POSTPRINT[1]: Coordinating Knowledge Work in Multiteam Programs: Findings From a Large-Scale Agile Development Program

Abstract

Software development projects have undergone remarkable changes with the arrival of agile development approaches. Although intended for small, self-managing teams, these approaches are used today for large development programs. A major challenge of such programs is coordinating many teams. This case study describes the coordination of knowledge work in a large-scale agile development program with 12 teams. The findings highlight coordination modes based on feedback, the use of a number of mechanisms, and how coordination practices change over time. The findings can improve the outcomes of large knowledge-based development programs by tailoring coordination practices to needs over time.

Keywords

program management, interteam coordination, agile software development, agile approaches, software engineering, method tailoring

Introduction

Software development has undergone remarkable changes since the arrival of agile development approaches in the late 1990s (Dingsøyr, Nerur, Baijepally, & Moe, 2012). Agile approaches emphasize customer involvement, technical product quality, incorporating changing and emerging requirements, and the idea that software development is best done in small, colocated, and self-managed teams (Hoda, Noble, & Marshall, 2012). These approaches have led to far-reaching changes in how software projects are planned and managed, with an increased focus on software development as teamwork (Melo, Cruzes, Kon, & Conradi, 2013; Moe, Dingsøyr, & Dybå, 2010). There has also been an emphasis on arenas for planning, synchronization, and review, and on practices to make teams work efficiently together, such as establishing shared code ownership and discussing and learning through practices such as programming in pairs.

From being used for small colocated teams, agile approaches are increasingly also being used in other settings, such as in large programs with multiple teams (Xu, 2009). Large programs generally incorporate technical and organizational complexity. This includes a large number of stakeholders, a large number of program participants, a large number of requirements, lines of software code, and often very complex interdependencies among tasks as well as teams that depend on other teams. Programs using agile approaches risk a lack of interaction and difficulties in communication (Xu, 2009) because most communication is done orally within

[1] The final version of this paper has been published in *Project Management Journal* Vol 49 / Issue 6, 2018 by SAGE Publications Ltd. © Project Management Institute, 2018. It is available as open access at: https://journals.sagepub.com/doi/10.1177/8756972818798980

the teams. Such complexity generally has a negative effect on project performance (Floricel, Michela, & Piperca, 2016). Large-scale programs pose a greater risk and are often associated with cost overruns, late completions, and project failures (Flyvbjerg, 2014; Flyvbjerg & Budzier, 2011).

The success of large programs is dependent on the program's ability to manage this complexity. Coordination is critical because the work is carried out simultaneously by many development teams (Fagan, 2004; Hoegl, Weinkauf, & Gemuenden, 2004). Therefore, it is important to study how coordination practices are used in large-scale agile development. The literature on coordination has emphasized permanent constellations such as organizations. There is less emphasis on temporal constellations such as projects and programs (Dietrich, Kujala, & Artto, 2013).

This study describes agile practices in a large, multiteam program, focusing on how the practices enable coordination of knowledge work on the interteam, project, and program levels. We describe development approaches that blend agile and traditional approaches and discuss how this combination improves coordination. We address the following research questions:

1. How are coordination practices used in large-scale agile development programs?

2. How do coordination practices change over time?

The understanding of coordination in large programs is currently limited (Dietrich et al., 2013). Software development programs have developed new ways of working that could provide relevant insight for other types of knowledge-intensive projects (Conforto, Salum, Amaral, da Silva, & de Almeida, 2014; Serrador & Pinto, 2015). In addition, agile development on a large scale challenges assumptions in existing approaches (Rolland, Fitzgerald, Dingsøyr, & Stol, 2016). Further, large programs are often critical for our society, and today most advice on conducting such programs is based on experience rather than research. This study offers rich descriptions of the use of concrete practices. These practices add to what is described in the existing advice on agile software development. Finally, how coordination changes over time is relatively unstudied in the literature (Jarzabkowski, Le, & Feldman, 2012). The size of the program will change during execution, and learning among participants and the state of the product could influence coordination needs. Understanding changes in practices will enable participants to adjust coordination practices to the needs of the program. We position this research in line with thoughts on rethinking project management, focusing on handling the complexity of projects, and aiming to develop theory for practice (Winter, Smith, Morris, & Cicmil, 2006).

Large-Scale Agile Development

Software development is a nonroutine activity because most systems developed are unique and cannot be developed again. Software development is often described as creative work where a single optimal solution may not exist, and progress toward completion can be difficult to estimate (Kraut & Streeter, 1995; Shepperd, 2014).

One reason for this is that interdependencies among different tasks may be uncertain or challenging to identify. This makes it difficult to know who should be involved in work and whether there is a correct order in which parties should complete their own specialized work (Okhuysen & Bechky, 2009). Changes in customer needs and in technology also pose challenges for software development projects and emphasize other needs for project management than what is found as engineering practices in other domains (Bryant, 2000). Agile software development allows for changing requirements throughout the development cycle and stresses collaboration with customers and early product delivery.

Agile development is an umbrella term for a range of approaches (Dingsøyr, Dybå, & Moe, 2010) that share a set of key ideas formulated in the *Agile Manifesto* (Manifesto for Agile Software Development, 2001). We define agile approaches as development approaches that "rapidly or inherently create change, proactively or reactively embrace change, and learn from change while contributing to perceived customer value (economy, quality and simplicity)" (Conboy, 2009, p. 340).

The most widely used agile approach thus far is Scrum (Rising & Janoff, 2000; Schwaber & Beedle, 2001). This approach also provides the most advice on how to manage a development project (Abrahamsson, Oza, & Siponen, 2010). However, because a development team is self-managing, the project manager role is removed. The only roles in the team are developers and a team facilitator—that is, the scrum master. The scrum master is responsible for solving problems that prevent the Scrum team (typically five to nine people) from working effectively. The scrum master works to remove impediments from the process. This role ensures decision making in daily meetings and validates decisions with management (Schwaber & Beedle, 2001). Software is developed by the self-managing team in iterations called sprints, starting with a planning meeting and ending with a review and demonstration of the product and a retrospective that focuses on process improvement. During a sprint, the team coordinates through daily meetings, often in front of a Scrum board. Features to be implemented are registered in a product backlog as *user stories* that should be understandable by the customer organization. User stories are often grouped into broader epics. The product owner provides priority on backlog items in dialogue with the team. The tasks to be performed in the next iteration are listed in the sprint backlog. Multiple stakeholders can participate in generating product backlog items such as customer, project team, marketing and sales, management, and support (Abrahamsson, Salo, Ronkainen, & Warsta, 2002).

Recently, there has been increasing attention placed on how agile approaches can be used in large development projects or programs (Hobbs & Petit, 2017). We define very large-scale agile development as "agile development efforts with more than ten teams" (Dingsøyr, Fægri, & Itkonen, 2014, p. 275), which have complex knowledge boundaries within the program as the result of many technical and business domains and the number of tasks and dependencies between tasks. Further, such programs are characterized by a complex interplay with a larger number of technologies involved and usually a large set of stakeholders (Rolland et al., 2016).

Coordination

Coordination can be understood as “managing of dependencies between activities” (Malone & Crowston, 1994, p. 90) and coordination mechanisms—"the organizational arrangements that allow individuals to realize a collective performance" (Okhuysen & Bechky, 2009, p. 472). Interdependencies include the sharing of resources, synchronization of activities, and prerequisite activities.

Basic mechanisms for coordination are discussed in management science (Mintzberg, 1989) and include direct supervision; mutual adjustment; and standardization of work, outputs, skills, and norms. Direct supervision is when one person is responsible for coordinating the work and gives directives to those who do the work. Mutual adjustment is when workers adjust themselves to one another as their work proceeds. The other mechanisms are different kinds of preplanned standardization: standardization of work, output, skills and knowledge, and norms.

Knowledge-intensive work such as developing services based on software brings a new sense of acuteness to the coordination challenge because the speed of innovation invalidates predetermined interdependencies (Ramesh, Pries-Heje, & Baskerville, 2002). In such work, team members need mutual awareness to coordinate themselves by adjusting their own work to the work of others. Research has proposed different conceptual approaches for such

adjustments—for example, transactive memory systems (Wegner, 1986), sensemaking (Weick, 1995), shared cognition (Cannon-Bowers & Salas, 2001), Complex Adaptive Systems (Vidgen & Wang, 2009), collective problem solving (Hutchins, 1991; Weick, Sutcliffe, & Obstfeld, 1999), and the collective mind (Crowston & Kammerer, 1998). These studies and studies on expertise coordination (Faraj & Sproull, 2000) offer insight into how team members can coordinate their actions in response to what other team members or people outside the team are doing. In studies of multiteam systems, intrateam coordination has been found to influence interteam coordination (Firth, Hollenbeck, Miles, Ilgen, & Barnes, 2015).

Agile approaches are designed to cope with change and uncertainty for small teams. These approaches "de-emphasize traditional coordination mechanisms such as forward planning, extensive documentation, specific coordination roles, contracts, and strict adherence to a pre-defined specified process" (Strode, Huff, Hope, & Link, 2012, p. 1222) and mainly promote informal coordination (Xu, 2009). Agile development approaches "embrace" change by moving decision authority to the team level, making the team responsible for rough long-term plans and detailed short-term plans. In their article entitled "Why Scrum Works," Pries-Heje and Pries-Heje (2011, p. 25) state that Scrum "requires very little time trying to foresee and negotiate the work flow and coordination mechanisms prior to actually conducting the work."

Pries-Heje and Pries-Heje (2011) emphasize four artifacts that they believe are especially useful for coordination: the product backlog, the sprint backlog, the Scrum board, and daily meetings. Strode et al. (2012) provide a comprehensive review of coordination studies in agile development and developed a model of coordination in agile software development projects (at the team level) that describes coordination strategies in terms of synchronization (activities and artifacts), structure (proximity of team members, availability of team member, substitutability of team members), and boundary spanning (interaction with other organizations outside of the project). A particular mechanism to facilitate synchronization is the length of iterations. Shorter iterations will increase coordination but at the cost of more frequent planning and review meetings. Two-week iterations are common in small project teams.

There is a small body of studies on how teams coordinate in very large-scale agile development, such as Xu's (2009) research. Vlietland and van Vliet (2015) propose that embedded coordination practices within and between Scrum teams positively impact delivery predictability in large projects. A study of Scrum of Scrums (a meeting to coordinate development teams) suggests that this forum did not lead to satisfactory coordination: Feature-specific or site-specific forums were better, but coordination at the project level was still a challenge (Paasivaara, Lassenius, & Heikkila, 2012). Paasivaara and Lassenius (2014) describe a very large-scale development initiative at Ericsson with 40 teams where four types of communities of practice are used to coordinate teams. A survey on coordination in large-scale software teams found that respondents hoped for more effective and efficient communication as well as an emphasis on the importance of good personal relationships (Begel, Nagappan, Poile, & Layman, 2009).

The influence of coordination configurations on coordination effectiveness has also been the focus of researchers working closely with SAP (Bick, Spohrer, Hoda, Scheere, & Heinzl, in press; Scheerer, Hildenbrand, & Kude, 2014; Scheerer & Kude, 2014) while inspired by work in organizational psychology on multiteam systems (Mathieu, Marks, & Zaccaro, 2001; Zaccaro, Marks, & DeChurch, 2012). Findings in this field include that interteam (cross-team) processes are more important than intrateam (within-team) processes for the performance of multiteam systems (Marks, DeChurch, Mathiu, Panzer, & Alonso, 2005). Multiteam systems are defined as "two or more teams that interface directly and interdependently in response to environmental contingencies toward the accomplishment of collective goals" (Mathieu et al.,

2001, p. 289). A project developing a software and hardware solution is described as coordinating through extensive use of face-to-face contact (Marks & Luvison, 2012).

Coordination Modes

Work is often given to teams in large projects and programs. Several factors then define the need for coordination between the teams. Van de Ven, Delbecq, and Koenig (1976) discussed three main determinants of coordination mechanisms for organizations:

• *Task uncertainty*—the difficulty and variability of work undertaken by an organizational unit. Higher degrees of complexity, thinking time to solve problems, or time required before an outcome is known all indicate higher task uncertainty.

• *Task interdependence*—the extent to which people in an organizational unit depend on others to perform their work. A high degree of task-related collaboration means high interdependence.

• *Size of work unit*—the number of people in a work unit. Increases in participants in a project or program mean an increase in the size of the work unit.

A number of mechanisms can be applied to achieve coordination, and coordination is usually exercised through several mechanisms (Dietrich et al., 2013). Van de Ven et al. (1976) proposed three coordinating modes that were used by Dietrich et al. (2013) in their study of multiteam projects. The first two are based on feedback (or mutual adjustment (Mintzberg, 1989)), and the last is based on codification:

Table 1. Coordination Modes, Definition, and Main Coordination Mechanisms (Dietrich, Kujala, & Artto, 2013)

Coordination Mode	Definition (Dietrich et al., 2013)	Coordination Mechanism (van de Ven et al., 1976)
Group mode of personal coordination	Use of mechanisms in which mutual adjustments occur in a group of occupants (more than two) through meetings	Scheduled meetings Unscheduled meetings
Individual mode of personal coordination	Use of mechanisms in which individual role occupants make mutual task adjustments through vertical or horizontal communication	Horizontal channels Vertical channels
Impersonal mode of coordination	Use of a codified blueprint of action that is impersonally specified	Blueprints of action

Group mode is the mechanism for mutual adjustment (Mintzberg, 1989). It is vested in a group of role occupants through scheduled or unscheduled meetings. Scheduled meetings are usually used for planned communication; unscheduled meetings are used for unplanned communication among more than two participants. In agile development, group mode coordination at the team level is ensured through sprint planning meetings, daily Scrum meetings, sprint demonstration meetings, and retrospectives (Strode et al., 2012; Xu, 2009).

Individual mode is where individual role occupants make mutual task adjustments through either vertical or horizontal channels of communication. In horizontal channels, the "linkage function is assumed by an individual unit member who communicates directly with other role actors on a one-to-one basis in a non-hierarchical relationship" (Van de Ven et al., 1976, p. 323). The mechanisms for vertical communication are usually line managers and unit supervisors. In large programs, this includes program management, project and subproject managers, and team leaders. In agile development, practices in extreme programming (Beck & Andres, 2004), such as pair programming, colocation, shared code ownership (Strode et al., 2012), and onsite customers (Xu, 2009), support horizontal coordination.

Impersonal mode involves coordination mechanisms that are programmed or codified. Once implemented, they require minimal verbal communication between people. Examples are pre-established plans, process documentation, intranet pages, information technology tools, and roadmaps. A "codified blueprint of action is impersonally specified" (Van de Ven et al., 1976, p. 323). This is present in agile approaches such as in coding standards (Xu, 2009), but we can also see agile approaches themselves as a type of impersonal mode. The Scrum approach codifies types of meetings and roles and sets expectations for stakeholders.

As determinants change, prior studies have indicated corresponding changes in coordination mode. Van de Ven et al. (1976) found that increases in task uncertainty lead to a substitution of impersonal coordination with horizontal coordination mechanisms and group meetings. Increased task uncertainty causes a need for extensive and dynamic knowledge exchange to solve problems and adjust for emerging changes (Van de Ven et al., 1976). Dietrich et al. (2013) also pointed to prior studies, which found that technological novelty relates to a higher rate of group meetings instituted by management. Project managers can achieve more control of work in such uncertain situations by relying on group-driven interaction in scheduled meetings.

Increased interdependence among people in units in general leads to an increased use of personal modes of coordination (Dietrich et al., 2013), especially in the individual mode (Van de Ven et al., 1976). Increased unit size, however, is associated with greater use of impersonal coordination and hierarchy (but no decrease in group mode coordination) (Dietrich et al., 2013).

In their study of multiteam projects, Dietrich et al. (2013) describe an information systems project in addition to five cases from other domains. The project had three concurrent teams as well as a project manager, steering group, a quality control group, a coordination group, and a one-person project office. All teams had a dedicated team leader. There is no information about the development process in the case description. This project was characterized by a high degree of use of personal coordination modes—especially with a high use of the individual coordination modes. The study also reports the use of some mechanisms in the impersonal mode.

In addition, large projects and programs are temporal constructions requiring a great need to learn because everyone is new to the program. In these programs, developers typically need to learn about the business domain. There are also constant developments in technology and work approaches that require learning. Changes in coordination practices are a significant influence on information sharing, work flow fluency between teams, and the efficiency of projects (Dietrich et al., 2013). Thus, it is interesting to investigate changes over time.

Change Over Time: From Coordination to Coordinating

Early research on coordination mechanisms adopted a static view on coordination. This focus was later criticized (Crowston, 1997; Harrison, Mohammed, McGrath, Florey, & Vanderstoep, 2003; Jarzabkowski et al., 2012; Okhuysen & Bechky, 2009). To capture the dynamic nature of coordination, Marks, Mathieu, and Zaccaro (2001) outlined a framework of

team processes focusing on team performance as a connected input-process-outcome occurring over time. As the interdependencies among different pieces of work become more uncertain, complex, or challenging to identify, more of the coordination becomes situated. Thus, common awareness is essential for effective coordination (Okhuysen & Becky, 2009). Effective temporal planning by groups creates temporal awareness norms that are necessary to adapt, change, and avoid problems to perform at high levels and reduce coordination difficulties (Janicik & Bartel, 2003). The nature of development teams will change over time: Teams that work over time have higher levels of speed and quality than one-shot teams (Harrison et al., 2003).

Temporal organizations such as projects and programs are likely to change coordination practices over time. The degree of coordination will depend on the nature of the project. Not surprisingly, studies of open source development projects indicate that emergent projects have more coordination than stable, mature projects (Chua & Yeow, 2010). An aspect that differentiates projects from permanent organizations is that projects have deadlines and milestones, and are thus affected by time pressure. Time pressure has been shown to weaken projects' ability to synchronize pace, ensure timely coordination, and utilize knowledge (van Berkel, Ferguson, & Groenewegen, 2016). Further, the three factors influencing coordination mode—task uncertainty, task interdependence, and size of work unit—are likely to change during the execution of a project. Task uncertainty is likely to change as project members develop understanding of the domain and the technical choices made. A study of evolution of coordination in outsourced software projects found that uncertainty changed because of the involvement of new individuals (Sabherwal, 2003). The degree of task interdependence can change as a result of technical choices made during a project. A large project will typically start with a small team, extend to a peak with several teams, and have a tail with fewer people involved.

The changing needs in different phases of a project are illustrated by a study on multiteam projects that examined differences in coordination in the concept and development phases. The study found that managing team interfaces is particularly important during the concept stage (Hoegl & Weinkauf, 2005). Further, a focus on structuring and supporting the project is most important during the development phase, though this activity can hinder team performance in the concept stage.

Coordination mechanisms adjust to adapt to uncertainty, novelty, and change over time (Jarzabkowski et al., 2012). This study explains coordination mechanisms as "dynamic social practices that are under continuous construction" and further describes how coordination mechanisms change over time. In uncertain situations with major changes, hierarchies and rule-based systems have been found to be less useful than informal and interpersonal communications.

Method and Case

This study builds on a revelatory case study that investigates how agile approaches can be adapted on a very large scale (Dingsøyr, Moe, Fægri, & Seim, 2017), which also includes a focus on coordination practices.

The case was chosen because practitioners described it as a successful, very large program that extensively used agile development approaches. The entire program was colocated and coordination mechanisms could be studied in a setting that is well suited for agile approaches. The Perform program developed a new office automation system for the Norwegian Public Service Pension Fund. The program was managed by the department and involved consulting companies Accenture and Sopra Steria as subcontractors in the project development. The

program ran from 2008 to 2012. At its highest point, 12 teams were working in parallel on development (175 total people).

Data Collection and Analysis

Data were collected through two sources: First, 12 retrospective group interviews were conducted with the public department and the two main consulting companies on interteam coordination and knowledge sharing as well as architecture and customer involvement. There were 24 program participants and each interview lasted two hours, producing 247 pages of transcribed material in total. The participants had roles that included project management, subproject management, technical architecture, functional architecture, testing, scrum masters, and development. All participants were seniors with at least four years of experience in software development. The interview guide for interteam coordination is provided in the Appendix at the end of this article. Second, we got access to three documents: an official report after program completion, an internal experience report, and the quality assurance report from the program. These reports contain 277 pages of text.

We analyzed the material with a software package for qualitative analysis (QSR International, 2017) and used the framework established in the background section to identify expressions relating to coordination modes and changes in coordination mechanisms over time. For example, the statement "On an overall level, the teams worked quite similarly after exchanging experience. So the Scrum boards were used quite similarly across the teams. There were differences in colors, but the function was quite similar" was coded as *Scrum boards*, which was later grouped with other concepts as *unscheduled meetings* under *group mode* in the framework created by Van de Ven et al. (1976). See Dingsøyr et al. (2017) for further details on data collection and analysis.

Case: The Perform Program

Perform is one of the largest IT programs in Norway, with a final budget of about EUR 140 million. The program started in January 2008 and lasted until March 2012. Of the 175 people involved, 100 were external consultants from five companies. The program used both time and material and target price contracts for subcontractors. About 800,000 person-hours were used to develop around 300 epics with a total of about 2,500 user stories. These epics were divided into 12 releases.

The program was managed by a program director who mainly focused on external relations; a program manager focusing on the operations; as well as a controller and four project managers responsible for the architecture, test, business, and development projects, respectively:

• *Architecture*—responsible for defining the overall architecture in the program and for detailing user stories in the solution description phase.

• *Test*—responsible for testing procedures and approving deliverables from the development teams.

• *Business*—responsible for analysis of needs through defining and prioritizing epics and user stories in a product backlog.

• *Development* —divided into three subprojects: one led by the Norwegian Public Service Pension Fund (six teams) with their own people and people from five consulting companies, and two other subprojects led by external consulting companies Accenture and Sopra Steria (three teams).

There were also projects for communication and adoption to prepare users for the new systems, totaling six projects.

The program used a matrix structure where the business and development projects took part in the architecture and test projects. This matrix structure meant that a feature team would mainly participate in project development while also devoting resources to project architecture (through a technical architect), business (through a functional architect), and test (through a test responsible).

Initially, the development process included four phases per release:

• *Analysis of needs*—walkthrough of the target functionality of a release and identification of high-level user stories.

• *Solution description*—user stories were assigned to epics and were described in more detail, including design and architectural choices.

• *Construction*—development and delivery of functionally tested solutions from the product backlog. Five to seven three-week iterations per release. The teams used Scrum with sprint planning, daily meetings, sprint demonstration, and sprint retrospectives.

• *Approval*—a formal functional and nonfunctional test to verify that the entire release worked according to expectations.

To ensure development work on high-priority user stories, there was pressure to have solution descriptions ready for the feature teams. This meant that releases were constantly being planned, constructed, and tested. Thus, given the roles in a team (developers, technical architect, functional architect, test responsible, and scrum master), a feature team would constantly be engaged in construction for the current release while approving delivered functionality in the previous release and analyzing the needs for the next release. After approval of the program, new releases were acceptance tested, set in production, and underwent an approval phase before being accepted by the operational IT section of the department.

Results

This section provides an overview of the main coordination modes used in the program, the group mode of personal coordination, the individual mode of personal coordination, and the impersonal mode of coordination. Coordination mechanisms found in these modes are shown in Table 2. In addition, we describe how the coordination mechanisms changed over time.

Table 2. Coordination Mechanisms After Coordination Mode.

Coordination Mode	Coordination Mechanism	Description
Group mode of personal coordination	Architecture project meeting	Meeting with lead architect and technical architects from teams
	Board discussions	Around a physical board showing team tasks
	Business project meeting	Meeting with product owners and functional architects from teams

	Demo	Meeting where developers show solution to customer and stakeholders
	Experience forum	Meeting in one subproject to share experience between teams on technical topics
	Lunch seminars	Meeting in one subproject to share experience between teams on self-selected topics
	Metascrum	Formal meeting on project level on project progress
	Open space technology	Informal meetings on topics raised by participants across all teams
	Open work area	Teams located around tables on one floor
	Retrospectives	Meeting at the end of an iteration to discuss changes in development process
	Scrum of Scrums	Meeting between representatives from teams in subprojects on progress
	Technical corner	Meeting on one subproject to share experience between teams on technical topics
	Test project meeting	Meeting with lead tester and test responsible from teams
Individual mode of personal coordination	Rotation of team members	Rotation of members between teams within subprojects
	Customer on-site	Customer representatives available in open office space for consultation
	Direct communication in open work area	

		Easy access to people in other teams in the open work area
Impersonal mode of coordination	Instant messaging	Tool that facilitated asynchronous communication among all project participants
	Masterplan	Plan for the main functions to be included in the solution (epics)
	Architectural guidelines	Description of main technical design decisions and standards for development
	Team routines	Descriptions in a wiki on expectations for teams
	Cross-team routines	Descriptions in a wiki on expectations for work across teams
	Solution descriptions in wiki	Description of what was to be implemented, including detailed user stories

Group Mode of Personal Coordination

The program was characterized by a number of scheduled meetings as well as arenas for unscheduled meetings for coordination in groups. We first describe scheduled meetings at the program and project levels.

At the program level, the only arena where everyone would meet was at demonstration meetings. These were held every three weeks. In addition, the program management met two times a week in the Metascrum. This meeting included managers from the main projects and the central program management, giving attention to high-level obstacles to progress and assessment of risks in the program. Well into the program, a new arena was introduced—the *open space technology*. Open space was a way to get the entire program to establish a number of meetings across the project organization to discuss challenges and improvement initiatives. The decision to use a chatting tool was a result of these meetings. In addition, there were separate meetings to identify dependencies in tasks before work was assigned to teams.

At the project level, there were three main types of scheduled meetings: meetings prescribed by the agile approach Scrum, meetings in the main projects in the program, and fora at the project level to share experience across the development teams.

The Scrum of Scrums was held in the three development subprojects with Scrum masters and subproject managers from three to six development teams. Project managers sometimes participated in these meetings. One subproject had daily Scrum of Scrum meetings in the beginning, but reduced the frequency to three times per week. A topic discussed here were

resources: "Now we have two people who are ill in the team, and we have given away a person to the environment team, how shall we manage to deliver our stories in the iteration?" (subproject manager). In addition, retrospectives were sometimes held across teams in the subprojects, but overall this was an activity within each team.

The main project's architecture, business, and test had meetings with their own staff and people with related roles in the development teams. In the business project, much of the work concentrated on managing dependencies: "There were dependencies throughout the program" (technical architect). One of the participants in meetings in the business project said, "When we talked to the product owner, the product owner said, 'We need you to do this,' but then we had to explain that to achieve that, we first need to do these tasks" (functional architect). The meetings in the project architecture focused on establishing architectural guidelines, but also focused on coordinating work among the development teams to reduce the number of teams working on the same part of the codebase: "This was to reduce the possibility of making trouble for each other—which we did." The codebase was organized to reduce these challenges. In the meetings, the teams declared, "This is our central area of work during this period, so please limit work in that area" (technical architect).

Experience sharing across teams was the focus of several scheduled meetings at subproject level, for example: experience forum, lunch seminars, and technical corner. One topic discussed at the experience forum was how to liven up the retrospectives. This was then a topic discussed among all participants in the development teams in one project. Participation in these meetings was voluntary.

Unscheduled meetings were easy to organize because of the open workspace. Unplanned meetings frequently occurred around the boards that were available for each team. These were used to "discuss solutions as well as draw and make sketches" (subproject manager). These discussions spanned development teams and roles. The project management was placed on tables so that people in management roles could see most of the boards and thus quickly get an overview of the status of the teams. If the project managers noticed discussions, then they could inquire about the issue and say, "I know this problem was addressed by another team two iterations ago; let us get Ola over here and see if he can help" (subproject manager). A Scrum master and developer stated that they learned "very much" in the program during these discussions around the boards, but it was important to have sufficient coordination arenas so that people realize that "we need to talk." The program also started to use a group chatting tool (Jabber) to ease informal coordination. This is a type of unscheduled virtual meeting. This tool was introduced during the program, and it enabled team members to ask several people for help without interrupting them. This channel was used for a number of purposes—from asking technical questions to informing people about the next wine lottery.

Informants emphasized the importance of the unscheduled meetings. One said, "I think the combination of scheduled and unscheduled coordination that just appeared was very important" (Scrum master and developer).

Individual Mode of Personal Coordination

The program was characterized by direct informal coordination among members of different teams using both horizontal and vertical channels.

The development program used a number of different arenas to coordinate work and share knowledge between teams. During the build-up phase, new team members were often enrolled; this was very important. This was facilitated by the occasional splitting of existing teams and even the distribution of new team members. Changes in team members helped alleviate problems in personal chemistry. The frequency of both types of changes to teams was considerably lower in later phases. Team changes were an important and consciously used mechanism for distributing knowledge and facilitating coordination, both horizontal and

vertical. Over time, resistance to the team changes markedly increased with strengthening team feeling: "There was a limited number of people who were candidates for such change due to competence, so I had to do some pep talks and get people to think positively" (subproject manager). In order to enable self-managed teams, the development program sought to limit the authority of Scrum masters compared to normal practice. A key motivation was to inspire the team members to take responsibility and coordinate internally and between teams. Scrum masters were, however, key to effective changes in team composition. They helped gather information by talking to all the team members to share the status of the work.

In terms of horizontal coordination, many people have emphasized the importance of informal coordination facilitated by the open work area. Team members asked for advice across the team and organizations: "We are here to succeed and no one can succeed alone" (subproject manager). Team members experienced personal coordination as crucial to solving interdependencies between tasks and keeping the schedule. This can be described as a direct contact between experts. As the program progressed, pragmatism in the allocation of tasks between suppliers became extensive. Eventually, one could just ask, "Can you help me with this?" and receive an "Okay. We'll help you with this now if you help us with something else in the future" (subproject manager). The management also sought to rotate the team members in such a way that some of the team members from the development team would also participate in the solution description. In addition, extensive personal coordination was possible because all contractors were working toward the same goals. Social arenas such as lunches, coffee breaks, and other happenings were described as important coordination mechanisms during the project.

One of the mechanisms for vertical coordination was management by walking around. It was used to get status from the team, help the teams, and spread important information such as solutions to common problems. A culture developed where decisions were discussed informally among relevant stakeholders; these decisions were subsequently formalized.

Impersonal Mode of Coordination

The main impersonal coordination mechanisms were the program plan, guidelines, and checklists.

The program plan included all work to be done and was described as epics. All epics and tasks were initially documented in an electronic spreadsheet. However, this spreadsheet was replaced because of two problems related to coordination: First, it was difficult to get a good overview of the entire plan by using spreadsheet technology because of the size of the program. Second, it was difficult to locate the latest version of the spreadsheet because multiple versions were created and distributed using various channels.

About a year into the program, an issue tracker (Jira) replaced the spreadsheet. This new tool for coordination was introduced together with a major revision of the plan. The new plan included 300 epics and 22 work packages. The 300 epics were later decomposed to 2,500 user stories with subtasks. Every team could follow the program progress in the tool. Although the issue tracker was mandatory, used by all teams, and regularly updated, all teams duplicated their tasks on stickers on a board close to the table where they were located. Each team had its own board with an overview of tasks that the team had committed to solve during the next iteration. A task was written on a sticker and moved when the status of the task changed.

Though the issue tracker was essential for coordination of tasks on the program and project level, the physical board remained important for coordination on the team level. In addition, management could easily see the status of the work going on in a team just by looking at the board. As a subproject manager said, “It takes two seconds to get an overview of status [in a team], and from my location [in the open work area], I could see almost all the boards, and then I would know what had happened at the end of yesterday [in each team].” Another

explained: “This was an important ceremony to move one sticker one the board. Changing the status in the issue tracker does not bring an applause” (subproject manager).

The issue tracker was used together with a tool for facilitating code reviews to coordinate work. When the review work was registered in the tool, there was a minimal need for verbal communication among the users.

All process description documents, guidelines, and checklists were available in a wiki (a website that provides collaborative modification of content and structure directly from the web browser). The wiki was available for everyone and mandatory to use. Examples of routines were team routines and routines describing cross-team collaboration such as the daily meeting and Scrum of Scrum meetings. Examples of guidelines for designers and programmers in the wiki were guidelines for graphical user interface design, how to use the programming language Java, and how to perform specific programming tasks. The guidelines included tips and experiences written by other people in the program. The content was regularly updated. An outcome from a sprint retrospective could initiate a change in a guideline.

Although most guidelines and checklists were defined before they were used, many were created on request. One example was a team that saw a need for new architectural guidelines during an iteration. This led the architects to come together to establish a new guideline so that the next team could use it. As one architect said: “It is better to define guidelines when someone needs them instead of us trying to identify all needed guidelines up front.”

The use of guidelines and plans was evaluated in the post-project review. Some were defined too late and caused problems with, for example, error handling. Not everyone followed the guidelines because they perceived that it made them inflexible. Another explanation for lack of use was related to the number of guidelines, rules, and processes. The size of the program made it hard to get a full overview—especially for newcomers.

Table 3. Characteristics of Coordination Mechanisms over Time.

Coordination Mode	Early in Program	Late in Program
Group mode of personal coordination	Many scheduled meetings	Many unscheduled meetings
Individual mode of personal coordination		More horizontal coordination
Impersonal mode of coordination	Plan in spreadsheet	Plan in issue tracker

Coordination over Time

At the team level, the main mechanisms for coordination remained constant during the program. However, on an interteam level, there a number of changes happened over time (see Table 3).

For group mode coordination, several meetings and fora were established early in the program, as shown in Table 2. An architect said, "There were arenas that emerged, and arenas

that disappeared;" "we saw there were information needs we had not covered." These fora and meetings built knowledge regarding who knows what. When people started to get an overview of whom to talk to, informants stated that they did not need the meetings anymore. One said: “We stopped doing some meetings because we could replace them with shorter meetings or because we got to know each other, then we could just talk to each other” (technical architect). So, instead of coordinating in scheduled meetings, people started approaching others directly, discussed issues by the coffee machine or by the boards, or arranged unscheduled meetings. One informant described the later part of the program as having "daily continuous communication" (subproject manager).

For impersonal mode coordination, the main transition was the change in tools. The product backlog—the project plan—was moved from a spreadsheet to an issue tracker. The spreadsheet separated functional and technical tasks and also did not contain the whole plan but referred to other documents, such as presentation slides. The plan was now integrated and described as epics and user stories. A functional architect said, "[The issue tracker] was a very useful tool to break down tasks." From 2009, all written internal status reporting was removed; this was shown in the issue tracker.

Informants report that this change led to improvements in using the plan to coordinate, as there were no longer several versions circulating and project management and team members now had easy access to information they could trust. The new plan also contained more details. A test responsible stated, "There was a functionally responsible listed in one of the fields in the [issue tracker] and then you knew very well who to invite to a demonstration." In addition to containing the plan, the issue tracker was used for external progress reporting, keeping track of bugs in the product, and listing risk mitigation efforts by the program management.

Discussion

We structure the discussion of our findings after our two research questions. First, we ask how coordination practices are used in large-scale agile development programs.

The results show that all three modes of coordination were used in the Perform program. The program was characterized by high uncertainty regarding the tasks, a high degree of task interdependencies, and a large unit size. Prior studies suggest that this situation would call for more coordination. Indeed, we identified a number of coordination mechanisms in use across all three modes of coordination. Our study did not measure the extent of use—we could only state that certain mechanisms were used in the program.

An increase in task uncertainty was found to lead to a substitution of impersonal coordination with horizontal coordination mechanisms and group meetings (Van de Ven et al., 1976). Intrateam horizontal coordination has been identified as a characteristic in agile projects (Xu, 2009), and a study on multiteam systems describes extensive face-to-face coordination (Marks & Luvison, 2012). In Perform, we found a high presence of horizontal coordination across teams, as well as a number of scheduled meetings. High interdependence among persons leads to an increase in personal modes of coordination. We identified many mechanisms that were widely used within these modes.

In addition, unit size is associated with greater use of the impersonal mode and hierarchy. We found many impersonal mechanisms in use, but most informants focused on the group mode when describing coordination practices. However, the organization of the program, with separate projects for architecture, business, and test, emphasized the establishment of guidelines and rules across the program. Plans were made visible at both the team and program level by showing the status of tasks on boards for the teams and in the issue tracker for aggregation and overall status.

In contrast to Dietrich et al. (2013), most of the mechanisms identified in our study relate to the group mode; Dietrich et al. found that most mechanisms relate to the individual mode of personal coordination. This could be because of the focus on practices for coordination in our data collection—we did not use a targeted data collection scheme for all three coordination modes.

Comparing our findings to prior work on coordination in agile development, we see that all four artifacts emphasized for coordination by Pries-Heje and Pries-Heje (2011) were used in Perform: the product backlog, the sprint backlog, the Scrum board, and daily meetings. Following the model of Strode et al. (2012), we see that synchronization was ensured via a number of practices, such as setting the iteration length to three weeks and following the Scrum approach on the team level. The open work area and full-time engagement of program members contributed to the structure. The matrix organization provided program-internal boundary spanners. If we compare the work in our program to work in a single agile team, we find a number of additional traditional practices focusing both on forward planning through the business and architecture projects as well as on documentation that represented the test project and criteria for accepting a developed user story. We also found a number of additional roles on different levels, such as the functional and technical architects at the team level as well as project managers and other administrative roles at the project and subproject level.

Second, we ask how coordination practices change over time.

Changes over time are particularly interesting in temporal organizations such as programs, which will experience changes in task uncertainty, task interdependencies, and the size of work units over time, and will also be under the influence of time pressure (van Berkel et al., 2016), which can limit coordination and knowledge exchange.

An interesting finding in our material is the gradual transition to unscheduled meetings in the group mode. Informants saw the scheduled meetings as a prerequisite for this transition. It is likely that many meetings scheduled early in the program established relations and knowledge of other people's skills. This echoes prior findings (Hoegl & Weinkauf, 2005) that managing teams' interfaces is particularly important in the initial stage of a multiteam program.

Furthermore, the matrix organization of the program, with team members taking part in all four major projects, involved a number of scheduled meetings with subsequent development of relationships and knowledge. The combination of arenas prescribed in agile development such as the Scrum of Scrums, demonstrations, and retrospectives gave room for bottom-up coordination. The scheduled meetings in the project architecture, business, and test areas gave management control.

In line with Jarzabkowski et al. (2012), our findings suggest that coordination mechanisms are not static, but dynamic structures that change over time. Though our explorative material does not allow us to show detailed traces of changes over time, our material shows a number of scheduled meetings that existed for a while and then disappeared, such as the experience forum and the technical corner. Informants state that informal communication in the open work area increased over time as people got to know one another. In addition, new mechanisms, such as open space technology and instant messaging, appeared. There were changes in rules and plans, from making use of traditional spreadsheets and documents in the initial phase of the program to establishing a new Masterplan in an issue tracker with details and rules of work procedures described in a wiki. We speculate that there were two main drivers of changes over time: First, the domain of the program was unknown to most external consultants working on development. This required much learning about the domain itself and about whom in the customer organization could answer questions. Second, the program was

split into two teams as it scaled. This was done to meet the strict deadline of the program and led to a renewed focus on learning later in the program.

There were changes over time regarding the use of agile approaches. The frequency of Scrum of Scrum meetings changed during the program. In one subproject, this changed from daily to three times a week. Also, the retrospectives were mainly conducted at the team level, but sometimes were also held at the subproject level. Informants stated that because most decisions were discussed informally toward the end of the program, these decisions were recorded in daily meetings, the Scrum of Scrums, or in the Metascrum.

This revelatory case study has several limitations: First, we have not been able to follow the program over time, but collected data after the program was finished. Second, as an exploratory study in a new area, the data collection was broad. We asked about coordination practices but did not ask explicitly about the impersonal mode of coordination. Thus, our material on coordination mechanisms might not provide a complete overview of the mechanisms used in the program. We have also mainly focused on interteam coordination; this is influenced by coordination at the intrateam level (Firth et al., 2015).

Conclusion

We described how coordination mechanisms are used in a large-scale agile development program and how these mechanisms change over time. Our case program was characterized by high task uncertainty, a high degree of interdependence for tasks, and a large number of people.

Our research developed three main insights that we think are relevant to the project management community when adopting practices from agile development.

First, there was an increase in task uncertainty that led to a substitution of impersonal coordination with horizontal coordination mechanisms and group meetings. We established a high presence of personal communication, both in the group mode and in the individual mode. Informants emphasized the importance of the open work landscape for horizontal personal coordination. This made vertical personal coordination easier as project managers could be quickly informed of the teams' status when having one-to-one discussions. Also, establishing a mixture of agile and traditional scheduled meetings was important for building knowledge and relations early in the program. There were many scheduled meetings at first, but over time there was a gradual transition to unscheduled meetings. Meetings related to the agile approach Scrum were kept throughout the program, and the iteration length remained three weeks. The frequency of scheduled meetings is very important when balancing the risk of developing unwanted functionality and costs of ceremony in the form of time spent on planning and review. Our study supports the finding that personal coordination is central to achieving interteam coordination in large programs.

Second, there were many coordination mechanisms in use spanning all three modes of coordination. Table 2 lists the 22 mechanisms identified in the program. In contrast, traditional large-scale agile development only explicitly focuses on the Scrum of Scrums as a mechanism for interteam coordination. One mechanism was also duplicated: The plan existed on program level in an issue tracker, while each team kept a version of the plan on its board. This duplication helped serve the needs for the plans at different levels. This suggests that one coordination mechanism is not sufficient; efficient coordination can depend on a variety of mechanisms.

Third, there were frequent changes in how coordination took place. Scheduled meetings were extensively used in the introductory phase of the program, but were later replaced by unscheduled meetings. New mechanisms were used as needs changed during the program execution. Coordination practices change over time.

Future work should develop further understanding regarding coordination modes and mechanisms in large development programs. One particularly interesting topic would be to investigate how coordination mechanisms are tailored to the specific context of a program as well as to further understand how coordination needs change over time (Jarzabkowski et al., 2012). Also, the relationship between intrateam and interteam coordination (Firth et al., 2015) should be further explored to provide research-based advice on coordination for multiteam programs.

This study highlights the number of mechanisms in use in a successful program and offers rich descriptions of such mechanisms. This provides a number of suggestions in addition to what is described in the agile development literature. Second, we emphasize the role that an open working space had in this case. It was an efficient enabler of coordination. Finally, we would like to underline the change in coordination needs over time, which emphasizes the importance of practices to reflect on and change the development approach as a program progresses, such as the common practice in agile development of conducting retrospectives.

Acknowledgment

We are very grateful to all participants in the group interviews, and in particular contact persons Mette Gjertsen at the Norwegian Public Service Pension Fund, Tor-Erik Mathiesen at Accenture, and Kjetil Røe at Sopra Steria. Further, we would like to thank our colleague Tore Dybå for taking part in data collection and analysis of interteam coordination, Tor-Erlend Fægri and Svein Hallsteinsen for data collection on architecture, and all three for discussions of the case. Thanks to master's student Anniken Østdahl at the Norwegian University of Science and Technology for conducting an independent analysis on parts of the material used in this article. We also thank members of the *research factory* at the Department of Computer Science at the Norwegian University of Science and Technology for feedback on an early version of this article. In addition, we are very grateful to the anonymous reviewers and the special section editor who especially provided important pointers to related literature. This work was supported by strategic internal projects at SINTEF on large-scale agile development and the project Agile 2.0 supported by the Research Council of Norway through grant 236759 and by the companies Equinor, Kantega, Kongsberg Defence & Aerospace, Sopra Steria, and Sticos.

References

Abrahamsson, P., Oza, N., & Siponen, M. T. (2010). Agile software development methods: A comparative review. In T. Dingsøyr, T. Dybå, & N. B. Moe (Eds.), *Agile software development: Current research and future directions* (pp. 31–59). Berlin Heidelberg, Germany: Springer Verlag.

Abrahamsson, P., Salo, O., Ronkainen, J., & Warsta, J. (2002). *Agile software development methods—Review and analysis* (VTT Electronics ed., Vol. 478). VTT Publications, Oulu.

Beck, K., & Andres, C. (2004). *Extreme programming explained: Embrace change* (2nd ed.). Addison-Wesley, Boston.

Begel, A., Nagappan, N., Poile, C., & Layman, L. (2009). Coordination in large-scale software teams. Paper presented at the *Proceedings of the 2009 ICSE Workshop on Cooperative and Human Aspects on Software Engineering*, Vancouver, Canada.

Bick, S., Spohrer, K., Hoda, R., Scheerer, A., & Heinzl, A. (In press). Coordination challenges in large-scale software development: A case study of planning misalignment in

hybrid settings. *IEEE Transactions on Software Engineering*. doi: 10.1109/TSE.2017.2730870

Bryant, A. (2000). It's engineering Jim . . . but not as we know it: Software engineering—solution to the software crisis, or part of the problem? *Proceedings of the 22nd International Conference on Software Engineering* (pp. 78–87). ACM, New York.

Cannon-Bowers, J. A., & Salas, E. (2001). Reflections on shared cognition. *Journal of Organizational Behavior, 22*(2), 195–202. doi: 10.1002/job.82

Chua, C. E. H., & Yeow, A. Y. K. (2010). Artifacts, actors, and interactions in the cross-project coordination practices of open-source communities. *Journal of the Association for Information Systems, 11*(12), 838.

Conboy, K. (2009). Agility from first principles: Reconstructing the concept of agility in information systems development. *Information Systems Research, 20*(3), 329–354.

Conforto, E. C., Salum, F., Amaral, D. C., da Silva, S. L., & de Almeida, L. F. M. (2014). Can agile project management be adopted by industries other than software development? *Project Management Journal, 45*(3), 21–34.

Crowston, K. (1997). A coordination theory approach to organizational process design. *Organization Science, 8*(2), 157–175.

Crowston, K., & Kammerer, E. E. (1998). Coordination and collective mind in software requirements development. *IBM Systems Journal, 37*(2), 227–245.

Dietrich, P., Kujala, J., & Artto, K. (2013). Inter-team coordination patterns and outcomes in multi-team projects. *Project Management Journal, 44*(6), 6–19.

Dingsøyr, T., Fægri, T., & Itkonen, J. (2014). What is large in large-scale? A taxonomy of scale for agile software development. In A. Jedlitschka, P. Kuvaja, M. Kuhrmann, T. Männistö, J. Münch, & M. Raatikainen (Eds.), *Product-focused software process improvement* (Vol. 8892, pp. 273–276). Springer International Publishing, Switzerland.

Dingsøyr, T., Dybå, T., & Moe, N. B. (2010). Agile Software Development: An Introduction and Overview. In T. Dingsøyr, T. Dybå, & N. B. Moe (Eds.), Agile Software Development: Current Research and Future Directions (pp. 1-13). Berlin Heidelberg: Springer Verlag.

Dingsøyr, T., Moe, N. B., Fægri, T. E., and Seim, E. A., "Exploring Software Development at the Very Large-Scale: A Revelatory Case Study and Research Agenda for Agile Method Adaptation," *Empirical Software Engineering,* vol. 23, pp. 490-520, 2018.

Dingsøyr, T., Nerur, S., Balijepally, V., & Moe, N. B. (2012). A decade of agile methodologies: Towards explaining agile software development. *Journal of Systems and Software, 85*(6), 1213–1221. doi: 10.1016/j.jss.2012.02.033

Fagan, M. H. (2004). The influence of creative style and climate on software development team creativity: An exploratory study. *Journal of Computer Information Systems, 44*(3), 73–80.

Faraj, S., & Sproull, L. (2000). Coordinating expertise in software development teams. *Management Science, 46*(12), 1554–1568.

Firth, B. M., Hollenbeck, J. R., Miles, J. E., Ilgen, D. R., & Barnes, C. M. (2015). Same page, different books: Extending representational gaps theory to enhance performance in multiteam systems. *Academy of Management Journal, 58*(3), 813–835.

Floricel, S., Michela, J. L., & Piperca, S. (2016). Complexity, uncertainty-reduction strategies, and project performance. *International Journal of Project Management, 34*(7), 1360–1383. doi: http://dx.doi.org/10.1016/j.ijproman.2015.11.007

Flyvbjerg, B. (2014). What you should know about megaprojects and why: An overview. *Project Management Journal, 45*(2), 6–19. doi: 10.1002/pmj.21409

Flyvbjerg, B., & Budzier, A. (2011). Why your IT project may be riskier than you think. *Harvard Business Review, 89*(9), 23–25.

Harrison, D. A., Mohammed, S., McGrath, J. E., Florey, A. T., & Vanderstoep, S. W. (2003). Time matters in team performance: Effects of member familiarity, entrainment, and task discontinuity on speed and quality. *Personnel Psychology, 56*(3), 633–669.

Hobbs, B., & Petit, Y. (2017). Agile methods on large projects in large organizations. *Project Management Journal, 48*(3), 3–19.

Hoda, R., Noble, J., & Marshall, S. (2012). Developing a grounded theory to explain the practices of self-organizing agile teams. *Empirical Software Engineering, 17*(6), 609–639.

Hoegl, M., & Weinkauf, K. (2005). Managing task interdependencies in multi-team projects: A longitudinal study. *Journal of Management Studies, 42*(6), 1287–1308.

Hoegl, M., Weinkauf, K., & Gemuenden, H. G. (2004). Interteam coordination, project commitment, and teamwork in multiteam R&D projects: A longitudinal study. *Organization Science, 15*(1), 38–55.

Hutchins, E. (1991). Organizing work by adaption. *Organization Science, 2*(1), 14–39.

Janicik, G. A., & Bartel, C. A. (2003). Talking about time: Effects of temporal planning and time awareness norms on group coordination and performance. *Group Dynamics: Theory, Research, and Practice, 7*(2), 122.

Jarzabkowski, P. A., Le, J. K., & Feldman, M. S. (2012). Toward a theory of coordinating: Creating coordinating mechanisms in practice. *Organization Science, 23*(4), 907-927. doi: 10.1287/orsc.1110.0693

Kraut, R. E., & Streeter, L. A. (1995). Coordination in software development. *Communications of the ACM, 38*(3), 69–81.

Malone, T. W., & Crowston, K. (1994). The interdisciplinary study of coordination. *ACM Computing Surveys (CSUR), 26*(1), 87–119.

Manifesto for Agile Software Development. (2001). Retrieved from http://agilemanifesto.org/

Marks, M. A., DeChurch, L. A., Mathieu, J. E., Panzer, F. J., & Alonso, A. (2005). Teamwork in multiteam systems. *Journal of Applied Psychology, 90*(5), 964.

Marks, M. A., & Luvison, D. (2012). Product launch and strategic alliance MTSs. In S. J. Zaccaro, M. A. Marks, & L. A. DeChurch (Eds.), Multiteam systems (pp. 33-52). New York: Routledge.

Mathieu, J. E., Marks, M. A., & Zaccaro, S. J. (2001). Multi-team systems. In N. Anderson, D. S. Ones, H. iK. Sinangil, & C. Viswesvaran (Eds.), *International handbook of work and organizational psychology* (Vol. 2, pp. 289–313). London, England; Thousand Oaks, CA, & New Delhi, India: Sage Publications.

Melo, C. D. O., Cruzes, D. S., Kon, F., & Conradi, R. (2013). Interpretative case studies on agile team productivity and management. *Information and Software Technology, 55*(2), 412–427.

Mintzberg, H. (1989). *Mintzberg on management: Inside our strange world of organizations*. New York, NY: Simon & Schuster.

Moe, N. B., Dingsøyr, T., & Dybå, T. (2010). A teamwork model for understanding an agile team: A case study of a Scrum project. *Information and Software Technology, 52*, 480–491. doi: 10.1016/j.infsof.2009.11.004

Okhuysen, G. A., & Bechky, B. A. (2009). Coordination in organizations: An integrative perspective. *The Academy of Management Annals, 3*(1), 463–502.

Paasivaara, M., & Lassenius, C. (2014). Communities of practice in a large distributed agile software development organization—Case Ericsson. *Information and Software Technology, 56*(12), 1556–1577. doi: 10.1016/j.infsof.2014.06.008

Paasivaara, M., Lassenius, C., & Heikkila, V. T. (2012). Inter-team coordination in large-scale globally distributed Scrum: Do Scrum-of-Scrums really work? *Proceedings of the ACM-IEEE International Symposium on Empirical Software Engineering and Measurement* (pp. 235-238). New York, NY: IEEE.

Pries-Heje, L., & Pries-Heje, J. (2011). *Why Scrum works: A case study from an agile distributed project in Denmark and India Agile Conference (AGILE)* (pp. 20-28). IEEE, Los Alamitos, CA.

QSR International. (2017). NVivo for Mac (Version 11.4.2).

Ramesh, B., Pries-Heje, J., & Baskerville, R. (2002). Internet software engineering: A different class of processes. *Annals of Software Engineering, 14*(1–4), 169–195.

Rising, L., & Janoff, N. S. (2000). The Scrum software development process for small teams. *IEEE Software, 17*(4), 26-+.

Rolland, K. H., Fitzgerald, B., Dingsøyr, T., & Stol, K.-J. (2016). *Problematizing agile in the large: Alternative assumptions for large-scale agile development*. Paper presented at the International Conference on Information Systems, Dublin, Ireland.

Sabherwal, R. (2003). The evolution of coordination in outsourced software development projects: A comparison of client and vendor perspectives. *Information and Organization, 13*(3), 153–202.

Scheerer, A., Hildenbrand, T., & Kude, T. (2014). Coordination in large-scale agile software development: A multiteam systems perspective. In R. H. Sprague (Ed.), *2014 47th Hawaii International Conference on System Sciences* (pp. 4780–4788). New York, NY: IEEE.

Scheerer, A., & Kude, T. (2014). *Exploring coordination in large-scale agile software development: A multiteam systems perspective*. Paper presented at the Thirty Fifth International Conference on Information Systems, Auckland, New Zealand.

Schwaber, K., & Beedle, M. (2001). *Agile software development with Scrum*. Upper Saddle River, NJ: Prentice Hall.

Serrador, P., & Pinto, J. K. (2015). Does agile work?—A quantitative analysis of agile project success. International Journal of Project Management, 33(5), 1040–1051.

Shepperd, M. (2014). Cost prediction and software project management. In G. Ruhe & C. Wohlin (Eds.), Software project management in a changing world (pp. 51–71). Springer.

Strode, D. E., Huff, S. L., Hope, B. G., & Link, S. (2012). Coordination in co-located agile software development projects. *Journal of Systems and Software, 85*(6), 1222–1238.

van Berkel, F., Ferguson, J. E., & Groenewegen, P. (2016). Speedy delivery versus long-term objectives: How time pressure affects coordination between temporary projects and permanent organizations. *Long Range Planning, 49*(6), 661–673. doi: 10.1016/j.lrp.2016.04.001

Van de Ven, A. H., Delbecq, A. L., & Koenig Jr., R. (1976). Determinants of coordination modes within organizations. *American Sociological Review*, 322–338.

Vidgen, R., & Wang, X. (2009). Coevolving systems and the organization of agile software development. *Information Systems Research, 20*(3), 355–376. doi: 10.1287/isre.1090.0237

Vlietland, J., & van Vliet, H. (2015). Towards a governance framework for chains of Scrum teams. *Information and Software Technology, 57*, 52–65. doi: 10.1016/j.infsof.2014.08.008

Wegner, D. M. (1986). Transactive memory: A contemporary analysis of the group mind. In B. Mullen & G.R. Goethals (Eds.), *Theories of group behaviour* (pp. 185–208). New York, NY: Springer-Verlag.

Weick, K. E. (1995). *Sensemaking in organizations*. Thousand Oaks, CA: Sage Publications.

Weick, K. E., Sutcliffe, K. M., & Obstfeld, D. (1999). Organizing for high reliability: Processes of collective mindfulness. In R. S. Sutton & B. M. Staw (Eds*.), Research in Organizational Behavior* (Vol. 21, pp. 81–123). Stamford, CT: Jai Press, Inc.

Winter, M., Smith, C., Morris, P., & Cicmil, S. (2006). Directions for future research in project management: The main findings of a UK government-funded research network. *International Journal of Project Management, 24*(8), 638–649.

Xu, P. (2009). Coordination in large agile projects. *The Review of Business Information Systems, 13*(4), 29.

Zaccaro, S. J., Marks, M. A., & DeChurch, L. (2012). *Multiteam systems: An organization form for dynamic and complex environments*. New York: Routledge.

Appendix 1: Interview Guide

- How was the work organized in your part of the program?
- What kinds of dependencies were there between the teams in your part of the project? (Examples?)
- How were dependencies managed? (Examples?)
- What was managed in established fora and what was managed outside of the fora? (Examples?)
- Who was involved in managing dependencies between teams? (Examples?)
- Did you encounter challenges with managing dependencies? (Examples?)
- Did you change the way you managed dependencies during the project? (Examples?)
- What practices do you think were most important in order to manage dependencies between teams? (Examples?)
- Are there any practices you think had little importance for managing dependencies?
- How did the division of the project into three main parts influence the coordination among teams?
- Were there differences in interteam coordination across the subprojects?
- What was the frequency of meetings/ How many persons were involved? / How long did the meetings last?