leader down there assigned the tasks to the team. I've been dealing only with him."*

The assignment of tasks to the Indian team became less detail oriented and instead there was an increased focus on continuous communication. It seems like the product owner tried to act more as described in the Scrum literature. She was maintaining the backlog and specifications, while letting the Indian team work

out the details: *“I don’t know everything, so I try to tell them that: ‘This is the use case, you need to solve it. Work it out.’ And it works. They ask: ‘Can we discuss this?’, and of course we do.”*

Now they use a backlog where the product owner registers items that are up to the Indian team manager to solve. The product owner said: *“It is up to him how they solve it. Now they make their own choices,”* and *“the mutual adjustment has increased, it has improved.”* Still, the remote team has not had any training in Scrum and the remote team manager coordinates his team with direct supervision.

Coordination of work with the remote team is mainly based on direct supervision and standardization in the form of written specifications and reporting of status, but the mutual adjustment is increasing (Figure 6). The team is also relying on frequent informal communication. However, the biggest challenge is to get feedback from the remote team. The product owner said: *“I miss that they detect problems and show initiative.”*

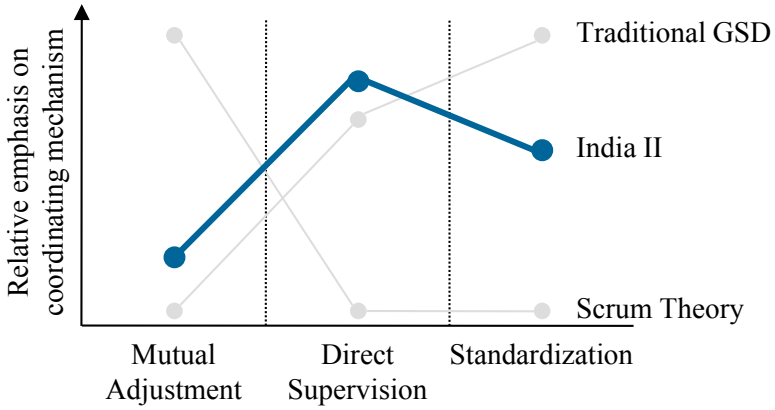


Figure 6: Relative emphasis on coordinating mechanisms between the onshore and offshore teams: More emphasis is placed on direct supervision and standardization than on mutual adjustment.

Transparency
The continuous integration and a semi-automatic deployment system provide transparency for the product owner. Still, the product owner relies heavily on frequent communication for transparency. *“We communicate every day, even several times a day,”* she said, and continued: *“Whenever something emerges, we communicate, and I inquire if there has been silence for a while. Communication is absolutely the prime critical factor to succeed.”*

4.3 Project Eastern Europe

The project is partly internally offshored to Eastern Europe. The goal of the project is to develop a system for collection and visualization of data from ship-inspections. When ships are inspected, the results are stored in the system, and the collected data are visualized through 3D models. The 3D engine was first developed as a prototype five years ago, before it was integrated into the core system and then released. Each time the product is sold to a new customer it requires adaptation and modification of the system. Several contracts with different customers from all over the world have been signed.

Four to five developers are situated in the remote team in an East European country, while two developers are situated in Norway, together with two persons from the support department, one from sales and a project manager acting as a product owner. The Norwegian team implements the daily Scrum. These meetings are also used for discussion of future solutions. They tried to implement sprints for the whole project, but failed. Tasks are mostly assigned to the Norwegian team's members by the project manager, who said, while pointing at the backlog: *"I've been putting some signatures on who is going to do what."*

Coordinating software development work in the Eastern Europe project. The project was originally applying a traditional, waterfall inspired model. This changed a year ago when a new project manager was assigned. The two distributed teams tried to use a common Scrum process. They were conducting several joint Scrum meetings each week, and implemented shared responsibilities with mutual adjustment. Originally, the remote team was only responsible for the creation of 3D models, but when it was decided to integrate them in the total development process, they faced new challenges. The project manager said: *"We thought that we should try Scrum, but because we wanted the remote team to take part in development and bug fixing, daily Scrum became a challenge. [...] We didn't manage to interact and cooperate. It became too time consuming."*

According to the project manager, the remote team was unfamiliar with the system. This unfamiliarity made communication time consuming. The project manager said: *"We felt that the Norwegian team members used too much time communicating with the remote team."* The project manager also felt that the remote team did not deliver as expected. She said: *"The software did sometimes seem inadequately tested."* This dissatisfaction was communicated to the remote team.

The project manager considered the problem to be difficulties gaining a thorough understanding of the complex source code, and commented on how tasks were divided: *"If we had managed to identify bigger chunks of new functionality to be developed by the remote team, it might have been easier for them."* To improve the situation it was decided to divide responsibility between the teams and to give the remote team tasks that required less cross-site coordination.

The Norwegian team is now responsible for the core system, bug fixing, new functionality and customer relations, while the remote team is mainly responsible for system configuration and the creation of 3D models for each customer. The project manager said: *"Because of their 3D competency, it works, because then they don't have to communicate with us all the time. [...] It's only if they lack a specification or domain knowledge, for instance when they miss an overview of what to put on the ship, then they come back and ask."* She also reported a lack of initiative.

Coordination of work between the teams is mainly based on standardization and direct supervision. The project manager said: *"They get told all the way, and they get asked all the way."* The level of mutual adjustment is low (Figure 7).

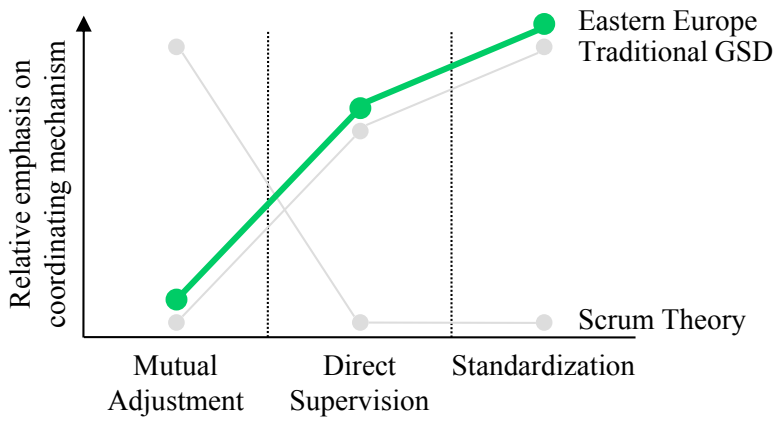


Figure 7: Relative emphasis on coordinating mechanisms between the onshore and offshore teams: More emphasis is placed on standardization and direct supervision than on mutual adjustment.

Transparency
The project manager complained about low awareness in the project: *"There are things that we just discover by incidence, things that we don't control. I feel that I don't control the resources. For instance, we don't know when they are at work."*

The local team provides some transparency through a spreadsheet with project data, mostly broken down to a ship-by-ship level, including data on who has implemented what. There was also a collocation of the teams for a two week period. The project manager said: *"We didn't manage to bring them through all the nitty-gritty stuff, so there was a lot of chat, mail and telephone afterwards."*

They also had a common platform for development, with tools and logs containing bugs and enhancements. The daily builds posed a particular problem, according to the project manager, who said: *"When the daily build failed, we had to find out who checked in the code that broke the build, and if it was someone remote, it might be that he wasn't there when we needed him."* Despite the common infrastructure, there was low awareness in crucial areas. The project manager thought it would be better if they were collocated. *"You can think out loud, and when you sit together, someone will answer, but you lose this if they sit in another country. You have to initiate a conversation to get the answer instead of just turning your head and ask."*

There was also low awareness of competencies and resources, as the project manager mentioned when she said: *"It was typically things they could have asked anybody about, not just that single person, but he sat there and felt that he had to assist them continuously."* She also said that: *"If they had been sitting next to me, we could have done it together, seen the result and adjusted,"* indicating that higher awareness and collocation could have solved some problems.

4.4 Project USA

The goal of the project is to develop the configurable core functionality for one in-house and about 150 worldwide partners or customers. The complex product has been released and is now in an endless maintenance and improvement phase. The project has been using Scrum for the last three years.

There are thirty experienced people working on the project, of which twenty-five are developing a common code base and five fill management roles. Three of the five managers are group managers, as the project is organized in three groups with different technological competencies. The five managers constitute a board that possesses the role of product owner, while the product manager is located in the USA. The group manager said: *"There is no one who has a complete overview. There is no such person."* Two of the managers have weekly calls with the product manager. Regarding her role, the group manager said that: *"She has suggestions, and we absorb them, but the decisions are made here."*

An overall roadmap is provided by the board in consultation with the product manager. This is a necessity because, as the group manager said: *"We have to comply with the planning regime of the rest of the company."* The roadmap specifies two releases a year. Sprints of four to six weeks are used to reach the targets of the upcoming release, and nothing is done out-of-sprint, they always work in sprints.

For each sprint they re-organize into new teams with members from the three groups. The composition of the teams depends on the work packages they handle and the functionality they deliver. The groups start every sprint by discussing and proposing work packages and team constellations for the board, which resolves conflicting proposals and approves. After approval, each team takes its work package and scopes its sprint, breaks the package down in tasks, estimates and designs. Daily scrums are also used. Interdependencies are handled by a Scrum of Scrums meeting where everyone attends but only the Scrum masters speak, and through drop-in API-change meetings. Changes to the API must be announced before the weekly meeting. The sprints end with reviews and demos. Retrospectives are done when starting the next sprint.

While the project's development is collocated, it has some distributed aspects to note. The product owner board answers to the product manager as well as the engineering department in the USA. There is also a group of support consultants who acquire feedback from customers, and a quality assurance team of four testers in India.

Coordinating software development work in the USA project.
On the team level they have mutual adjustment both in planning and daily work. Team members are made aware of each other's work and have the ability to coordinate. The group manager said: *"People meet in their teams of three to six persons at ten o'clock every day. But then we have a Scrum of Scrum at ten past ten, where all thirty attend."* Only the Scrum masters are speaking in the Scrum of Scrums meeting, but the attendance of everyone is good for transparency, and makes everyone aware of impediments that concern them.

It is also clear that everyone on the project has their say, which is according to mutual adjustment. This is particularly visible when planning, as the group manager said: *"Each group has a meeting where they choose work packages from the roadmap. We use work packages, which is the amount of work we can manage to do in one sprint with a certain number of people. It is important that the groups make these proposals themselves."* The teams chose their own work, and how to do their work. The group manager continued: *"While we, the managers, constitute a board that practically decides the priorities, we try to absorb input from the entire project."* While the product owner board could enforce direct supervision and control, they merely prioritize the backlog of work packages and allow the teams to take initiative.

The product owner board is also committed to the company's management in USA, and is constantly adjusting towards them. The group manager said: *"When planning the roadmap, we ask the product manager: 'Should we do this or that?' Almost every suggested solution comes from us. If we discuss priorities, she has a say, and we listen, but the decision is made by us."* Although they may receive demands from engineering on details like using a "float" instead of an "int", they engage in negotiations, explain why it might be unwise to do so, e.g. due to performance issues, and resolves the disagreements by enforcing mutual adjustment.

It is clear that the project has a unique position because of their domain knowledge, which makes them able to negotiate and initiate mutual adjustment with both their supervisors and their customers. The group manager said: *"The customers are eager to assign tasks, but we don't allow them, we control the terms. Engineering, on the other hand, may assign to some degree, but there is always dialog too."*

Their initiative seems to come from the innovative nature of the project. The group manager said: *"Because the product manager is rather far away, and because it is difficult for anyone outside the core environment to have a good enough understanding to propose feature suggestions, we are responsible for intercepting any needs, demands or possible features. We discover new solutions for our customers based on feedback from them."* Their domain knowledge makes it difficult for outsiders to initiate constructive interventions, and customers do not know that they need the new features before they have been invented and demonstrated. This allows the project the privilege of high esteem and enforces coupling of decision power with knowledge.

It is also clear that they do not consider standardization an option for remote coordination, as the group manager said: *"Some customers are very absorbed in standardization, both regarding requirement specifications and the assembly-line principle. They ask for specified solutions, but we say: 'No, we do not work like that.' When we tried to comply, it was a very sub-optimal process. We have learned that it cannot be done like that."* They can do this because of their unique position in the domain and their knowledge. He said: *"We are able to cooperate with the customers on our terms using our process."*

While most of their distributed coordination is with customers and management, they also have an Indian quality assurance team of four. The group manager describes their activities like this: *"Because we impose low formalism, we may lack specifications that the QA team needs to create tests. They visit us twice a year, and we brief them about the next version. Then they walk around and speak with the developers. The documentation they receive is mostly user manuals and technical documentation. They do also have access to the ticket system of reported bugs, and they verify those. Based on all this, they create a test specification that we review and verify. Then we have one person dedicated to automate the tests. We want to make them fully automatic."*

The quality assurance team has specialized tasks, and can be considered a case of standardization of skills and knowledge. While their product is documentation, their process relies on mutual adjustment. This would be impossible without giving them access to all the information they need.

Based on the way the project coordinates with various distributed actors, and in accordance with the responses on distributed coordinating mechanisms from the

group manager, a plot was drawn to show the relative emphasis on the three coordinating mechanisms (Figure 8). Project USA relies heavily on mutual adjustment, and much less on direct supervision and standardization.

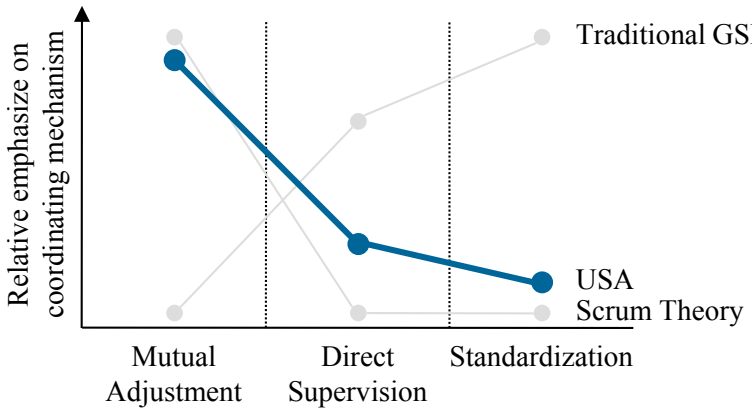


Figure 8: Relative emphasis on coordinating mechanisms between the onshore and offshore teams: More emphasis is placed on mutual adjustment than on direct supervision and standardization.

The group manager suggests that the reason for their success with agile development is their freedom: *"We are innovating. There are no needs to deliver according to order. The reason we succeed is the lack of a deliver-as-ordered model."* A consequence of this is that distributed work is easier, because of their local product owner board, which maintains the requests and needs from the real product owners in USA. Because the board has been given authority to make decisions regarding priorities, they constitute a proper product owner, and the need for distributed communication is somewhat reduced.

Transparency
The progress of the project is always visible through a tool that shows remaining hours, as well as through daily scrum. Transparency of impediments is provided by the Scrum of Scrums, as the group manager said: *"In the Scrum of Scrums meeting, I report on the status of the automatic build system, and direct focus to needed solutions. And then we have weekly drop-in API-change meetings, where proposals are announced beforehand."* Awareness of changes to interfaces is handled by API-change meetings.

They also have a wiki that gives an overview of work. The group manager said: *"We have a wiki page with work packages that we think we can finish before the release, and the groups select from this list of work packages. Each work package has a wiki page with information about the work package, the scope, status, a list of tasks, who has done what and when, and how to demonstrate completeness. The information is updated when something changes, and it is also available as an archive afterwards."*

While everything is transparent on the collocated site, only what is intentionally communicated seems to be visible to the management in USA. The quality assurance team, on the other hand, has access to every artifact and to the developers.

5 Discussion

In this section the key observations are discussed in light of the research questions:

1. How are tasks coordinated in global software development teams applying agile methods?
2. How does the level of geographical transparency affect the level of mutual adjustment?

5.1 Coordination in agile global software development

To answer the first question, it would be beneficial to evaluate how agile the projects are. One approach is to evaluate how well they conform to the generally accepted elements of the Scrum methodology. This can be done by applying the Nokia test[4]:

Are you doing iterative development?

1. Do you have fixed iterations?
2. At the end of the iteration, do you have working software?"
3. Do you start the iteration before you have a specification in complete detail?
4. Do you have testing in the middle of the development?

Are you doing Scrum?

1. Do you have a product owner that represents the customer?
2. Do you have a product backlog with estimated pieces that is prioritized by business value?
3. Do you have a burndown chart that tracks your progress and tells the velocity of the team?
4. Is the Scrum team self-organizing in the sense that the team is responsible for choosing the work, assigning themselves the work and figuring out the fastest path to deliver the work?

Project India I passes the test for the local team, but lacks a corresponding process at the remote site. The India II project manages to do iterative development, but has neither a burndown chart nor a self-organizing team. Project Eastern Europe is not doing iterative development, but implements some Scrum practices. The USA project passes the test if their product owner board is considered a proper product owner.

None of the projects succeeded in implementing a shared Scrum process for both the local and remote team (considering that the USA project lacked a remote team). This resulted in the use of different coordinating mechanisms for local and remote (strengthening proposition 1.2, Appendix E.C, p. 44). The three projects at the primary company tried to implement mutual adjustment in the distributed process, by treating the people at the remote site as equal members of the team. This was a natural move when implementing agility, because agile development

[4] http://www.infoq.com/interviews/jeff-sutherland-scrum-rules (last visited 10.06.2008). This interview with Jeff Sutherland is the most reliable freely available source on the Nokia test, a list of criteria used by Nokia to verify that their teams are using Scrum.

relies on mutual adjustment (Nerur et al., 2005). Because of problems, the three projects ended up using the traditional approach relying on direct supervision and standardization when coordinating remote work (strengthening proposition 1.3 and 1.7, Appendix E.C, p. 44). The USA project was the only one that coordinated with remote actors through mutual adjustment, but there was also some coordination by direct supervision.

The means by which the three projects of the primary company tried to achieve mutual adjustment was the daily Scrums. Daily Scrum is the most important instrument for mutual adjustment in Scrum. However, they all experienced these meetings to be time consuming, because of the flow of questions from the remote site. Language and cultural differences were also a reason for the problems with these meetings. Communication problems are often reported in global software development projects (Herbsleb and Mockus, 2003, Ramesh et al., 2006).

The daily meetings were abandoned because of the communication problems, and replaced with fewer meetings with fewer people (weakening proposition 1.5, Appendix E.C, p. 44). After a while, none of the projects coordinated through daily distributed scrums. This means that it was either too complicated for the projects level of experience, or quite as likely, none of the projects investigated resided in a proper context. For instanced, the USA project had no need for distributed daily Scrums because the developers were collocated. They did however have multiple teams performing their own daily Scrum. They did also achieve mutual adjustment in their distributed work when collaborating with customers to solve their problems.

The three projects of the primary company started to coordinate by direct supervision and detailed specifications again. This probably made it difficult to solve the communication problems, discuss the backlog, and to self-organize; one of the key tenets of agile development (Dyba and Dingsoyr, 2008). Lateral communication may be required even for low level interdependencies, so technical leads and managers ought to promote lateral communication (Cataldo et al., 2007). This is in accordance with Ramesh et al. (2006). Since this is a problem in both traditional and agile approaches to distributed software development, it is important to facilitate the needed communication because there is no guarantee that the problems are solved by introducing coordination by standardization. There is still a need for lateral communication when using the traditional approach (Cataldo et al., 2007).

Ramesh et al. (2006) suggest four practices to improve communication; synchronize work hours, provide for informal communication through formal channels, balanced coordination and constant communication. India II was only partly synchronized, but managed to communicate frequently and relied on formal channels, i.e. communication through people with dedicated roles. India I reduced the need for synchronization and coordination through modularization and communicated frequently with the remote Scrum master. The team from Eastern Europe used synchronized work hours, enabling constant communication, but the amount of communication and the lack of formalized channels negated the positive effect. Project USA had less need for continuous communication because of the lack of a distributed team, but they did provide for regular communication with management.

As mentioned, there was no joint Scrum process between the teams; however India I succeeded in implementing Scrum in Norway by dividing the project into

modules, appointing a remote Scrum master, and by moving her to Norway for periods. The other projects used a similar approach, making the remote team responsible for specific modules (strengthening proposition 1.4, Appendix E.C, p. 44). This reduced the need for everyone to communicate with everyone, and made communication less critical. The Eastern Europe project chose to assign standardized tasks to the remote team, as less complex tasks reduce the need for mutual adjustment (Mintzberg, 1989, p. 101-102). Fowler (2003) argues, in accordance with Smite et al. (2008), that this kind of modularization is important to succeed with distributed Scrum, because a remote team that is responsible for an entire module from planning to testing gets a deeper understanding of the tasks it is working on. He also suggests continuous integration to avoid surprises when integrating the modules.

Literature on traditional distributed software development suggests dividing the work into separate modules that then can be distributed to different sites to be developed (Herbsleb and Grinter, 1999). These modules should be independent in order to minimize communication between sites (Herbsleb and Grinter, 1999). The authors emphasize that it is possible to split only well-understood products for which architecture and plans are likely to be stable. However, in the current development environment with a lot of uncertainties, dividing the software into modules and specifying the modules in detail upfront is often impossible (Paasivaara and Lassenius, 2006).

Modularization also makes it possible to implement a Scrum of Scrums approach (Sutherland et al., 2007), where several teams follow their own Scrum process. The total process will then be coordinated through meetings between the Scrum masters. India I was in an early phase of implementing Scrum of Scrums.

Two of the projects improved their level of mutual adjustment after first substituting this coordinating mechanism with standardization and direct supervision (strengthening proposition 1.1, Appendix E.C, p. 44). Electronic chatting was the best remedy to support mutual adjustment, since it is instant, written text is less hampered by noise than speech, and it was perceived as timesaving compared to using a telephone conference. The increase in mutual adjustment suggests that modularization is a good approach for the transfer towards distributed Scrum.

All the projects that focused on direct supervision after failing to use Scrum felt they could reduce their level of direct supervision after some months because of an increased level of trust. Among the reasons for increased trust are frequent and reliable communication (Moe and Smite, 2007) and frequent visits by distributed partners (Ramesh et al., 2006). All the projects investigated used some level of collocation, for various reasons. Most of the projects reported that the collocation increased the level of trust. Moe and Smite (2007) shows that trust is a prerequisite for effective mutual adjustment.

It is also worth noting the major difference in how work is coordinated between the primary company and the second company. While the primary company had been using traditional approaches for distributed work for a long time and only recently started to use agile approaches, the second company had been using Scrum for three years. There is also a difference in the distributed nature of the projects, as noted several time, namely the lack of a distributed development team in the second company. The final substantial difference is in how the remote teams of the projects were perceived. Projects in the primary company reported frustration with the lacking initiative of their remote teams, while the

USA project claims that all the initiative comes from them. They show a lot of initiative to the management in the USA, who obviously trusts them a lot.

Project USA's initiative is closely connected to the complexity of their task. They are capable within their domain, and no one else has their competency or overview. They have to be innovative. This relates to Mintzberg (1989, p. 113), describing innovative organizations as depending on mutual adjustment. This is in strong contrast to the three remote teams of the primary company. All the remote teams were coordinated locally by direct supervision and the developers were highly specialized. According to Sutherland et al. (2007), it is likely that an experienced Scrum team at the remote site together with an equally skilled Scrum team at the local site would make it possible to achieve the single common distributed team approach that the three projects of the primary company set out for in the beginning.

The discussion of the four project's approaches summarizes to three approaches to agile distributed software development with Scrum:

1. Local Scrum independent of the remote team's approach
2. Multiple Scrum teams coordinated with Scrum of Scrums
3. Geographic transparency and a single distributed Scrum team

These three approaches are listed by increasing complexity and thus with increasing demands for dicipline and care. According to Mintzberg (1989, p. 102) it would also indicate that there would be more need for mutual adjustment in the last approach.

5.2 The effect of transparency

The second research question can be answered partly by theory and partly by results from the case study. Theory indicates that lack of geographical transparency disables mutual adjustment (strengthening proposition 2.1, Appendix E.C, p. 45), as noted by Smite et al. (2008) and Damian et al. (Damian et al., 2007). While it is logical that the contrary is also true, that high levels of geographical transparency enables mutual adjustment, there is no evidence for this (weakening proposition 2.2, Appendix E.C, p. 45). The reason for this is that there are other factors involved in enabling mutual adjustment, so that a disabler might hinder mutual adjustment despite the benefits of geographical transparency.

In particular the Eastern Europe project experienced a lack of transparency. This is also the project with the lowest level of mutual adjustment and highest reliance on direct supervision and standardization. It is not definite that the low mutual adjustment was because of lack of transparency, but there was clear indication that increased awareness would have enabled them to improve their mutual adjustment. This, together with theory, indicates that the level of geographical transparency is not irrelevant for the level of mutual adjustment (weakening proposition 2.4, Appendix E.C, p. 45).

If trust is low and geographical transparency is lacking, the management feel that they loose control (Moe and Smite, 2007). To regain control, they enforce monitoring, which increase their visibility of the project (Moe and Smite, 2007). This means that lack of trust does not necessarily decrease the geographical transparency (weakening proposition 2.3, Appendix E.C, p. 45). However, if the team feel that they are not trusted, they might withhold information (Moe and

Smite, 2007), which definitely would decrease the geographical transparency (strengthening proposition 2.3, Appendix E.C, p. 45). The opposite of this might be what happens in the India I project, where collocation, communication and increased visibility have slowly led to increased trust and mutual adjustment.

Since “mutual adjustment” by its very name indicates coordination between peers, one would expect little need for vertical transparency to coordinate by mutual adjustment. Horizontal transparency would be much more useful. Much time can be spent on finding the right person to get help from if a person does not have visibility into the activities of people on remote sites. The absence of available information will lead to assumptions that may conflict with requirements or assumptions made elsewhere in the project, and this introduces problems into the software (Sangwan et al., 2006, p. 5).

The importance of vertical transparency should not be overlooked. As seen from Moe and Smite (2007), the lack of vertical transparency might easily lead to increased formalization because of lacking trust. That would indirectly reduce the level of mutual adjustment. One should therefore facilitate both vertical and horizontal transparency to enable mutual adjustment (weakening proposition 2.5, Appendix E.C, p. 45).

6 Conclusion and future work

All three projects were using Scrum for the first time, and it is possible that more mature Scrum teams would communicate more efficiently because they may be more knowledgeable about and have a better understanding of issues related to applying an agile approach in a global software development project. Furthermore, none of the remote teams were trained in Scrum and this probably resulted in a lack of process understanding. Dyba and Dingsoyr (2008) report that no less than 73 % of the studies on agile projects were on projects with less than a year of experience in agile development. This is unfortunately the case for three of the four projects in this study as well.

This report presented data from a multiple case study. Three of the four projects did not succeeded in implementing mutual adjustment, and Scrum was only implemented in two local teams. In the end, the less mature projects applied a subset of Scrum practices. This study found that:

- There are at least three approaches to distributed Scrum, listed by increased complexity:
    - Local Scrum independent of the remote team's approach
    - Multiple Scrum teams coordinated with Scrum of Scrums
    - Geographic transparency and a single distributed Scrum team
- A high level of trust is important for reducing direct supervision and standardization which is important to enable mutual adjustment.
- Geographic transparency enables mutual adjustment.
- Co-locating the remote Scrum master with the local team and making the remote team responsible for dedicated modules, makes it possible to implement Scrum in part of a global software development project, and to implement Scrum of Scrums. This also reduces the need for everyone to communicate with everyone in the global software development project.
- The communication problems caused by distribution are a threat to mutual adjustment, however electronic chatting enables mutual adjustment.

In addition, there is a need for more research utilizing formal analytical methods on how work is coordinated in mature agile global software development teams, e.g. teams using Scrum of Scrums, and when there is a common Scrum process.

A References

Abrahamsson, P., Salo, O., Ronkainen, J. & Warsta, J. (2002) *Agile software development methods - Review and analysis*, VTT Publications.

Abrahamsson, P., Warsta, J., Siponen, M. T. & Ronkainen, J. (2003) New directions on agile methods - A comparative analysis. *International Conference on Software Engineering - ICSE***,** 244-254.

Agerfalk, P. J. & Fitzgerald, B. (2006) Flexible and distributed software processes: Old petunias in new bowls? *Communications of the ACM,* 49**,** 26-34.

Amrit, C. (2005) Coordination in software development: the problem of task allocation. *Proceedings of the 2005 workshop on Human and social factors of software engineering.* St. Louis, Missouri, ACM.

Beck, K. & Andres, C. (2004) *Extreme programming explained: Embrace change*, Addison-Wesley.

Berczuk, S. (2007) Back to basics: The role of agile principles in success with a distributed Scrum team. *AGILE 2007***,** 382-388.

Boden, A., Nett, B. & Wulf, V. (2007) Coordination practices in distributed software development of small enterprises. *International Conference on Global Software Engineering - ICGSE***,** 235-246.

Borchers, G. (2003) The software engineering impacts of cultural factors on multi-cultural software development teams. *International Conference on Software Engineering - ICSE***,** 540-545.

Canfora, G., Cimitile, A., Di Lucca, G. A. & Visaggio, C. A. (2006) How distribution affects the success of pair programming. *International Journal of Software Engineering and Knowledge Engineering,* 16**,** 293-313.

Carmel, E. & Agarwal, R. (2001) Tactical approaches for alleviating distance in global software development. *IEEE Software,* 18**,** 22-29.

Cataldo, M., Bass, M., Herbsleb, J. D. & Bass, L. (2007) On coordination mechanisms in global software development. *International Conference on Global Software Engineering - ICGSE***,** 71-80.

Cockburn, A. (2002) *Agile software development*, Addison-Wesley Boston.

Cockburn, A. & Highsmith, J. (2001) Agile software development: The people factor. *Computer,* 34**,** 131-133.

Cohen, D., Lindvall, M. & Costa, P. (2004) An introduction to agile methods. IN Zelkowitz, M. V. (Ed.) *Advances in Computers, Advances in Software Engineering.* Amsterdam, Elsevier.

Damian, D., Izquierdo, L., Singer, J. & Kwan, I. (2007) Awareness in the wild: Why communication breakdowns occur. *International Conference on Global Software Engineering - ICGSE***,** 81-90.

Damian, D. & Moitra, D. (2006) Global software development: How far have we come? *IEEE Software,* 23**,** 17-19.

Dourish, P. & Bellotti, V. (1992) Awareness and coordination in shared workspaces. *Proceedings of the 1992 ACM conference on Computer-supported cooperative work***,** 107-114.

Dyba, T. (2000) Improvisation in small software organizations. *IEEE Software,* 17**,** 82-87.

Dyba, T. & Dingsoyr, T. (2008) Empirical studies of agile software development: A systematic review. *Information and Software Technology*.

Erickson, J., Lyytinen, K. & Siau, K. (2005) Agile modeling, agile software development, and extreme programming: The state of research. *Journal of Database Management,* 16**,** 88 - 100.

Farmer, M. (2004) DecisionSpace infrastructure: Agile development in a large, distributed team. *Agile Development Conference.*

Fowler, M. (2003) Using an agile software process with offshore development. *www.martinfowler.com*.
Gutwin, C., Penner, R. & Schneider, K. (2004) Group awareness in distributed software development. *Proceedings of the 2004 ACM conference on Computer supported cooperative work*, 72-81.
Herbsleb, J. D. (2007) Global software engineering: The future of socio-technical coordination. *International Conference on Software Engineering - ICSE*, 188-198.
Herbsleb, J. D. & Grinter, R. E. (1999) Architectures, coordination, and distance: Conway's law and beyond. *IEEE Software,* 16, 63-70.
Herbsleb, J. D. & Mockus, A. (2003) An empirical study of speed and communication in globally distributed software development. *IEEE Transactions on Software Engineering,* 29, 481-494.
Herbsleb, J. D., Paulish, D. J. & Bass, M. (2005) Global software development at Siemens: Experience from nine projects. *International Conference on Software Engineering - ICSE*.
Hirschheim, R. & Dibbern, J. (2006) *Information technology outsourcing in the new economy - An introduction to the outsourcing and offshoring landscape*, Springer Berlin Heidelberg.
Holmstrom, H., Fitzgerald, B., Agerfalk, P. J. & Conchuir, E. O. (2006) Agile practices reduce distance in global software development. *Information Systems Management,* 23, 7-18.
Korkala, M. & Abrahamsson, P. (2007) Communication in distributed agile development: A case study. *33rd EUROMICRO Conference on Software Engineering and Advanced Applications*, 203-210.
Kraut, R. E. & Streeter, L. A. (1995) Coordination in software-development. *Communications of the ACM,* 38, 69-81.
Krishna, S., Sahay, S. & Walsham, G. (2004) Managing cross-cultural issues in global software outsourcing. *Communications of the ACM,* 47, 62-66.
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, 781-794.
Lee, A. S. (1989) A scientific methodology for MIS case studies. *MIS Quarterly,* 13, 33-50.
Levitt, T. (1983) The globalization of markets. *Harvard Business Review*, 2-11.
Malone, T. W. & Crowston, K. (1994) The interdisciplinary study of coordination. *ACM Computing Surveys (CSUR),* 26, 87-119.
Mintzberg, H. (1989) *Mintzberg on management: Inside our strange world of organizations*, Free Press. New York, USA.
Moe, N. B., Dingsoyr, T. & Dyba, T. (2008) Understanding self-organizing teams in agile software development. *Australian Software Engineering Conference - ASWEC*.
Moe, N. B. & Smite, D. (2007) Understanding lacking trust in global software teams: A multi-case study. *Lecture Notes in Computer Science,* 4589, 20-32.
Nerur, S. & Balijepally, V. G. (2007) Theoretical reflections on agile development methodologies. *Communications of the ACM,* 50, 79-83.
Nerur, S., Mahapatra, R. & Mangalaraj, G. (2005) Challenges of migrating to agile methodologies. *Communications of the ACM,* 48, 72-78.
Nisar, M. F. & Hameed, T. (2004) Agile methods handling offshore software development issues. IN Hameed, T. (Ed. *International Multitopic Conference.*
Pressman, R. S. (2005) *Software engineering: A practitioner's approach*, McGraw-Hill Professional

Prikladnicki, R., Audy, J. L. N., Damian, D. & de Oliveira, T. C. (2007) Distributed software development: Practices and challenges in different business strategies of offshoring and onshoring. *International Conference on Global Software Engineering - ICGSE*, 262-274.
Paasivaara, M. & Lassenius, C. (2004) Using iterative and incremental processes in global software development. *International Conference on Software Engineering - ICSE*, 42-47.
Paasivaara, M. & Lassenius, C. (2006) Could global software development benefit from agile methods? *International Conference on Global Software Engineering - ICGSE*, 109-113.
Ramesh, B., Cao, L., Mohan, K. & Xu, P. (2006) Can distributed software development be agile? *Communications of the ACM,* 49, 41-46.
Rising, L. & Janoff, N. S. (2000) The Scrum software development process for small teams. *IEEE Software,* 17, 26-32.
Robertson, R. (1992) *Globalization: Social theory and global culture*, Sage Publications Inc.
Salas, E., Sims, D. E. & Burke, C. S. (2005) Is there a "big five" in teamwork? *Small Group Research,* 36, 555-599.
Sangwan, R., Bass, M., Mullick, N., Paulish, D. J. & Kazmeier, J. (2006) *Global software development handbook*.
Schmidt, K. & Bannon, L. (1992) Taking CSCW seriously: Supporting articulation work. *Computer Supported Cooperative Work - CSCW,* 1, 7-40.
Schwaber, K. (1995) Scrum development process. *OOPSLA95 Workshop on Business Object Design and Implementation*.
Schwaber, K. (2004) *Agile project management with Scrum,* Redmond, WA, USA, Microsoft Press.
Schwaber, K. (2007) *The enterprise and Scrum,* Redmond, WA, USA, Microsoft Press.
Schwaber, K. & Beedle, M. (2001) *Agile software development with Scrum*, Prentice Hall PTR Upper Saddle River, NJ, USA.
Smite, D., Moe, N. B. & Torkar, R. (2008) Pitfalls in remote team coordination: Lessons learned from a case study. *PROFES.* Rome, Italy.
Sulfaro, M. (2007) Agile practices in a large organization: The experience of Poste Italiane. *Lecture Notes in Computer Science,* 4536.
Sutherland, J., Viktorov, A., Blount, J. & Puntikov, N. (2007) Distributed Scrum: Agile project management with outsourced development teams. *Hawaii International Conference on System Sciences - HICSS*, 274.
Webster, J. & Watson, R. T. (2002) Analyzing the past to prepare for the future: Writing a literature review. *MIS Quarterly,* 26, 13-23.
Yin, R. K. (2003) *Case study research: Design and methods*, Sage Publications Inc.

B Glossary

Agile software development is lightweight processes or practices for software development which is incremental, cooperative, straightforward and adaptive (Abrahamsson et al., 2002)

Awareness is an understanding of the activities of others, which provides a context for one's own activities (Dourish and Bellotti, 1992)

Crystal (Methods) is several related methodologies applicable based on heaviness

Coordination is the process of managing dependencies between activities (Malone and Crowston, 1994).

Coordinating mechanism is a defined approach to achieve management of dependencies between tasks (see direct supervision, mutual adjustment and standardization)

Direct supervision achieves coordination by having one person take responsibility for the work of others whose work interrelates, by issuing instructions and monitoring their actions, and thus enforcing control (Mintzberg, 1989, p. 101)

Distributed software development is software development spanning multiple geographical sites

Dynamic Systems Development Methodology (DSDM) is an agile approach focused on fixing time and resources allocated, while adjusting the amount of functionality accordingly

Extreme Programming is twelve software development practices taken to the extreme

Global comprise the concepts: Worldwide, universal, pertaining the whole world

Global software development is software development spanning multiple geographical sites distributed beyond national borders, but not necessarily multiple organizations

Globalization is a process of interaction and integration among the people, companies, and governments of different nations, a process driven by international trade and investment and aided by information technology. (www.globalization101.org)

International comprise pertaining multiple nations, two or more

Multinationals are multi-national companies, companies with markets in several countries

Mutual adjustment achieves coordination by the simple process of informal communication (Mintzberg, 1989, p. 101)

NTNU is the Norwegian University of Science and Technology

Offshoring is relocation of business processes from one country to another, both within the company and to external contractors

Outsourcing is relocation of internal business processes to an external company

Scrum is an agile approach to the management of software development projects (Erickson et al., 2005, Cohen et al., 2004, Abrahamsson et al., 2002)

SINTEF is the Norwegian association for industrial and technological research, the largest independent research organization in Scandinavia

Software development is the production of software, spanning business goals, requirements, planning, coding, testing, releasing and often maintenance

Standardization is either standardization of work processes, standardization of outputs, standardization of skills (as well as knowledge), standardization of norms or a multitude of these (Mintzberg, 1989, p. 101)

Transparency is the availability of the appropriate knowledge or information for coordination

C List of figures

Figure 1: The Scrum methodology (Schwaber, 1995) 8
Figure 2: Cycles in the Scrum methodology (Schwaber, 2004, p. 6) 10
Figure 3: The change of primary coordinating mechanism with increasing task complexity (Smite et al., 2008). 12
Figure 4: Relative emphasis on coordinating mechanisms: Agile development relies purely on mutual adjustment, while global software development (GSD) emphasizes standardization and some direct supervision. 14
Figure 5: Relative emphasis on coordinating mechanisms between the onshore and offshore teams: More emphasis is placed on direct supervision and mutual adjustment than on standardization. 23
Figure 6: Relative emphasis on coordinating mechanisms between the onshore and offshore teams: More emphasis is placed on direct supervision and standardization than on mutual adjustment. 25
Figure 7: Relative emphasis on coordinating mechanisms between the onshore and offshore teams: More emphasis is placed on standardization and direct supervision than on mutual adjustment. 27
Figure 8: Relative emphasis on coordinating mechanisms between the onshore and offshore teams: More emphasis is placed on mutual adjustment than on direct supervision and standardization. 30
Figure 9: Gantt diagram showing the research schedule 47

D List of tables

Table 1: Organizational and geographical distributions (Prikladnicki et al., 2007). 2
Table 2: Concepts and keywords used when searching for literature 5
Table 3: Literature databases used when searching for literature 5
Table 4: Comparison of traditional and agile development (Nerur et al., 2005) 7
Table 5: Best practices from literature on distributed software development (Cataldo et al., 2007). Conformity with mutual adjustment (MA), direct supervision (DS) and standardization (S) is based on Mintzberg (1989, p. 101).. 13

E Research Plan

Case study lacks the best reputation as a proper scientific method, but proper design and execution will counteract this. A good design is fundamental to achieve high internal and external validity and to support theories and claims based on a case study. A theory can be tested by observance of predictions that are logically derived from the theory. If the predictions are supported, the theory is strengthened, but it is refuted if the predicted results are lacking (Yin, 2003, p. 29). This matches scientific research, where falsifiability, logical consistency and the theory's ability to explain or predict indicates its quality (Lee, 1989). Needless to say, the theory must remain unfalsified.

In addition to focusing on the scientific quality of the case study, I want to be observant of changes. Has it always been like this? When did it change? Why? Or what was done to change it? Answers to these questions may give a view of change in "controlled variables", thus giving an "experimental" touch to the case study and the ability to compare "response variables". These changes will be important when evaluating the validity of the theory.

E.A Method

Through a case study inspired by (Yin, 2003) I will try to identify different kinds of coordination in distributed agile software development, and maybe how coordinating through agile methods affect effectiveness of distributed teams.

E.B Research questions

Based on the focus on coordination of tasks in agile global software development teams/projects, I state the following questions:

1. How are tasks coordinated in global software development teams applying agile methods?
2. How does the level of geographical transparency affect the level of mutual adjustment?

E.C Propositions (Hypothesis to refute by observations)

These are propositions that might or might not be true. They are based on the research questions above.

RQ1: How are tasks coordinated in global software development teams applying agile methods?

1-1. Scrum introduces more mutual adjustment.
1-2. Coordination of local and distributed tasks is the same.
1-3. Distributed tasks are coordinated through supervision.
1-4. Distributed tasks are coordinated through modularization, local tasks through daily Scrums (standardization vs. mutual adjustment).
1-5. Tasks are coordinated through daily distributed Scrums facilitating mutual adjustment.
1-6. Responsibility for coordinating work is different for planning and sprints.
1-7. Scrum is not really used for the distributed parts.

RQ2: How does the level of geographical transparency affect the level of mutual adjustment?

2-1. Lack of geographical transparency disables mutual adjustment between teams.
2-2. High level of geographical transparency enables mutual adjustment.
2-3. Lack of trust discourages geographical transparency.
2-4. Geographical transparency is irrelevant for the level of mutual adjustment.
2-5. Vertical transparency and horizontal transparency is not equally important when facilitating mutual adjustment.

E.D Unit(s) of analysis

Data are collected through two instances. The first comprise an initial interview with the project leaders of three projects. The second comprise further interviews with project leaders and team members, possibly some observations of work practices and hopefully email or phone interviews with distributed members. The initial interviews should also reveal what kind of documentation is available. For observations, it would be interesting to observe the team during daily Scrums, scrum planning, review and retrospective.

E.E Logic linking data to propositions

Findings should be able to refute or support these statements, and the following list shows what findings one should anticipate for support or refutation.

RQ1: How are tasks coordinated in global software development teams applying agile methods?

1-1. Scrum introduces more mutual adjustment.
- Claims that team members show more initiative together with less assignment of tasks and claims of self-organization indicate support.

1-2. Coordination of local and distributed tasks is the same.
- Lack of evidence that coordination is different for tasks handled by the distributed team indicates refutation.

1-3. Distributed tasks are coordinated through supervision.
- Indications that tasks are handed over or assigned to the distributed team by the local team or by local team members support, while the distributed team's participation in daily Scrums or choice of tasks indicate refutation.

1-4. Distributed tasks are coordinated through modularization, local tasks through daily Scrums (standardization vs. mutual adjustment).
- Claims of differences between local and distributed coordination and indications of modularization supports, while distributed participation in the daily Scrum indicates refutation.

1-5. Tasks are coordinated through daily distributed Scrums facilitating mutual adjustment.
- Claims of daily Scrums where each member is responsible or co-responsible for choosing tasks indicate support, while assignment of tasks by a leader or by specialization indicates refutation.

1-6. Responsibility for coordinating work is different for planning and sprints.
- Identification of different support for the three coordination mechanisms between planning phases and during sprints supports the claim, while identical coordination mechanisms indicate refutation.

Identification of different roles in coordination would also indicate support.

1-7. Scrum is not really used for the distributed parts.
   - Clear division between the distributed team and the local team together with differences in processes indicate support, while lack of differences indicates refutation.

RQ2: How does the level of geographical transparency affect the level of mutual adjustment?

2-1. Lack of geographical transparency disables mutual adjustment between teams.
   - Claims that both geographical transparency and mutual adjustment are absent indicate support, while identification of mutual adjustment without geographical transparency leads to refutation.

2-2. High level of geographical transparency enables mutual adjustment.
   - Identification of both geographical transparency and mutual adjustment indicate support, while identification of geographical transparency without mutual adjustment leads to refutation.

2-3. Lack of trust discourages geographical transparency.
   - Statements that indicate withholding of information because of distrust supports the proposition, while identification of transparency despite a lack of trust leads to refutation.

2-4. Geographical transparency is irrelevant for the level of mutual adjustment.
   - If there are no correlation between geographical transparency and mutual adjustment, the hypothesis is supported, otherwise it is refuted.

2-5. Vertical transparency and horizontal transparency is not equally important when facilitating mutual adjustment.
   - Claims that horizontal transparency plays a greater role in facilitating mutual adjustment indicate support, or if vertical transparency is emphasized, while identification of equal importance of both vertical and horizontal transparency indicate refutation.

E.F Criteria for interpreting the findings

It is important for the validity of the findings to not jump to conclusions. This means that refutation of the hypothesis through contradicting findings should be valued more than findings supporting the hypothesis, and supporting findings should be multiple and unambiguous to be considered substantially supporting.

E.G Schedule

February
- Research Plan
- Introduction
- Theory

March
- Case study
- Introduction
- Theory

April

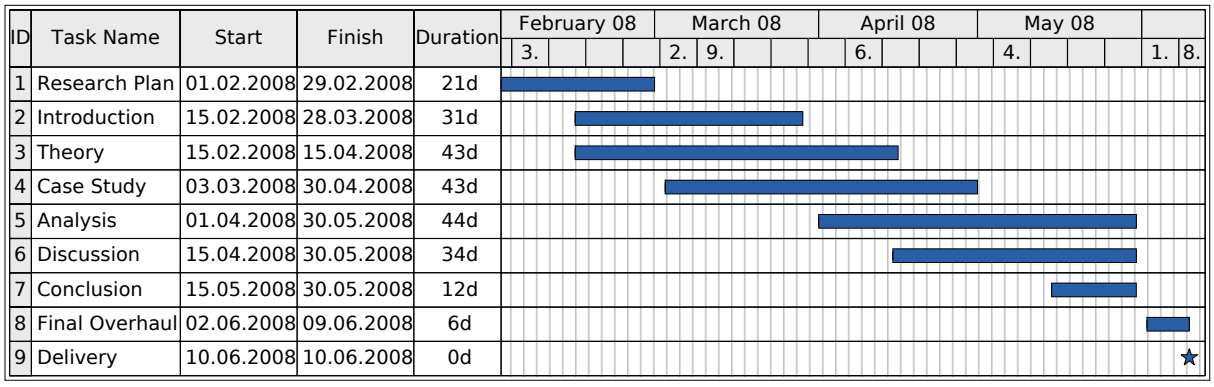
- Case study
- Theory
- Analysis
- Discussion

May

- Analysis
- Discussion
- Conclusion

June

- Final overhaul
- Latest possible delivery: June 10th



ID	Task Name	Start	Finish	Duration
1	Research Plan	01.02.2008	29.02.2008	21d
2	Introduction	15.02.2008	28.03.2008	31d
3	Theory	15.02.2008	15.04.2008	43d
4	Case Study	03.03.2008	30.04.2008	43d
5	Analysis	01.04.2008	30.05.2008	44d
6	Discussion	15.04.2008	30.05.2008	34d
7	Conclusion	15.05.2008	30.05.2008	12d
8	Final Overhaul	02.06.2008	09.06.2008	6d
9	Delivery	10.06.2008	10.06.2008	0d

Figure 9: Gantt diagram showing the research schedule

F Interview guides

The first interview session is with three project leaders of three different projects utilizing varying degree of distribution. It is important to identify the degree of distribution, the level of compliance with Scrum and the coordination mechanisms present in the projects.

The second interview session takes place more than a month later, and should involve the same project leaders as well as Scrum masters and members of the teams, and possibly product owners. It is also desirable to interview the members of the teams that are not co-located.

The interviews are unstructured, but utilize an interview guide.

The interview guides state several categories of concepts that are relevant when answering the research questions. There are several questions and statements within each category. The questions aim to be general and without expectations of a certain answer, and the statements reminds the interviewer of concepts that are relevant within the category.

F.A Questions for the first session (06.03.2008)

The first session encompasses three projects at a Norwegian company. Two of the projects outsource tasks to India, while one project offshore tasks to an in house site in Poland. The main goal of this interview is to find out:

- How is work coordinated in the projects?
- Are there different coordination mechanisms for local and remote?

Introduction

- Present myself, SINTEF, Evisoft
- May I record the interview? (Only me and Nils Brede will use the recording, to remember what has been said)
- Expected duration 30 min

Project 5 min

- About the project
- The project’s importance, duration
- How many people and what roles? (Local and remote)
- How long? Experience? Distributed?
- Using agile methods? For how long? Other method?
- Satisfaction with the project

Role 2 min

- Your role
- Who does what?
- And the remote team?
- Who is product owner? What does he/she do?
- Do members take responsibility?

Changes 3 min

- What is different with Scrum?
- How were you working before Scrum?

- Distributed?
- Change in efficiency?
- Completion?
- Why?

Mintzberg 2 min
- Present the graph
- Ask for a plot, one for local, one for remote

Collaboration 10 min
- How do you make plans?
- When do you plan?
- How are the tasks specified? Oral, backlog, documentation.
- How are tasks assigned?
- How is architecture agreed upon?
- Why and how was/is it done?
- What kinds of documentation exist? (Backlog, burndown, other)
- How much time is spent on daily Scrums and task discussions?
- How do you solve problems in the team?
- Why?

Distribution 5 min
- Why people from another country?
- What are the main differences between the local team and the remote team?
- Same kind of tasks? Any tasks that could not be done by remote?
- If a task is complex, local or remote (or both)?
- How are their approaches different?
- How are their approaches similar?
- 'Our' and 'their' way of doing things?
- All the way? (Specification, implementation, test, maintenance)?
- What if remote is stuck?
- How is feedback given? Easy to give feedback/response? What is feedback generally about?
- Why?

Communication 2 min
- How frequent is distributed communication?
- Through which channels? (documents, meetings, virtual meetings, e-mail, phone, skype, msn, other)
- Language? Everybody comfortable?
- Time zones?

Finalizing
- Is there anything else we should have covered/discussed?
- Thank you

F.B Questions for the second session (05.05.2008)

Questions for the second session are based on the analysis of the first session. Due to difficulties with gaining further access to the original organization, a search for a new organization has been initiated. This reduces the possibility of

doing a further depth study. To remedy some of this, I will search for an organization with more mature Scrum teams, since the lack of such in the original organization is a challenge when generalizing the findings.

The main goal of this interview is to find out:

- How is work coordinated in the projects?
- Are there different coordination mechanisms for local and remote?
- Who are informed about what, or shielded from information?
- Is there enough transparency to allow any team member to take initiative?

Introduction

- Present myself, SINTEF, Evisoft
- May I record the interview? (Only I will use the recording, to remember what has been said, and quotes may be used in articles.)
- Expected duration 1 hour

Project 10 min

- About the project nature, local organization, agility
- The project's importance, duration
- How many people and what roles? (Local and remote)
- How long? Experience? Distributed?
- Using agile methods? For how long? Other method?
- Satisfaction with the project

Role 2 min

- Your role
- Who does what?
- And the remote team?
- Who is product owner? What does he/she do?
- Do members take responsibility?

Changes 5 min

- How were you working before Scrum/Agile?
- What is different with Scrum/Agile?
- Distributed?
- Change in efficiency?
- Completion?
- Why?

Mintzberg 5 min

- Present the graph
- Ask for a plot, one for local, one for remote

Collaboration 10 min

- How do you make plans?
- When do you plan?
- How are the tasks specified? Oral, backlog, documentation.
- How are tasks assigned?
- How is architecture agreed upon?
- Why and how was/is it done?
- What kinds of documentation exist? (Backlog, burndown, other)
- What information is available to everyone?
- What kind of information is limited?

- How much time is spent on daily Scrums and task discussions?
- How do you solve problems in the team?
- Why?

Transparency 5 min

- Who has access to information?
- Does everyone have knowledge of everyone?
- Does everyone know who's doing what?
- Are there restrictions on information?

Distribution 15 min

- What are the main differences between the local team and the remote teams?
- Same kind of tasks? Any tasks that could not be done by remote?
- If a task is complex, local or remote (or both)?
- How are their approaches different?
- How are their approaches similar?
- 'Our' and 'their' way of doing things?
- All the way? (Specification, implementation, test, maintenance)?
- What if remote is stuck?
- How is feedback given? Easy to give feedback/response? What is feedback generally about?
- Why?

Communication 5 min

- How frequent is distributed communication?
- Through which channels? (documents, meetings, virtual meetings, e-mail, phone, skype, msn, other)
- Language? Everybody comfortable?
- Time zones?

Finalizing

- Is there anything else we should have covered/discussed?
- Thank you