International Journal of
Information Systems and Project Management
ISSN (print):2182-7796, ISSN (online):2182-7788, ISSN (cd-rom):2182-780X
*Available online at www.sciencesphere.org/ijispm*

# To schedule or not to schedule? An investigation of meetings as an inter-team coordination mechanism in large-scale agile software development

**Nils Brede Moe**
SINTEF Digital
7465 Trondheim, Norway
www.shortbio.org/nils.b.moe@sintef.no

**Torgeir Dingsøyr**
SINTEF Digital
7465 Trondheim, Norway
www.shortbio.org/torgeird@sintef.no

**Knut Rolland**
SINTEF Digital
7465 Trondheim, Norway
www.shortbio.org/knut.rolland@sitnef.no

**Abstract:**
Coordination of teams is critical when managing large programmes that involve multiple teams. In large-scale software development, work is carried out simultaneously by many developers and development teams. Results are delivered frequently and iteratively, which requires coordination on different levels, e.g., the programme, project, and team levels. Prior studies of knowledge work indicate that such work relies heavily on coordination through "personal" modes such as mutual adjustment between individuals or through scheduled or unscheduled meetings. In agile software development processes, principles and work structures emerge during the project and are not predetermined. We studied how coordination through scheduled and unscheduled meetings changes over time in two large software development programmes relying on agile methods. Our findings include transitions from scheduled to unscheduled meetings and from unscheduled to scheduled meetings. The transitions have been initiated both bottom-up and top-down in the programme organizations. The main implication is that programme management needs to be sensitive to the vital importance of coordination and the coordination needs as they change over time. Further, when starting a program, we recommend to early identify the important scheduled meetings, as having enough scheduled meetings is important to develop a common understanding of domain knowledge.

**Keywords:**
coordination, programme management, agile software development, large-scale agile, software engineering.

**DOI:** 10.12821/ijispm060303

**Manuscript received:** 8 April 2018
**Manuscript accepted:** 14 June 2018

## 1. Introduction

Coordination of work teams are of critical importance when managing large projects that involve multiple teams. Multi-team projects are used in many domains, often to "achieve high quality innovations in a satisfactory time-to-market" [1], and in such programmes "hundreds of people may be required to develop components of a new product simultaneously" (ibid). In large innovative projects, the degree of complexity and uncertainty is high, as the work executed in teams is influenced by the work and inputs from other teams. As a consequence, choosing the right coordination practices is important, as they have significant influence on information sharing, work flow fluency between teams, efficiency of the project, and learning outcomes [2]. In an editorial in the Journal of the Association for Information Systems arguing for research on programme management, Jiang et al. [3] raise the question of how interdependencies among projects can be leveraged to improve coordination.

Much of the resources used on innovations today are used on software development. Coordination was early identified as a particular challenge in software development projects. In the 1990s, software projects were often associated with overruns on time and cost, and many referred to a "software crisis". As Kraut and Streeter [4] state, "While there is no single cause of the software crisis, a major contribution is the problem of coordinating activities while developing large software systems". Since then, new methods for software development have been suggested, what is referred to as agile software development [5, 6]. The practices in this field have also inspired the project management discipline [7]. These methods were, however, intended for small, self-managing and co-located teams. Nevertheless, the popularity of these methods has spurred their use also in large development programmes [8].

Coordination in large-scale software development is of paramount importance, since the work is carried out simultaneously by many developers and development teams. Delivering results frequently and iteratively requires work and knowledge coordination on different levels, e.g., the programme, project, and team levels. Additional supporting roles are critical in large-scale projects for managing the exponential growth of interdependencies and mitigating associated risks [9]. In such projects, interdependencies are more uncertain than in small projects; therefore, teams need to know who the experts are and which experts to coordinate work with, particularly when they are outside the team or even at a different site. Dingsøyr et al. [10] describe 14 mechanisms for inter-team coordination in a large-scale software project. Further, agile methods are emergent [11], which means that processes, principles, and work structures emerge during the project rather than being predetermined. As a consequence, how an agile project is coordinated changes during a project. Therefore, to understand coordination in large-scale agile projects there is a need to study coordination over time.

Van de Ven et al. [12] propose three coordinating modes: by programming or codification (impersonal mode), and coordination by feedback on the individual (personal mode) or on a group level (group mode). In the case of high uncertainty in multi-team projects, the work relies heavily on coordination through group mode [2]. This article examines the use of the scheduled and unscheduled meetings (group mode) in large-scale agile development programmes. We analyse how coordination through scheduled and unscheduled meetings change over time in two large software development programmes that make use of agile development methods. We ask the following research question:

*How do inter-team group mode coordination mechanisms change over time in large-scale agile development?*

In answering this research question, this paper presents results from two case studies on large-scale agile software development programmes. In both cases, albeit, in different ways, coordination mechanisms break down, are tested, and become established over time. In this sense, based on our study, coordination mechanisms are not stable but dynamic and are always in the making through the lifetime of a project. Especially in terms of coordination mechanisms for the group mode, this is important in large-scale agile software development programmes. In more practical terms, for project managers, we argue that it is not only important to be aware of how to organize a project in a start-up phase but also to continuously evaluate and change coordination mechanisms over time as the project is progressing. As such, new coordination mechanisms may emerge out of the practise of project participants in a bottom-up fashion, or they may be established top-down by project managers. Our cases show that different strategies are followed at different stages or phases in the project depending on the problem situation at hand.

The remainder of the paper is organized as follows. Section 2 outlines related work. In Section 3, we describe our research methodology. In Section 4, we present our findings from the cases and cross-case analysis, which are further discussed in Section 5. Finally, Section 6 concludes the paper with a summary of major findings. This article is a revision and extension of Dingsøyr et al. [13].

## 2. Coordination in large-scale agile software development

### *2.1 Coordination in agile project management*

Software development projects are often complex undertakings that involve multiple interdependencies between resources, tasks, teams, roles and various software components and systems [14]. IT projects as a particular category of projects often imply blurred boundaries with other projects that require specific coordination [15]. Therefore, it becomes essential for project managers to pay attention to and implement means for efficient coordination. Important in relation to coordination, is the difference between traditional project management approaches and more agile approaches. Whereas traditional approaches typically focus on formal coordinating mechanisms through a pre-defined process, precise and in-depth documentation, and high levels of specialization in role assignments [16], agile approaches tend to favour self-management (teams determine the best way to handle work), emergent processes (processes, principles, and work structures emerge during the project rather than being predetermined), and more informal coordinating mechanisms [11]. Coordination in agile projects will therefore change over time. Although software development today is primarily conducted using agile methods [6], also agile approaches to project management involve coordination challenges – especially in larger projects and programmes [17]. As shown in recent literature on agile project management, agile projects involve specific challenges related to coordinating and communicating with multiple stakeholders as agile development often require frequent releases and collaboration with customers [18]. Also, in large-scale agile projects the more informal approach to coordination can become challenging [23]. Moreover, recent studies suggest that organizing projects in larger programmes may help solving some of the coordination problems across different projects [15].

### *2.2 Coordination and coordination modes*

Software development teams must coordinate the efforts of those who are part of the process, as well as ensure coordination with suppliers, clients, and other groups both outside and inside the organization. The team has to make sure that the work is complete and fits together, there is no duplication and components of the work are handed off expeditiously [4].

A widely used definition of coordination is Malone and Crowston´s: “Coordination is managing dependencies between activities” [19], published in computer science. This definition emphasize dependencies, which are the constraints on action in a situation.  Van de Ven et al. [12], from the field of sociology, define coordination as "integrating or linking together different parts of an organization to accomplish a collective set of tasks”. Because Van de Ven et al. focuses on the coordination of different parts of an organization (e.g., linking teams), their model is highly suitable in this case study, where the focus is coordination in multi-team programmes.

Coordination in large-scale projects is exercised through several mechanisms [2]. Van de Ven et al. [12] propose three coordinating modes: by programming or codification (impersonal mode), and coordination by feedback on the individual (personal mode) or on a group level (group mode). Once implemented, the impersonal coordination mechanisms are codified and require minimal verbal communication between people. Examples include pre-established plans, process documentation, intranet pages and roadmaps. Coordination by mutual adjustment or feedback is based on informal communication. In the personal mode, individual role occupants serve as the mechanism for making mutual task adjustments through either vertical or horizontal channels of communication. The mechanisms for vertical communication are usually line managers. In the group mode, the mechanism for mutual adjustment is vested in a group of role occupants through scheduled or unscheduled meetings. In projects with high task uncertainty (like in a software development project) there is a need for an extensive and dynamic knowledge exchange between and within teams to solve problems and adjust for emerging changes [20]. The scheduled meetings are therefore effective because physical proximity allows richer communication, which enables swifter and more flexible coordination [21]. Dietrich [2] also points to prior studies that found that technological novelty relates to a higher rate of group meetings instituted by management. As a consequence, planned and unplanned meetings (group mode) are important in large complex projects. Scheduled meetings are typically used for routine meetings, involving planned communication, while unscheduled meetings are used for unplanned communication between more than two participants.

### *2.3 Group mode in large agile projects*

Relying on group mode for coordination is challenging when scaling a project. In large software projects, group mode can take part within teams, between group of managers or groups of team representatives acting on behalf of their teams. Through a project hierarchy it is possible to achieve a kind of layered mutual adjustment, but only with strong elements of hierarchy and bureaucratic control [21]. A key challenge with layered mutual adjustment is that it is not always clear who should be involved in which coordination activities.

In agile software development, group mode coordination by scheduled meetings at the team level is ensured through practices like iteration planning meetings, daily meetings, iteration demonstration meetings and retrospectives [22-24]. Scrum, a project-management-oriented agile development method, was inspired by a range of fields, such as complexity theory, system dynamics, and Nonaka and Takeuchi’s theory of knowledge creation [25]. In Scrum, a self-managing team develops software in increments (sprints); each sprint starts with a planning meeting and ends with a retrospective and a review meeting. The team coordinates on a daily basis through a 15-minute daily Scrum (a daily reporting and discussion meeting) [26, 27]. Features to be implemented are registered in a product backlog, and a Product owner decides which backlog items should be developed in the following sprint. The product backlog comprises a prioritized and constantly updated list of business and technical requirements for the system being built or enhanced. Multiple stakeholders, such as clients, project teams, architects, designer, marketing and sales, management, and support, can participate in the planning phase (usually through meetings) to identify the product backlog items. During the planning meeting (usually every second, third or fourth week), the Product owner is responsible for presenting a prioritized product backlog to the team. The highest priority items from the product backlog are then detailed in a sprint backlog during a team-planning meeting. Because the team and the Product owner is responsible for defining and improving coordination practices, agile can be understood as a bottom-up approach to coordination.

Group mode coordination by unscheduled meetings is ensured at the team and inter-team level by team members and teams sitting together in the same office. Strode et al. [22] found both unscheduled cross-team talks and backlog specification meetings emerged as a result of co-location. Similarly, Nyrud and Stray [28] observed that informal and ad hoc conversations emerged in a large-scale web-program as a result of teams being co-located in an open office. While open office is an enabler for unscheduled meetings and many scheduled meetings and forums increase the amount and frequency of communication between teams, Smite et al. [29] found that it was difficult to have unplanned meetings because of too many scheduled meeting and a lack of meeting rooms.

In a large-scale setting the most common strategy for coordination across several teams is Scrum of Scrum. Scrum of Scrum is a scheduled meeting were one team-member acts as "ambassador" to participate in a daily meeting with ambassadors from other teams. However, Scrum of Scrum has been found to be inefficient in larger projects [30, 31]. As a consequence, agile consultants have created several frameworks for scaling agile, such as the Large-Scale Scrum (LeSS) [32] and Scaled Agile Framework (SAFe) [33]. The LeSS framework offers less structure and gives suggestions, tools and tips of practices that can be used for coordination, such as communities of practice and scheduled multi-team meetings. In the LeSS, any team or team member should be able and expected to reach out to another team if there is an issue to be solved (both through scheduled and unscheduled meetings). The LeSS can be understood as a bottom-up coordination approach of coordination. The SAFe is a more structured way of organizing the work, this includes, e.g., a common release calendar with joint programme increment planning days. Thus, the SAFe seems to create a structure with more organizational control, which might leave less flexibility for meetings to emerge and for teams to take the initiative for coordination. The SAFe can be understood as a more top-down approach to coordination.

### *2.4 Coordination over time*

In large-scale projects coordination mechanisms seem to change over time as involved actors need to solve new problems implicating previously unknown interdependencies [31]. In conceptualizing such dynamic processes of coordinating, Jarzabkowski et al. [34] suggests that new coordination mechanisms are gradually established through existing social practices of coordinating. Hence, all elements of a coordinating mechanism do not exist prior to coordinating – but is rather bootstrapped out of coordinating itself. So, for example, in a large-scale agile project using meetings to coordinate activities in different developments teams (Scrum of Scrums), new forms of coordinating may emerge out of participants' practices when they discover absences in the current coordinating mechanism. Arguably, this is what often happens in large-scale agile projects that start off with simple coordinating mechanisms in Scrum (daily meetings, demos, sprint planning, Scrum of Scrum, etc.) only to discover that multi-team projects (e.g., [30, 31]) often require additional mechanisms for coordinating, for example architecture meetings across teams, upfront meetings involving both customer and software provider, and communities-of-practice. Moreover, the complexity of large-scale agile projects typically involves unintended changes, twists and turns that may "disrupt" existing coordination mechanisms making them obsolete. More concretely, based on their qualitative study of coordination mechanisms, Jarzabkowski et al. [34] develop a process model consisting of five cycles that describe how coordinating mechanisms are 1) disrupted by external events, 2) absences are discovered, 3) new elements of coordinating are created, 4) new patterns of coordinating are established, and 5) stabilizing patterns of coordinating.

## 3. Method

This study builds on two broad case studies of large-scale development programmes, Alpha and Beta, which investigates how agile methods were adapted in the very large scale. Changing or introducing new ways of coordinating work requires changes at the procedural, structural and even strategic level. Such organizational changes take a relatively long time [35]. Therefore, to understand coordination in a large-scale agile project we have studied how coordination changes over time in the two case studies. Previous studies [17, 36] show how large development programmes dealt with method tailoring, technical architecture, customer involvement and inter-team coordination. We have taken material from two cases and further analysed our data material on coordination, focusing on the use of the group mode (see characteristics of the programmes in Table 1).

Alpha was chosen because practitioners described it as a successful, very large programme that used agile development methods to a large degree. The whole programme was co-located, and coordination mechanisms could be studied in a setting that is well suited for agile methods. The Alpha programme developed a new office automation system for a public department. The programme was managed by the department and involved two main consulting companies as subcontractors in the project development.

Beta was selected as one of Norway's largest IT-programme with an extensive use of agile methods and was partly co-located. The programme involved complex integration among a wide variety of internal and external information systems, involving various stakeholders with divergent interests. Moreover, before starting Beta, the supplier, an international consulting company, had been part of Alpha.

Our study draws on the established tradition with theoretically informed interpretive case studies in information systems [37, 38] and hence aims at following relevant guidelines for such research [39, 40].

### *3.1 Programme context and delivery model*

Both programmes were planned according to a model based on PRINCE2 [41] with distinct phases. The programmes included projects for architecture, business, development and test with project managers. At Alpha, the development project was split into three subprojects, two managed by external consulting companies and one managed by the customer itself. An external consulting company was managing the Beta programme, and there was less involvement from the customer. In both programmes, the software development was conducted using the agile development method Scrum with an iteration length of three weeks. This meant that there would be a demonstration of the product every three weeks, and teams made detailed plans for three-week iterations. Each team was physically placed around a table, with a board showing progress on one side and with space for making notes during discussions on the other side. In both programmes, the teams were physically co-located. The delivery model included the following four phases:

- *Analysis of needs* - This phase started with a walkthrough of the target functionality of a release and identification of high-level user stories. Product owners prioritized the product backlog.
- *Solution description* - The user stories were assigned to epics, and the user stories were described in more detail, including design and architectural choices. User stories were estimated and assigned to a feature team.
- *Construction* - Development and delivery of functionally tested solutions from the product backlog, with five to seven iterations per release.
- *Approval* - A formal functional and non-functional test to verify that the whole release worked according to expectations. This included internal and external interfaces as well as interplay between systems.

To keep the schedule, solution descriptions needed to be ready in time for the teams. This meant that releases were constantly being planned, constructed, and tested (Approval phase). Thus, a team would constantly be engaged in construction for release n, approving delivered functionality in release n-1, and analysing needs for the next release (n+1).

Table 1. Characteristics of the *Alpha* and *Beta* programmes.

| **Characteristic** | **Alpha programme** | **Beta programme** |
|---|---|---|
| Number of people involved at the most | 175 | 120 |
| Number of development teams | 12 | 5 |
| Employees in customer organization | 380 | 7 000 |
| Duration | 5 years | 4 years |
| Product releases | 12 | 3 |

## *3.2 Data collection*

Our data collection started when the programmes were finished, using individual interviews in Beta, group interviews in Alpha and internal and external documents for both cases as shown in Table 2. We analysed the material in a tool for qualitative analysis, focusing on reporting findings related to group mode coordination and how it changed over time.

Table 2. Data collection from the Alpha and Beta programmes.

| **Data source** | **Alpha programme** | **Beta programme** |
|---|---|---|
| Individual interviews | 0 | 27 |
| Group interviews | 9 two-hour interviews with a total of 24 participants | 0 |
| Documents | External experience report | Tender documents |
| | Internal experience report | Project documents such as plans and scope |
| | | IT-strategy documents |

### *3.3 Data analysis*

We imported all interview transcripts and documents into tools for qualitative analysis and did a descriptive and holistic coding [42] of the topic coordination. Units of text ranged from sentences to whole pages and were coded into topics such as “Scrum of Scrums”, “daily meetings”, “table discussions”, “ad hoc conversations” and “coffee breaks”. The results were presented and discussed with the rest of the research team. Given the various topics and backgrounds of the researchers, the level of detail in the coding varied between the research teams.

## 4. Results

In both Alpha and Beta, group mode coordination took place through a number of scheduled meetings as well as unscheduled meetings shown in Table 3. Both types of meetings were seen as important, as one said,

*"I think the combination of scheduled and unscheduled coordination that just appeared was very important" (Scrum master and developer, Alpha).*

We first describe scheduled meetings at the programme and project levels. The programme consisted of several sub-projects. We report on scheduled meetings common for both Alpha and Beta, and those that only existed in one of the programmes. Then, we repeat the structure for unscheduled meetings. We do not describe meetings that only included one team (e.g., Daily Scrum, Retrospective, and team coffee breaks).

Table 3. Examples of scheduled and unscheduled meetings in programmes Alpha and Beta.

| **Examples of meetings** | | **Alpha** | **Beta** | **Description** |
|---|---|---|---|---|
| Scheduled | Metascrum | X | X | A regularly meeting with project managers from the development, architecture, test and the business projects, as well as subproject managers from the development projects. |
| | Scrum of Scrums | X | X | Scrum of Scrum meetings several times a week. One team-member act as "ambassador" to participate with ambassadors from other teams. Scrum masters and project manager attended, and sometimes stakeholders such as product owners and test managers. |
| | Bug board | X | X | Meeting to discuss errors identified and agree on which to correct, and which team should be assigned to do the correction. Test manager, test responsible and sometimes also developers participated. |
| | Architecture meeting | X | X | A regularly meeting for the architects discussing the overall software architecture, establishing architectural guidelines, and for coordinate work between the teams. |
| | Product owner meeting | X | X | A regularly meeting for the Product owners. |
| | Lunch seminars | X | | Seminar where 2–3 people gave short presentations during lunch on topics such as new architectural components, project management or on how to follow up on a team. |
| | Subproject meetings | X | | Meetings within the subproject. |
| | Open Space | X | | A process where all participants suggested topics for discussion, which is made into an agenda and participants are free to join discussion groups of interest. Used per release during parts of the project. |
| | Experience forum | X | | A meeting forum at one subcontractor for Scrum masters, development manager and agile coach focusing on development method. |
| | Ready-to-sprint meeting | | X | Members from different teams to coordinate and uncover interdependencies involved in the following sprint. |

| Examples of meetings | | Alpha | Beta | Description |
|---|---|---|---|---|
| | Task force | | X | Individuals grouping together across teams in order to solve technical problems. |
| Unscheduled | Coffee breaks | X | X | Unscheduled meetings at the coffee machine. |
| | Discussions on team tables | X | X | The teams were organized around tables. Many discussions emerged at the team tables both with team members and team-external people. |
| | Spontaneous Discussions in open work area | X | X | The project with all teams and project management was situated in an open-plan office space. Many of the decisions made in the project were discussed between relevant stakeholders informally in the open work area and then and then officially decided upon in one of the scheduled meetings. |
| | Group chat tool | X | | Instant messaging to all participants was set up after a need was identified in an open space session. Was used for open technical questions but also for social activities such as wine lottery. |

### *4.1 Scheduled meetings*

At the programme level, the only arena where everyone would meet was at the demonstration meetings, which were held every three weeks. In addition, the programme management met two times a week in a forum, which was called "Metascrum". The Metascrum included managers from the main projects and the central programme management, giving attention to "high-level" obstacles to progress and the assessment of risks in the programme. At Alpha, a new arena was introduced well into the programme, the "open space technology". Open space was a way to motivate the whole programme to discuss challenges and improvement initiatives. This included both technical and business topics that people thought «we need to discuss». One result from the open space sessions was that the programme started using a group chat tool, Jabber, described under unscheduled meetings.

In addition, there were separate meetings to identify dependencies in tasks before work was assigned to teams. At Beta, the meetings varied over the nearly four years of development, but meetings concerning overall software architecture, project managers meeting, and project owners meeting were conducted regularly. These meetings involved participants from both the Consultant Company and the Customer. In the later part of the programme, a meeting referred to as the “Bug Board” was also established to coordinate actions for solving critical problems on technical issues, mercantile issues or processual issues.

In Alpha and Beta, at the project level, there were three main types of scheduled meetings: meetings prescribed by the agile method Scrum, meetings in the main projects in the programme, and meetings in fora at the project level to share experiences across the development teams.

Scrum of Scrums were held in the three development subprojects at Alpha and in the main programme at Beta with Scrum masters and subproject managers from 3-6 development teams. Project managers sometimes participated in these meetings. One subproject at Alpha had daily Scrum of Scrum meetings in the beginning but reduced the frequency to three times per week. In this meeting strategic decisions were taken, e.g., on resources. One subproject manager gave an example of a typical discussion in the Scrum of Scrum meeting:

*"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?"*

In addition, retrospectives were sometimes held across teams in the subprojects, but overall this was an activity within each team.

In Alpha, the projects architecture, business and test had meetings with their own staff and the people who held 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 project architecture focused on establishing architectural guidelines but also focused on coordinating work amongst 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 and in meetings teams declared that "this is our central area of work this period, so please limit work in that area" (technical architect).

In Beta, several other meetings for coordinating across teams and roles were also established in the later parts of the programme. Most profoundly, some members from different teams, to coordinate and uncover interdependencies involved in the following sprint, first practised a meeting referred to as "ready-to-sprint". This meeting turned out to be crucial to distribute work in a way that made the different teams work as autonomous units as far as possible. These meetings had different participants as roles and individuals relevant for uncovering and analysing interdependencies varied from sprint to sprint. These meetings first grew out of the pressing need for coordinating across teams experienced by individual team members and were later sanctioned by project managers as a practice to adopt in a more systematic manner.

Another example of how coordinating mechanisms for groups changed over time in Beta is what was referred to as 'task forces'. As the project progressed, individual team members experienced that existing coordinating mechanisms like Scrum of Scrums and architecture meetings were not enough for solving especially complex problems involving interdependencies across teams. These were often highly technical problems relating to for instance security issues, integration with legacy systems, and performance issues. Hence, there was an absence of coordinating mechanisms for handling such emerging problems. The coordinating mechanism referred to by participants as 'task forces' emerged out of individuals grouping together across teams to solve these technical problems. The group meeting could last for several days until a solution was found. As explained by the project manager in the later phase of the project:

*"We had a special task force for solving issues on performance where we had experts how hunted down components that had poor performance" (Project manager, Beta).*

The project also experienced situations where existing coordinating mechanisms that used to work out well collapsed. One example of this was a specialized architecture meeting referred to as the 'Service Oriented Architecture forum'. As explained by the one architect, this suddenly stopped working as a coordinating mechanism:

*"We had a group called the Service Oriented Architecture-forum during the whole project, but over time, it did not work. In the beginning, it worked as a meeting for making decisions regarding the software bus [used in the customer organization], but after a while it stopped working because we were waiting for information – and hence we had poor progress" (Solutions architect, Beta).*

Experience-sharing across teams were the focus of several scheduled meetings at the sub-project level: "Experience forum", "Lunch seminars" and "Technical corner" are examples of meetings that existed during the Alpha programme. A topic discussed at the experience forum was how to liven up the retrospectives. This was then a topic discussed amongst all participants in the development teams in one project. Participation in these meetings was voluntary.

### *4.2 Unscheduled meetings*

Unscheduled meetings were easy to organize due to the open workspace. Unplanned meetings frequently took place around the boards that were available for each team. These were used to "discuss solutions, draw and make sketches" (subproject manager, Alpha). These discussions spanned development teams and roles. The project management was placed on tables so that they could see most of the boards and thus quickly obtain an overview of status of the teams.

If the project managers noticed discussions, they could inquire about the issue and say that,

*"This problem I know was addressed by another team two iterations ago, let us get 'Ola' over here and see if he can help" (Subproject Manager, Alpha).*

A Scrum master and developer stated that they learned "very much" in the programme 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 programme also started to use the group chatting tool (Jabber) to ease informal coordination, what we can see as a type of unscheduled virtual meetings. This tool was introduced during the programme, which enabled asking several people for help without interrupting them. This channel was used for several purposes, as a technical architect from Alpha expressed:

*"It was used ad hoc...to whatever people wanted to use it for...technical things, wine lottery...and to ask «can anyone tell me about a certain topic» - when you do not know exactly who to ask..."* (Technical architect, Alpha).

In the Beta project, as most of the project was co-located, some of the early coordinating mechanisms like the Service Oriented Architecture forum collapsed, and project participants began to 'know the organization', the role of unscheduled meetings increased. Additionally, as the project progressed, more interdependencies were discovered, and thus more coordinating was needed. In this situation, a primary coordinating mechanism emerged in terms of situational unscheduled meetings between only a few project members for a relatively short time (less than an hour). As explained,

*"By and large [coordination] is ad hoc. It was common practice to just walk over to each other [other teams] to discuss and solve issues there and then. And it was also a common understanding that such issues needed to be solved at once. And if [everyone] did so, this would certainly reduce the frictions between teams" (Developer, Beta).*

## 5. Discussion

We have described the use of group mode coordination in two large-scale software development programmes using the agile development method Scrum and planned according to PRINCE2 with distinct phases. We have presented how interdependencies are managed using scheduled and unscheduled meetings at different levels in programme organizations, partially answering one of the questions raised by Jiang et al. [3]. The programmes included projects for architecture, business, development and test. We have relied on Van de Ven et al. [12], who define coordination as "integrating or linking together different parts of an organization to accomplish a collective set of tasks". We found that the group mode (scheduled and unscheduled meetings) was extensively used in the two large-scale agile programmes.

In large-scale agile software projects, a common strategy for coordination across several teams is Scrum of Scrum, in which one team-member acts as "ambassador" to participate with ambassadors from other teams. We found 15 examples of scheduled and unscheduled meetings, which include Scrum of Scrum, backlog meeting, sprint related meetings and workshops (Table 3). These are the same types of multi-team meetings that are recommended by the large-scale agile framework LeSS [32]. In their study of six multi-team projects, Dietrich et al. [2] found the use of 11 coordination mechanisms in the group mode. One explanation that Dietrich et al. reported fewer coordination mechanisms could be that their projects were product development or organizational development projects, which from their descriptions seem less complex than Alpha and Beta. Another explanation could be that both Alpha and Beta are large programmes while five of the six cases studied by Dietrich et al. were smaller projects with a maximum of 40 people. There is a distinct difference between managing a project and managing a programme in that the latter involves more coordination that the former [3].

We now discuss our research question "*How do inter-team group mode coordination mechanisms change over time in large*-scale *agile development?"* through emphasizing how coordination changes over time, and if changes are initiated bottom-up or top-down in the programme organisation.

### *5.1 Inter-team Group mode coordination over time*

While some coordination mechanisms changed over time, the meetings related to the agile method Scrum were kept throughout the programme (e.g., Scrum of Scrums, Meta Scrum, demonstrations and meetings at team level). Furthermore, the iteration length remained at three weeks for both programmes, resulting in many synchronized meetings (e.g., in the Scrum of Scrum, and ready-for-sprint meeting). In addition, people in the programmes were co-located; therefore, coordination could easily emerge (coffee breaks and walking over to other teams). Organizing meetings at the same interval throughout the programme (synchronization) and co-location (structure) was found to support coordination effectively as described by the members of the programme. This is consonant with Strode et al. [22] who found synchronization and structure enhance coordination effectiveness.

We have described two main transitions over time within the group mode: at Alpha, there was a high number of scheduled meetings initially, but a gradual transition to unscheduled meetings. Informants stated that the initial scheduled meetings were very important for the efficient use of unscheduled meetings later and that the unscheduled meetings became more important than the scheduled meetings. The importance of unscheduled meetings is consonant with Van de Ven [12] who found that unscheduled meetings are used to a greater extent than scheduled meetings in larger units and when task uncertainty is high. At Beta, we found a transition from unscheduled to scheduled meetings over time. The main reason for the transition at Beta was that the programme management identified the importance of these unscheduled meetings, and therefore formalized them. Several of the scheduled and unscheduled meetings emerged during the lifetime of the two programmes. Our findings are consistent with those of Jarzabkowski et al. [34], who argue that coordinating mechanisms do not arise as ready-to-use procedures but are constituted as actors go about the process of coordinating. Further, coordinating mechanisms are not stable entities, but emerge through their use in ongoing interactions. Letting coordination mechanisms emerge is also recommended in the LeSS framework. In the LeSS any team or team member should be able and expected to reach out to another team if there is an issue to be solved. Dietrich et al. [2] did not distinguish between scheduled and unscheduled meetings.

We believe that having many meetings was important for inter-team group mode coordination mechanisms to change over time. Many meetings enabled building knowledge and relations among the team early in each of the programmes. Our findings are consonant with the finding of Smite et al. [29] in that many meetings and forums increase the amount and frequency of communication between teams outside of the meetings. Frequent participation in forums and meetings increases the size of a team's social networks and gives the team a good overview of what is going on in the project (ibid). Our findings are also in agreement with Strode et al. [43] who argue that 'knowing who is doing what' and 'knowing who knows what' are two important components of coordination effectiveness.

### *5.2 Group mode changes; top-down and bottom-up*

Group mode coordination mechanisms changed over time in the two programmes. We found both a top-down approach to coordination (mechanisms identified by the programme management) in addition to mechanisms that emerged bottom-up by teams and members in teams. Examples of top-down mechanism were Meta Scrum and Scrum of Scrum. Examples of mechanism that emerged bottom-up was the group chat tool identified in the open space, lunch forums and technical meetings. Top-down initiatives defined by managers were important to establish many group mode coordination mechanisms, which were important for new mechanisms to emerge.

Top-down initiatives can also 'disrupt' existing forms of coordination, and thereby kick off a process of establishing something new. In the current literature on project management this issue is often debated under the heading of 'project governance' [44]. A study by Klakegg et al. [45] argues that approaches to governance of large-scale projects varies from top-down approaches using frameworks from the Association for Project Management to more bottom-up approaches. Both Alpha and Beta are complex agile projects, and therefore need more flexible forms of management focusing on facilitating collaboration and communication, rather than pure top-down approaches to governance. Our findings are consonant with previous research on project management [46].

## 6. Conclusion

Our study supports the finding that group mode coordination is central to achieving inter-team coordination in large programs. In particular, we highlight the role of scheduled and unscheduled meetings to achieve effective coordination. We have shown that the use of these meetings changes over time in two large-scale agile development programs. The transitions have been initiated both bottom-up and top-down in the programme organizations.

When starting a program, we recommend to early identify the important scheduled meetings, as having enough scheduled meetings are important to develop a common understanding of domain knowledge. An answer to the question of "to schedule or not to schedule" would be to ensure a sufficient number of scheduled meetings initially, and then reduce as the coordination needs are handled more informally. When identifying which scheduled meetings to start with in a program, the meetings reported in table 3 can be used as a starting point.

While starting with enough scheduled meetings is important, we believe the unscheduled meetings are of great importance in knowledge work and programme managers should strive to facilitate these meetings. Programme management needs to be sensitive to the vital importance of coordination as well as the coordination needs as they change over time in large programs. Further, program managers need to balance top-down and bottom-up coordination initiatives when changing, terminating and identifying new scheduled and unscheduled meetings.

In future work, we plan to develop a further understanding of the "layered mutual adjustment" we have identified in large-scale software development programmes, and how coordination mechanism emerge, terminate and how they are connected in an ecology of coordinating mechanisms.

**Acknowledgements**

This project 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, the A-team supported by the Research council of Norway through grant 267704, and the companies Kantega, Sbanken, Storebrand and Knowit.

**References**

[1] M. Hoegl and K. Weinkauf, "Managing task interdependencies in Multi-Team projects: A longitudinal study," *Journal of Management Studies*, vol. 42, no.6, pp. 1287-1308, 2005.

[2] P. Dietrich, J. Kujala, and K. Artto, "Inter-team coordination patterns and outcomes in multi-team projects," *Project Management Journal*, vol. 44, no.6, pp. 6-19, 2013.

[3] J. Jiang, G. Klein, and W. Fernandez, "From Project Management to Program Management: An Invitation to Investigate Programs Where IT Plays a Significant Role," *Journal of the Association for Information Systems*, vol. 19, no.1, pp. 40-57, 2018.

[4] R. E. Kraut and L. A. Streeter, "Coordination in software development," *Communications of the ACM*, vol. 38 no.3, pp. 69-82, 1995.

[5] P. Abrahamsson, O. Salo, J. Ronkainen, and J. Warsta, "Agile software development methods: Review and analysis," VTT Technical report, 2002.

[6] T. Dingsøyr, S. Nerur, V. Balijepally, and N. B. Moe, "A decade of agile methodologies: Towards explaining agile software development," *Journal of Systems and Software*, vol. 85, no.6, pp. 1213-1221, 2012.

[7] E. C. Conforto, F. Salum, D. C. Amaral, S. L. da Silva, and L. F. M. de Almeida, "Can agile project management be adopted by industries other than software development?," *Project Management Journal*, vol. 45, no.3, pp. 21-34, 2014.

[8] B. Hobbs and Y. Petit, "Agile Methods on Large Projects in Large Organizations," *Project Management Journal*, vol. 48, no.3, pp. 3-19, 2017.

[9] B. S. Blichfeldt and P. Eskerod, "Project portfolio management – There's more to it than what management enacts," *International Journal of Project Management*, vol. 26, no.4, pp. 357-365, 2008.

[10] T. Dingsøyr, N. B. Moe, T. E. Fægri, and E. A. Seim, "Exploring software development at the very large-scale: a revelatory case study and research agenda for agile method adaptation," *Empirical Software Engineering*, vol. 23, no.1, pp. 490-520, 2018.

[11] B. Boehm and R. Turner, "Management Challenges to Implementing Agile Processes in Traditional Development Organizations," *IEEE Software*, vol. 22, no.5, pp. 30-39, 2005.

[12] A. H. Van de Ven, A. L. Delbecq, and R. Koenig Jr, "Determinants of coordination modes within organizations," *American sociological review* no. 41, pp. 322-338, 1976.

[13] T. Dingsøyr, K. Rolland, N. B. Moe, and E. A. Seim, "Coordination in multi-team programmes: An investigation of the group mode in large-scale agile software development," *Procedia Computer Science*, vol. 121, pp. 123-128, 2017.

[14] S. Bathallath, Å. Smedberg, and H. Kjellin, "Managing project interdependencies in IT/IS project portfolios: a review of managerial issues," *International journal of information systems and project management*, vol. 4, no.1, pp. 67-82, 2016.

[15] A. Elbanna, "Rethinking IS project boundaries in practice: A multiple-projects perspective," *The Journal of Strategic Information Systems*, vol. 19, no.1, pp. 39-51, 2010.

[16] S. Nerur, R. Mahapatra, and G. Mangalaraj, "Challenges of migrating to agile methodologies," *Communications of the ACM*, vol. 48, no.5, pp. 72-78, 2005.

[17] K. H. Rolland, B. Fitzgerald, T. Dingsøyr, and K.-J. Stol, "Problematizing Agile in the Large: Alternative Assumptions for Large-Scale Agile Development," *International Conference on Information Systems*, Dublin, Ireland, 2016.

[18] J. Nuottila, K. Aaltonen, and J. Kujala, "Challenges of adopting agile methods in a public organization," *International Journal of Information Systems and Project Management*, vol. 4, no.3, pp. 65-85, 2016.

[19] T. W. Malone and K. Crowston, "The interdisciplinary study of coordination," *ACM Computing Surveys*, vol. 26, no.1, pp. 87-119, 1994.

[20] M. Hoegl, K. Weinkauf, and H. G. Gemuenden, "Interteam coordination, project commitment, and teamwork in multiteam R&D projects: A longitudinal study," *Organization science*, vol. 15, no.1, pp. 38-55, 2004.

[21] L. Groth, *Future organizational design: the scope for the IT-based enterprise*, New York: John Wiley & Sons, 1999.

[22] D. E. Strode, S. L. Huff, B. G. Hope, and S. Link, "Coordination in co-located agile software development projects," *Journal of Systems and Software*, vol. 85, no.6, pp. 1222-1238, 2012.

[23] P. Xu, "Coordination in large agile projects," *The Review of Business Information Systems*, vol. 13, no.4, pp. 29, 2009.

[24] N. B. Moe, T. Dingsøyr, and T. Dybå, "A teamwork model for understanding an agile team: A case study of a Scrum project," *Information and Software Technology*, vol. 52, no.5, pp. 480-491, 2010.

[25] I. Nonaka and H. Takeuchi, *The knowledge-creating company: how Japanese companies create the dynamics of innovation*. New York: Oxford University Press, 1995.

[26] V. Stray, D. I. Sjøberg, and T. Dybå, "The daily stand-up meeting: A grounded theory study," *Journal of Systems and Software*, vol. 114101-124, 2016.

[27] V. Stray, N. B. Moe, and G. R. Bergersen, "Are Daily Stand-up Meetings Valuable? A Survey of Developers in Software Teams," *International Conference on Agile Software Development*, 2017, pp. 274-281.

[28] H. Nyrud and V. Stray, "Inter-Team Coordination Mechanisms in Large-Scale Agile," *Proceedings of the Scientific Workshop Proceedings of XP2017*, 2017, pp 1-6.

[29] D. Šmite, N. B. Moe, A. Šāblis, and C. Wohlin, "Software teams and their knowledge networks in large-scale software development," *Information and Software Technology*, vol. 8671-86, 2017.

[30] M. Paasivaara, C. Lassenius, and V. T. Heikkila, "Inter-team coordination in large-scale globally distributed scrum: Do Scrum-of-Scrums really work?," *Empirical Software Engineering and Measurement (ESEM), 2012 ACM-IEEE International Symposium on*, 2012, pp. 235-238.

[31] K. H. Rolland, V. Mikkelsen, and A. Næss, "Tailoring Agile in the Large: Experience and Reflections from a Large-Scale Agile Software Development Project," *International Conference on Agile Software Development*, 2016, pp. 244-251.

[32] C. Larman and B. Vodde, *Large-scale scrum: More with LeSS*: Addison-Wesley Professional, 2016.

[33] D. Leffingwell, *SAFe® 4.0 Reference Guide: Scaled Agile Framework® for Lean Software and Systems Engineering*: Addison-Wesley Professional, 2016.

[34] P. A. Jarzabkowski, J. K. Lê, and M. S. Feldman, "Toward a theory of coordinating: Creating coordinating mechanisms in practice," *Organization Science*, vol. 23, no.4, pp. 907-927, 2012.

[35] P. S. Adler and A. Shenhar, "Adapting your technological base:The organizational challenge," *Sloan Management Review*, vol. 32, no.1, pp. 25–37, 1990.

[36] T. Dingsøyr, N. B. Moe, and E. A. Seim, "Coordinating Knowledge Work in Multi-Team Programs: Findings from a Large-Scale Agile Development Program," *Project Management Journal*, 2018.

[37] G. Walsham, "Interpretive case studies in IS research: nature and method," *European Journal of Information Systems*, vol. 4, no.2, pp. 74-81, 1995.

[38] G. Walsham, "Doing interpretive research.," *European journal of Information Systems*, vol. 15, no.3, pp. 320-330, 2006.

[39] H. K. Klein and M. D. Myers, "A set of principles for conducting and evaluating interpretive field studies in information systems," *MIS Quarterly*, vol. 23, no.1, pp. 67-93, 1999.

[40] S. Sarker, X. Xiao, and T. Beaulieu, "Guest editorial: qualitative studies in information systems: a critical review and some guiding principles," *MIS Quarterly*, vol. 37, no.4, pp. iii-xviii, 2013.

[41] C. Bentley, *Prince2: a practical handbook*: Routledge, 2010.

[42] J. Saldaña, *The coding manual for qualitative researchers*: Sage, 2015.

[43] D. E. Strode, B. G. Hope, S. L. Huff, and S. Link, "Coordination Effectiveness In An Agile Software Development Context," *PACIS*, 2011, pp. 183.

[44] S. Pemsel and R. Müller, "The governance of knowledge in project-based organizations," *International Journal of Project Management*, vol. 30, no.8, pp. 865-876, 2012.

[45] O. J. Klakegg, T. Williams, O. M. Magnussen, and H. Glasspool, "Governance frameworks for public project development and estimation," *Project Management Journal*, vol. 39, no.1, pp. 27-42, 2008.

[46] B. Bygstad and G. Lanestedt, "ICT based service innovation–A challenge for project management," *International Journal of Project Management*, vol. 27, no.3, pp. 234-242, 2009.

## Biographical notes

**Nils Brede Moe**
Nils Brede Moe works with software process improvement, intellectual capital, and agile and global software development as a senior scientist at SINTEF. His research interests are related to organizational, socio-technical, and global/distributed aspects. His publications include several longitudinal studies on self-management, decision making, innovation, and teamwork. He has co-edited the books Agile Software Development: Current Research and Future Directions and Agility Across Time and Space: Implementing Agile Methods in Global Software Projects. His thesis was, "From Improving Processes to Improving Practice - Software Process Improvement in Transition from Plan-driven to Change-driven Development". He holds an adjunct position at the Blekinge Institute of Technology in Sweden.

*www.shortbio.org/nils.b.moe@sintef.no*

**Torgeir Dingsøyr**
Torgeir Dingsøyr focuses on software process improvement and knowledge management as chief scientist at the SINTEF research foundation. In particular, he has studied agile software development through a number of case studies, co-authored the systematic review of empirical studies, co-edited the book Agile Software Development: Current Research and Future Directions, and co-edited the special issue on Agile Methods in the Journal of Systems and Software. He wrote his doctoral thesis on Knowledge Management in Medium-Sized Software Consulting Companies at the Department of Computer and Information Science, Norwegian University of Science and Technology, where he is adjunct professor.

*www.shortbio.org/torgeird@sintef.no*

**Knut Rolland**
Knut Rolland focuses on conducting qualitative research in the field of information systems (IS) on various topics and in various organizational settings, as associate professor at the University of Oslo. In particular, he has interest in studying implementation and organizational consequences of corporate-wide digital infrastructures. He got 8 years of experience as a practitioner participating on some of the largest software development projects in Norway. His main research interests are: Digital infrastructures and software platforms, Large-scale IS projects and complexity, IT innovation in organizations and Qualitative research methods.

*www.shortbio.org/knut.rolland@sitnef.no*